Wprowadzenie do Android Studio

Jako nowoczesny użytkownik, Twój wybór produktów może oznaczać różnicę między zmaganiem się a prosperowaniem. To normalne, że zawsze szukamy narzędzi, które zwiększają produktywność i optymalizują pracę. Poszczególne narzędzia mają zalety, które są tak oczywiste, że przyjmuje się je od razu. Jednym z nich jest Android Studio. Świat został wprowadzony do Android Studio zaledwie kilka dni po jego przedpremierowym wydaniu w systemie przetwarzania Google w 2013 roku. Przed premierą używaliśmy narzędzi Android Developer Tools (ADT) zarówno zawodowo, jak i jako narzędzie do nauki. ADT to środowisko programistyczne Androida oparte na zintegrowanym środowisku programistycznym typu open source (IDE) o nazwie Eclipse. Android Studio to efekt doskonałej współpracy JetBrains i Google. Android Studio został zbudowany na IntelliJ JetBrain, co oznacza, że jego funkcjonalność jest nadzbiorem IntelliJ. Prawie wszystko, co możesz zrobić z IntelliJ, powinieneś być w stanie zrobić w Android Studio. Android Studio jest rewolucyjne w tej dziedzinie, ponieważ usprawnia proces tworzenia Androida i sprawia, że tworzenie Androida jest znacznie bardziej przystępne niż wcześniej. Android Studio jest teraz oficjalnym IDE dla Androida. Platforma Android umożliwia programistom skryptowanie kodu zarządzanego przy użyciu języka Java do obsługi i sterowania urządzeniem z systemem Android. W tym samym czasie Android Studio zastąpiło Eclipse jako preferowane IDE do tworzenia aplikacji na Androida. Wcześniej rozwój Androida był rozpowszechniany za pomocą platformy Eclipse z zestawem Android Development Kit (ADK), dostarczonym przez Google, który uruchomił platformę Android Studio. IDE Android Studio ma kilka wielkich zalet, na przykład menedżer zależności Gradle, również oparty na IntelliJ, który jest powszechnie używany na całym świecie. Ta aplikacja jest jedną z największych zalet edytora platformy, ponieważ oferuje programiście więcej opcji w czasie kompilacji. Android Studio można zainstalować w systemach operacyjnych Windows, OSX i Linux. Ponadto samo Google zaleca, aby sprzęt miał co najmniej 4 GB pamięci i 1 GB wolnego miejsca na dysku twardym, ponieważ w przeciwnym razie Android Studio może działać trochę wolno. Powinieneś również mieć zainstalowaną Javę na maszynie za pośrednictwem Java Development Kit (JDK), ponieważ programowanie na Androidzie jest konieczne, a wszystkie klasy programistyczne Java muszą być obecne na komputerze. Przed wydaniem rozwój Androida był zarządzany głównie przez środowisko Eclipse IDE, które oferowało bardziej ogólne środowisko Java IDE niż wiele innych języków programowania. Teraz można śmiało twierdzić, że Android Studio może znacznie ułatwić życie każdemu w porównaniu z oprogramowaniem niespecjalistycznym, ale jednocześnie ma jeszcze długą drogę do przebycia, zanim z pewnością stanie się całkowicie intuicyjnym i płynnym doświadczeniem. Dla zupełnie początkujących jest tu wiele do nauczenia się, a wiele dostępnych informacji – nawet za pośrednictwem oficjalnych kanałów – może początkowo wydawać się zbyt gęste, aby je przyswoić. W tym rozdziale wyjaśnimy bardziej szczegółowo, czym zajmuje się Android Studio, oraz omówimy podstawowe funkcje, których potrzebujesz, aby zacząć. Staramy się, aby wszystko było jak najłatwiejsze i mamy nadzieję, że będzie to pierwszy krok w Twojej podróży do tworzenia Androida.

CO TO JEST STUDIO ANDROID

Ci z was, którzy nie mają wcześniejszego doświadczenia w kodowaniu, mogą się zastanawiać, jaka dokładnie jest rola Android Studio, jeśli chodzi o rozwój i co i tak ma z tym wspólnego IDE? Po pierwsze, obowiązkiem Android Studio jest zapewnienie interfejsu do tworzenia aplikacji i obsługi wielu złożonych procesów zarządzania plikami za kulisami. Językiem programowania, którego będziesz używać, jest Java lub Kotlin. Jeśli zdecydujesz się na Javę, należy ją zainstalować osobno na swoim komputerze. Android Studio to po prostu płótno, na którym będziesz pisać, edytować i zapisywać swoje projekty oraz pliki składające się na te projekty. Jednocześnie Android Studio zapewni dostęp do zestawu SDK (Android Software Development Kit). Należy to postrzegać jako rozszerzenie kodu Java,

które pozwala na płynne działanie na urządzeniach z Androidem i wykorzystanie natywnego sprzętu. Mówiąc prościej, Java jest niezbędna do skryptowania programów, Android SDK jest potrzebny, aby te programy działały na Androidzie, a Android Studio ma za zadanie złożyć to wszystko razem. Z drugiej strony Android Studio umożliwia również uruchomienie kodu przez emulator, dzięki czemu będziesz mógł „debugować" program podczas jego działania i otrzymywać informacje zwrotne wyjaśniające awarie, dzięki czemu możesz szybciej rozwiązać ten problem. Google osiągnął wiele, czyniąc Android Studio tak potężnym i użytecznym, jak to tylko możliwe. Na przykład dodali wskazówki na żywo, dzięki czemu podczas kodowania otrzymasz sugestie dotyczące wprowadzenia niezbędnych zmian, które mogą naprawić błędy lub wyostrzyć kod. Ponadto, jeśli zmienna nie jest używana, zostanie podświetlona na szaro. A gdy zaczniesz pisać wiersz kodu, Android Studio wyświetli listę sugestii autouzupełniania, które pomogą Ci ją dokończyć, co jest świetne, gdy nie pamiętasz poprawnej składni lub po prostu chcesz zaoszczędzić trochę czasu. Ponadto przed zainstalowaniem narzędzia zaleca się sprawdzenie, czy spełniasz następujące podstawowe wymagania systemowe:

Wersja systemu operacyjnego:

* Microsoft Windows 7/8/10 (32- lub 64-bitowy) (Emulator Androida obsługuje tylko 64-bitowy system Windows)

* Mac OS X 10.10 [(Yosemite) lub nowszy, do 10.14 (macOS Mojave)]

* Pulpit GNOME lub KDE [Testowane na gLinuksie opartym na Debianie (4.19.67-2rodete2)].

Pamięć o dostępie swobodnym (RAM): minimum 4 GB RAM; Zalecane 8 GB pamięci RAM.

Bezpłatna pamięć cyfrowa: minimum 2 GB dostępnej pamięci cyfrowej, zalecane 4 GB (500 MB dla IDE + 1,5 GB dla Android SDK i obrazu systemu emulatora).

Minimalna wymagana wersja JDK: Java Development Kit 8

Minimalna rozdzielczość ekranu: 1280 × 800

Android Studio zawiera świetne narzędzia, takie jak Android Virtual Device Manager i Android Device Monitor. Zawiera również zestaw narzędzi Gradle, który pozwala zautomatyzować proces kompilacji i zarządzać nim, co pozwala określić elastyczne niestandardowe konfiguracje kompilacji. Inne kluczowe funkcje Android Studio obejmują:

• Wsparcie dla szybkiego emulatora

• Obsługa wielu szablonów kodu i integracja z GitHub

• Obsługa kreatorów opartych na szablonach do tworzenia projektów i komponentów systemu Android

• Wsparcie dla edytora bogatego układu

•Wsparcie dla głębokiej analizy kodu

•Obsługa szerokiego zestawu narzędzi i frameworków

Ponadto istnieją pewne funkcje edytora kodu IntelliJ, które mogą zwiększyć Twoją produktywność podczas tworzenia aplikacji na Androida, takie jak:

•Ujednolicone środowisko, w którym można tworzyć programy na wszystkie urządzenia z systemem Android

•Zastosuj zmiany w celu wypychania kodu i zmian zasobów w uruchomionej aplikacji bez ponownego uruchamiania aplikacji

•Narzędzia Lint do przechwytywania wydajności, użyteczności, zgodności wersji i innych problemów

• Obsługa C++ i Native Development Kit (NDK)

•Wbudowana obsługa Google Cloud Platform, ułatwiająca integrację Google Cloud Messaging i App Engine

GŁÓWNE CECHY

Każdy projekt rozpoczęty w Android Studio zawiera jeden lub więcej modułów z plikami kodu źródłowego i plikami zasobów. Standardowe typy modułów obejmują moduły aplikacji na Androida, moduły biblioteki i moduły Google App Engine. Zwykle Android Studio wyświetla pliki projektu w widoku projektu Androida, aby zapewnić szybki dostęp do kluczowych plików źródłowych projektu. W związku z tym wszystkie pliki kompilacji powinny być widoczne na najwyższym poziomie w skryptach Gradle, a każdy moduł aplikacji powinien mieć następujące foldery:

• Manifests zawiera plik AndroidManifest.xml.

• Java przechowuje pliki kodu źródłowego Java, w tym kod testowy JUnit.

• Res przechowuje wszystkie zasoby niebędące kodem, takie jak układy XML, ciągi interfejsu użytkownika i obrazy bitmapowe.

Architektura projektu systemu Android na dysku różni się od typowej spłaszczonej reprezentacji. Możliwe jest również dostosowanie widoku plików projektu, aby skupić się na określonych aspektach tworzenia aplikacji. Na przykład wybranie widoku Problemy w projekcie wyświetla łącza do plików źródłowych zawierających wszelkie rozpoznane błędy w kodowaniu i składni, takie jak brakujący znacznik zamykający element XML w pliku układu.

Interfejs użytkownika

Główne okno Android Studio składa się z kilku głównych obszarów logicznych:

• Pasek narzędzi umożliwia wykonywanie szerokiego zakresu czynności, takich jak uruchamianie aplikacji i uruchamianie narzędzi systemu Android.

• Pasek nawigacyjny pomaga zrewidować projekt i wybrać określone pliki do edycji. Daje bardziej zwarty widok całej struktury widocznej w oknie projektu.

• Okno edytora to miejsce, w którym piszesz i edytujesz kod. W zależności od aktualnego typu pliku, edytor można modyfikować. Na przykład podczas przeglądania pliku układu edytor wyświetla Edytor układu.

• Pasek okna narzędzi biegnie na zewnątrz okna IDE i zawiera przyciski, które umożliwiają rozwijanie lub zmniejszanie poszczególnych okien narzędzi.

• Okna narzędzi zapewniają dostęp do określonych zadań, takich jak zarządzanie projektami, wyszukiwanie i kontrola wersji. Możesz je także rozszerzać lub zmniejszać.

• Pasek stanu służy do przeglądania stanu projektu oraz środowiska IDE, w tym wszystkich ostrzeżeń i komunikatów.

Pamiętaj, że możesz zorganizować główne okno, aby zapewnić sobie więcej miejsca na ekranie, ukrywając lub przesuwając paski narzędzi i okna narzędzi. Możesz także wstawić skróty klawiaturowe, aby uzyskać dostęp do większości funkcji IDE. Ponadto w dowolnym momencie możesz aktywować opcję wyszukiwania w kodzie źródłowym, bazach danych, akcjach i elementach interfejsu użytkownika, po prostu dwukrotnie naciskając klawisz Shift lub klikając ikonę lupy w prawym górnym rogu okno Android Studio. Może to być bardzo pomocne, jeśli próbujesz zlokalizować konkretną funkcję IDE, o której zapomniałeś aktywować.

Okna narzędziowe

Zamiast korzystać z gotowych opcji, Android Studio podąża za oryginalnym kontekstem, aby móc automatycznie wyświetlać odpowiednie okna narzędzi podczas pracy. Domyślnie najczęściej wyszukiwane okna narzędzi są przypięte do paska okna narzędzi na krawędziach okna aplikacji.

• Aby rozwinąć lub zwinąć okno narzędzia, kliknij nazwę narzędzia na pasku okna narzędzi. W tym miejscu możesz także przeciągać, przypinać, odpinać, dołączać lub odłączać okna narzędzi.

• Aby powrócić do bieżącego domyślnego układu okna narzędzi, kliknij opcję Okno > Przywróć układ domyślny lub dostosuj układ domyślny, klikając opcję Okno > Zapisz bieżący układ jako domyślny.

• Aby pokazać lub ukryć cały pasek okna narzędzia, kliknij ikonę okna w lewym dolnym rogu okna Android Studio.

• Aby zlokalizować określone okno narzędzia, przejdź nad ikoną okna i wybierz okno narzędzia z menu.

Alternatywnie możesz także użyć skrótów klawiaturowych do otwierania okien narzędzi

Okno narzędzi: Windows i Linux: Mac

1. Projekt : Alt+1 : Command+1

2. Kontrola wersji: Alt+9: Command+9

3. Uruchom: Shift+F10: Control+R

4. Debugowanie: Shift+F9: Control+D

5. Logcat: Alt+6: Command+6

6. Wróć do edytora : Esc : Esc

7. Ukryj wszystkie okna narzędzi: Control+Shift+F12: Command+Shift+F12

Jeśli chcesz ukryć wszystkie paski narzędzi, okna narzędzi i karty edytora, kliknij Widok > Wejdź w tryb bez rozpraszania uwagi. Włączy to tryb bez rozpraszania uwagi. Aby wyjść z trybu bez rozpraszania uwagi, kliknij Widok > Wyjdź z trybu bez rozpraszania uwagi. Możesz także aktywować Szybkie wyszukiwanie, aby wyszukiwać i filtrować w większości okien narzędzi w Android Studio. Aby skorzystać z szybkiego wyszukiwania, wybierz okno narzędzia, a następnie wpisz zapytanie. Ponadto Android Studio ma również trzy rodzaje uzupełniania kodu, do których można uzyskać dostęp za pomocą następujących skrótów klawiaturowych podanych w tabeli . Możliwe jest również wykonanie szybkich poprawek i pokazanie działań intencji, naciskając Alt + Enter. Ponadto przeglądarka Code Sample Browser w Android Studio może pomóc w wyszukiwaniu wysokiej jakości, dostarczonych przez Google próbek kodu dla systemu Android na podstawie aktualnie podświetlonego symbolu w projekcie.

Nawigacja

Opcję nawigacji omówimy szczegółowo w Części 4. Ale podstawowe funkcje obejmują: Możesz przełączać się między ostatnio otwieranymi plikami, stosując akcję Ostatnie pliki. Aby wywołać akcję Ostatnie pliki, naciśnij klawisze Control+E (Command+E na komputerze Mac) i domyślnie zostanie wybrany ostatnio otwierany plik. Możesz również uzyskać dostęp do dowolnego okna narzędzia za pośrednictwem lewej kolumny w tej akcji. Jeśli chcesz zobaczyć strukturę bieżącego pliku, możesz użyć akcji Struktura pliku. Wywołaj akcję Struktura pliku, naciskając klawisze Control+F12 (Command+F12 na komputerze Mac) i szybko przejdź do dowolnej części bieżącego pliku. Możliwe jest również wyszukanie i przejście do określonej klasy w projekcie za pomocą akcji Przejdź do klasy. Wywołaj funkcję, naciskając Control+N (Command+O na Macu) i przejdź do Class obsługuje zaawansowane wyrażenia, w tym garby wielbłąda, ścieżki, nawigację po linii i dopasowanie drugiego imienia. Jeśli wywołasz go dwa razy z rzędu, wyświetli wyniki z klas projektu. Aby przejść do pliku lub folderu, użyj akcji Przejdź do pliku. Aby wywołać akcję Przejdź do pliku, naciśnij klawisze Control+Shift+N (Command+Shift+O na komputerze Mac). Aby szukać folderów, a nie plików, dodaj a/na końcu wyrażenia. Przejdź do metody lub pola według nazwy za pomocą czynności Przejdź do symbolu. Opcję Przejdź do symbolu można aktywować, naciskając klawisze Control+Shift+Alt+N (Command+Option+O na komputerze Mac). Co więcej, aby znaleźć wszystkie fragmenty kodu odwołujące się do klasy, metody, pola, parametru lub instrukcji, możesz nacisnąć Alt+F7 (Option+F7 na Macu).

Styl i formatowanie

Android Studio automatycznie stosuje formatowanie i style zgodnie z ustawieniami stylu kodu, gdy przechodzisz do procesu edycji. Ustawienia stylu kodu można dostosować za pomocą języka programowania, w tym identyfikowania konwencji tabulacji i wcięć, spacji, zawijania i nawiasów klamrowych oraz pustych wierszy. Aby dostosować ustawienia stylu kodu, podążaj ścieżką: Plik > Ustawienia > Edytor > Styl kodu (Android Studio > Preferencje > Edytor > Styl kodu na Macu). Nawet jeśli IDE automatycznie zastosuje formatowanie podczas pisania skryptu, możesz również chętnie Wywołaj akcję Reformatuj kod, naciskając klawisze Control+Alt+L (Opt+Command+L na komputerze Mac) lub automatycznie wcinaj wszystkie wiersze, naciskając klawisze Control+Alt+I (Control+Option+I na komputerze Mac).

Podstawy kontroli wersji

Jak wspomniano wcześniej, Android Studio obsługuje różne systemy kontroli wersji (VCS), w tym Git, GitHub, CVS, Mercurial, Subversion i Google Cloud Source Repositories. Po zaimportowaniu aplikacji do Android Studio, możesz uzyskać dostęp do opcji menu Android Studio VCS, aby włączyć obsługę VCS dla wymaganego VCS, utworzyć repozytorium, zaimportować nowe pliki do kontroli wersji i wykonać inne operacje kontroli wersji. W tym celu w menu Android Studio VCS kliknij Włącz integrację kontroli wersji. Z menu rozwijanego wybierz system VCS do połączenia z katalogiem głównym projektu, a następnie kliknij przycisk OK. Menu VCS wyświetli wtedy szereg opcji kontroli wersji w zależności od wybranego systemu. Można również aktywować VCS za pomocą opcji menu Plik > Ustawienia > Kontrola wersji, aby skonfigurować i zmodyfikować ustawienia kontroli wersji.

System budowania stopni

Android Studio wykorzystuje Gradle jako podstawę systemu kompilacji, a więcej możliwości specyficznych dla Androida zapewnia wtyczka Androida dla Gradle. Ten system budowania działa jako zintegrowane narzędzie z menu Android Studio i niezależnie od wiersza poleceń. Możesz użyć funkcji systemu kompilacji, aby wykonać następujące czynności:

• Dostosowywanie, konfigurowanie i rozszerzanie procesu kompilacji

• Twórz wiele pakietów aplikacji na Androida dla swojej aplikacji, z różnymi funkcjami przy użyciu tego samego projektu i modułów

• Ponownie zastosuj kod i zasoby w różnych zestawach źródłowych

Wykorzystując elastyczność Gradle, możesz wykonać wszystkie te zadania bez konieczności zmiany podstawowych funkcji źródłowych aplikacji. Zwykle pliki kompilacji Android Studio mają nazwę build.gradle. Są one traktowane jako zwykłe pliki tekstowe, które używają składni Groovy do edycji elementów dostarczanych przez wtyczkę Androida dla Gradle. Każdy projekt musi mieć jeden plik kompilacji najwyższego poziomu dla całego projektu i oddzielne pliki kompilacji na poziomie modułu dla każdego modułu. Po zaimportowaniu istniejącego projektu Android Studio automatycznie tworzy niezbędne pliki kompilacji.

Warianty kompilacji

System budowania pomaga w tworzeniu różnych wersji tej samej aplikacji z jednego projektu. Jest to szczególnie przydatne, gdy masz zarówno darmową, jak i płatną wersję swojej aplikacji lub jeśli musisz rozpowszechniać wiele pakietów aplikacji na Androida dla różnych konfiguracji urządzeń w Google Play. Obsługa wielu pakietów aplikacji dla systemu Android Obsługa wielu pakietów aplikacji dla systemu Android umożliwia tworzenie wielu pakietów w oparciu o gęstość ekranu lub interfejs binarny aplikacji. Na przykład możesz utworzyć oddzielną obsługę pakietów aplikacji dla gęstości ekranu hdpi i mdpi, jednocześnie uznając je za pojedynczy wariant i umożliwiając im udostępnianie ustawień javac, dx i ProGuard.

Kurczenie się zasobów

Zmniejszanie zasobów w Android Studio automatycznie usuwa nieużywane zasoby z zależności aplikacji i bibliotek w pakiecie. Jeśli na przykład Twoja aplikacja korzysta z usług Google Play w celu uzyskania dostępu do funkcji Dysku Google, a obecnie nie korzystasz z funkcji Logowanie przez Google, zmniejszanie zasobów może spowodować usunięcie różnych zasobów przycisków logowania.

Zarządzanie zależnościami

Zależności projektu są identyfikowane według nazwy w pliku build.Gradle. Gradle odpowiada za wyszukiwanie Twoich zależności i udostępnianie ich w Twojej kompilacji. W pliku build.Gradle można zadeklarować zależności modułów, zdalne zależności binarne i lokalne zależności binarne. Android Studio konfiguruje projekty tak, aby domyślnie korzystały z centralnego repozytorium Maven, które jest domyślnie zawarte w pliku kompilacji najwyższego poziomu dla projektu.

Narzędzia do debugowania i profilowania

Android Studio pomaga również w debugowaniu i poprawianiu wydajności kodu, w tym w narzędziach do debugowania i analizy wydajności. Należy zastosować debugowanie w wierszu, aby ulepszyć przewodniki po kodzie w widoku debugera z weryfikacją w wierszu odwołań, wyrażeń i wartości zmiennych. Funkcja danych debugowania w linii zazwyczaj obejmuje:

• Wartości zmiennych in-line

• Odwoływanie się do obiektów, które odwołują się do wybranego obiektu

• Zwracane wartości metody

• Wyrażenia lambda i operatory

• Wartości podpowiedzi

Aby włączyć debugowanie w linii, przejdź do okna Debug, kliknij Ustawienia i zaznacz pole wyboru Pokaż wartości w linii.

Profilerzy wydajności

Android Studio udostępnia profilery wydajności, dzięki którym możesz śledzić wykorzystanie pamięci i jednostki centralnej aplikacji, wyszukiwać cofnięte obiekty, lokalizować wycieki pamięci, optymalizować wydajność grafiki i analizować żądania sieciowe.

Zrzut stosu

Podczas profilowania użycia pamięci w Android Studio można jednocześnie zainicjować wyrzucanie elementów bezużytecznych i zrzucić stertę Java do migawki sterty w pliku binarnym A Heap/CPU Profiling Tool (HPROF) specyficznym dla systemu Android. Przeglądarka HPROF wyświetla klasy, instancje każdej klasy oraz drzewo referencyjne, aby pomóc w śledzeniu użycia pamięci i znajdowaniu przecieków pamięci.

Profiler pamięci

Za pomocą programu Memory Profiler można śledzić alokację pamięci i sprawdzać, gdzie umieszczane są obiekty podczas wykonywania określonych czynności. Świadomość tych alokacji umożliwia optymalizację pojemności aplikacji i wykorzystania pamięci poprzez analizę wywołań metod związanych z tymi akcjami.

Dostęp do plików danych

Narzędzia Android SDK, takie jak Systrace i logcat, przeglądają dane dotyczące wydajności i debugowania w celu szczegółowej analizy aplikacji. Aby zobaczyć wszystkie wygenerowane pliki danych, otwórz okno narzędzia Przechwytuje. Na liście wygenerowanych plików kliknij dwukrotnie plik, aby wyświetlić dane. Następnie kliknij prawym przyciskiem myszy dowolny plik .hprof, aby przekonwertować go na standardowy format pliku użycia pamięci RAM.

Inspekcje kodu

Za każdym razem, gdy kompilujesz swój program, Android Studio automatycznie uruchamia skonfigurowany Lint i inne inspekcje IDE, aby pomóc Ci zidentyfikować i naprawić problemy z jakością strukturalną Twojego kodu. Narzędzie Lint koryguje pliki źródłowe projektu Androida pod kątem potencjalnych błędów i ulepszeń optymalizacji pod kątem poprawności, bezpieczeństwa, jakości, użyteczności, dostępności i internacjonalizacji. Oprócz sprawdzania Lint, Android Studio przeprowadza również inspekcje kodu IntelliJ i weryfikuje adnotacje, aby dodać je do przepływu pracy kodowania.

Adnotacje w Android Studio

Android Studio obsługuje adnotacje do zmiennych, parametrów i wartości zwracanych, co pomaga zapobiegać błędom, wyjątkom wskaźnika zerowego i konfliktom typów zasobów. Pakiety Android SDK Manager i biblioteka Support-Annotations w repozytorium Android Support Repository mogą być również używane do sprawdzania poprawności skonfigurowanych adnotacji podczas inspekcji kodu.

Wiadomości dziennika

Po skompilowaniu i uruchomieniu aplikacji za pomocą Android Studio możesz wyświetlić dane wyjściowe adb i komunikaty dziennika urządzenia w oknie Logcat. Jeśli potrzebujesz profilować jednostkę centralną, pamięć i wydajność sieci swojej aplikacji, po prostu uzyskaj dostęp do Android Profiler, klikając Widok > Narzędzia Windows > Android Profiler.

Zaloguj się na swoje konto programisty

Możesz zalogować się na swoje konto programisty w Android Studio, aby sprawdzić dodatkowe narzędzia wymagające uwierzytelniania, takie jak Cloud Tools dla Android Studio i narzędzie testowe App Actions. Logując się, upoważniasz te narzędzia do administrowania i obsługi Twoich danych w usługach Google. Po otwarciu projektu w Android Studio możesz zalogować się na swoje konto programisty lub zmienić konto programisty, klikając ikonę profilu na końcu paska narzędzi, aby się zalogować. W wyświetlonym oknie oczekuje się, że wykonasz jedną z następujące czynności: Jeśli nie jesteś jeszcze zalogowany, kliknij Zaloguj się i zezwól Android Studio na dostęp do wymienionych usług. Lub, jeśli jesteś już zalogowany, kliknij Dodaj konto, aby zalogować się na inne konto Google. Możesz też kliknąć Wyloguj się i powtórzyć poprzednią procedurę, aby zalogować się na inne konto.

KRÓTKA HISTORIA ANDROID STUDIO

Historia sukcesu Androida sięga 2003 roku, kiedy Andy Rubin, Rich Miner, Nick Sears i Chris White współzałożyli start-up o nazwie Android Inc. w Palo Alto w Kalifornii. Później firma musiała zmierzyć się z niewystarczającymi inwestycjami, które spowodowały pojawienie się Google. Google znał prawdziwy potencjał produktu i podpisał umowę o wartości 50 milionów dolarów na zakup Androida w 2005 roku. W ten sposób wszyscy czterej współzałożyciele trafili do Googleplexu i dalej rozwijali system operacyjny pod rządami nowych właścicieli. Pierwsza publiczna wersja beta systemu Android Beta 1.0 została wkrótce opublikowana 5 listopada 2007 r. Android to wyjątkowa platforma obliczeniowa oparta na systemie operacyjnym Linux. Pierwsza komercyjna wersja Androida przejęła rynek w 2008 roku w postaci platformy telefonu komórkowego, kiedy najpopularniejszym telefonem komórkowym dla użytkownika biznesowego był BlackBerry, kiedy iPhone miał właśnie dokonać znaczących kroków we wszystkich sektorach, i kiedy większość użytkowników telefonów wciąż pisała SMS-y z telefonu z klapką. Od tego czasu Android z pewnością stał się jednym z najpopularniejszych systemów operacyjnych na świecie pod każdym względem. Pomimo ogromnej popularności nowoczesnej i modnej platformy Apple iPhone, dostawy Androida na całym świecie znacznie przewyższają oferty Apple. I podczas gdy urządzenia Apple nadal wymagają coraz wyższych cen, produkty z systemem Android skalują globalny rynek. Z pewnością są super drogie modele z Androidem umieszczane obok najnowszego iPhone′a, ale są też stosunkowo niedrogie telefony i tablety z Androidem dostępne w sprzedaży w sklepach masowych. Co więcej, wraz z dojrzewaniem Androida trafia do różnych urządzeń, w tym telewizorów, projektorów i samochodów. Najnowsza wersja obejmuje interfejs z ekranem dotykowym opartym na systemie Android do manipulowania elementami sterującymi w domu i systemem sterowania opartym na systemie Android pojazdu rekreacyjnego. Żeby było jasne, istnieje wiele tego typu interfejsów, które trafiają na rynek. Ta sekcja ma na celu przedstawienie platformy Android i omówienie sposobów jej wykorzystania zarówno w aplikacjach mobilnych, jak i niemobilnych. Celem jest przygotowanie Cię na ścieżce do tworzenia niesamowitych aplikacji na każdą platformę, którą Twoim zdaniem najlepiej będzie wnieść swój wkład. System operacyjny Android jest największą zainstalowaną bazą spośród różnych platform mobilnych na całym świecie. Setki milionów urządzeń mobilnych jest obsługiwanych przez Androida w ponad 190 krajach świata. Do 2020 roku zdobył około 75% udziału w światowym rynku, a trend ten narasta z każdym dniem. Zanim Google przejął całą firmę, konsorcjum firm o nazwie Open Handset Alliance opracowało Androida w oparciu o zmodyfikowaną wersję jądra Linux i inne oprogramowanie typu open source. We wrześniu 2008 roku na rynku pojawiło się pierwsze urządzenie z systemem Android. Android naprawdę dominuje w branży systemów operacyjnych dla urządzeń mobilnych ze względu na długą listę funkcji, które obsługuje. Jest przyjazny dla użytkownika, ma świetne wsparcie społeczności i oferuje większy zakres personalizacji. W rezultacie obserwujemy gwałtowny wzrost zapotrzebowania rynku na tworzenie aplikacji mobilnych na Androida, a wraz z tym firmy poszukują inteligentnych

programistów z odpowiednim zestawem umiejętności. Początkowo celem Androida było mobilny system operacyjny. Jednak wraz z rozwojem bibliotek kodu i jego popularnością wśród programistów z różnych domen, Android staje się absolutnym zestawem oprogramowania dla wszystkich urządzeń, takich jak tablety, urządzenia do noszenia, dekodery, telewizory inteligentne i notebooki. Android to potężny system operacyjny typu open source, który zapewnia przydatne funkcje, w tym:

• Narzędzie Android Open Source Project do dostosowywania wymagań systemu operacyjnego.

• Android obsługuje różne typy połączeń GSM, CDMA, Wi-Fi i Bluetooth do rozmów telefonicznych lub przesyłania danych.

• Zawiera wiele interfejsów API do obsługi usług śledzenia lokalizacji, takich jak GPS.

• Możemy zarządzać wszystkimi czynnościami związanymi z przechowywaniem danych za pomocą menedżera plików.

• Zawiera szeroką gamę obsługiwanych nośników, takich jak AVI, MKV, FLV, MPEG4 itp., do odtwarzania lub nagrywania różnych materiałów audio/wideo.

• Obsługuje również różne formaty obrazów, takie jak JPEG, PNG, GIF, BMP, MP3 itp.

• Obsługuje sterowanie sprzętem multimedialnym w celu odtwarzania lub nagrywania za pomocą kamery i mikrofonu.

• Android ma zintegrowaną przeglądarkę internetową typu open source opartą na układzie WebKit, która obsługuje interfejsy użytkownika, takie jak HTML5, CSS3 itp.

• Android obsługuje wielozadaniowość, co oznacza, że możemy jednocześnie uruchamiać wiele aplikacji i przełączać się między nimi.

• Zapewnia obsługę wirtualnej rzeczywistości lub grafiki 2D/3D.

Google uruchomiło pierwszą wersję platformy Android 5 listopada 2007 roku. Od tego czasu wydało wiele innych wersji Androida, takich jak Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Jellybeans, Kitkat, Lollipop , marshmallow, Nougat i Oreo z dodatkowymi funkcjami i nowymi funkcjami. Poniższa lista przedstawia szczegóły wersji Androida wydawanej przez Google od 2007 roku do dnia dzisiejszego.

Apple Pie: Android 1.0: 23 września 2008 r.

Chleb bananowy: Android 1.1: 9 lutego 2009 r.

Cupcake: Android 1.5: 30 kwietnia 2009 r.

Donut: Android 1.6: 15 września 2009 r.

Eclair: Android 2.0–2.1: 26 października 2009 r.

Froyo: Android 2.2–2.2.3: 20 maja 2010 r.

Gingerbread: Android 2.3–2.3.4: 6 grudnia 2010 r.

Honeycomb: Android 3.0.x–3.2.x: 22 lutego 2011 r.

Ice Cream Sandwich: Android 4.0–4.0.4: 18 października 2011 r.

Jelly Bean: Android 4.1–4.1.2: 9 lipca 2012 r.

KitKat : Android 4.4–4.4.4 : 9 lipca 2012 r.

Lollipop: Android 5.0–5.1: 17 października 2014 r.

Marshmallow: Android 6.0–6.0.1: 5 października 2015 r.

Nugat: Android 7.0–7.1: 22 sierpnia 2016 r.

Oreo: Android 8.0: 21 sierpnia 2017 r.

Pie : Android 9.0: 6 sierpnia 2018 r.

Android P: Android 10.0: 3 września 2019 r.

Android 11: Android 11.0: 8 września 2020 r.

Językami programowania używanymi do tworzenia aplikacji na Androida są Java i Kotlin. Wcześniej Java była uważana za oficjalny język programowania dla Androida. Ale od 2017 Kotlin (opracowany i utrzymywany przez JetBrains) był używany do tworzenia aplikacji na Androida, ponieważ stał się oficjalnym językiem dla Android Development.

ZALETY ROZWOJU ANDROIDA

Ponieważ Android jest systemem operacyjnym typu open source, ma ogromną społeczność wsparcia. Projekt aplikacji na Androida posiada wytyczne od Google, co ułatwia programistom tworzenie bardziej intuicyjnych aplikacji użytkownika. Fragmentacja daje aplikacjom systemu Android większy potencjał do uruchamiania wielu działań na jednym ekranie. Ponadto wypuszczenie aplikacji na Androida w sklepie Google Play jest łatwiejsze w porównaniu z innymi platformami.

WADY ROZWOJU ANDROIDA

Chociaż fragmentacja zapewnia bardzo intuicyjne podejście do doświadczenia użytkownika, nadal ma poważne wady, ponieważ zespół programistów zazwyczaj spędza trochę czasu na dostosowywaniu różnych rozmiarów ekranów smartfonów mobilnych, które są obecnie dostępne na rynku, i aktywowaniu określonych funkcji w aplikacji. Jednocześnie urządzenia z Androidem mogą się znacznie różnić; dlatego testowanie aplikacji staje się jeszcze trudniejsze. A ponieważ tworzenie i testowanie zajmuje więcej czasu i innych zasobów, koszt aplikacji może wzrosnąć, w zależności od złożoności aplikacji. Od momentu wprowadzenia Android jest oferowany na rynku przez wielu graczy na całym świecie i w różnych branżach. Na przykład Samsung mógł stać się wiodącym producentem smartfonów na całym świecie dzięki Androidowi i chociaż jedno urządzenie zaczęło wszystko, urządzenia z Androidem są teraz dostępne praktycznie na każdym rynku na świecie – i to nie tylko dla telefonów komórkowych. To wykracza poza zakres tego rozdziału, ale spróbuj zadać sobie pytanie, czy nie ma korelacji między odnoszącą największe sukcesy firmą internetową/wyszukiwarką na świecie, która jest również siłą napędową najpopularniejszej platformy mobilnej na świecie. Tak więc, jeśli chcesz napisać kod, który może działać i latać dosłownie w dowolnym miejscu na świecie, to wybrałeś właściwy czas na zapoznanie się z Androidem. Patrząc na zakres możliwości Androida, łatwo pomylić go z komputerowym systemem operacyjnym. Należy postrzegać Androida jako środowisko warstwowe, które jest zbudowane na fundamencie jądra Linuksa i zawiera bogatą funkcjonalność w wielu obszarach. Podsystem interfejsu użytkownika ma wszystko, czego można oczekiwać od rozwiniętego środowiska systemu operacyjnego, w tym okna, widoki i widżety do wyświetlania podstawowych elementów, takich jak pola edycji, listy lub listy rozwijane. Przeglądarka jest dostępna zarówno do ogólnego przeglądania stron internetowych, jak i do osadzenia bezpośrednio we własnej aplikacji. W ciągu ostatniej dekady sieć mobilna została znacznie zmieniona przez wprowadzenie smartfonów do

aplikacji konsumenckich i biznesowych, w tym Androida. Android był w stanie objąć takie responsywne technologie internetowe dzięki znacznemu ulepszeniu następujących warstw oprogramowania:

• Aplikacje: wbudowane aplikacje, takie jak telefon, kontakty, przeglądarka i inne. Przyjęcie kilku konkretnych aplikacji różni się w zależności od wersji Androida i producenta. Aplikacje komercyjne z marketplace′ów, takich jak Google Play, Amazon i inne.

• Ulepszanie struktur aplikacji, takich jak menedżer komórek, menedżer lokalizacji, menedżer powiadomień, dostawcy treści, okienka i menedżer zasobów.

• Zwiększanie pojemności bibliotek, takich jak biblioteki graficzne, biblioteki multimediów, biblioteki baz danych, czujniki itp.

• Środowisko wykonawcze systemu Android jest odpowiedzialne za uruchamianie aplikacji i zarządzanie nimi podczas ich uruchamiania. W tym miejscu powinniśmy wspomnieć o jądrze Linuksa, w tym o zasilaniu, systemie plików, sterownikach i zarządzaniu procesami.

Ponadto Android ma szeroki wachlarz opcji łączności, w tym Wi-Fi, Bluetooth, NFC i połączenia komórkowe w każdej sieci, którą możesz sobie wyobrazić. Możesz także uzyskać dostęp do dowolnych usług opartych na lokalizacji, które obsługują popularne aplikacje mapowe i nawigacyjne. Aparaty cyfrowe zasadniczo wycofały się z rynku ze względu na jakość aparatów zainstalowanych w nowoczesnych smartfonach. Android obsługuje wiele kamer z możliwością przechwytywania wideo w pełnym ruchu. Jedną z popularnych klas zastosowań jest widzenie maszynowe, w którym aplikacje wykorzystują kamerę jako urządzenie wejściowe do wykonywania inspekcji produkcyjnych. Instalacja usług głosowych zmieniła również nowoczesne urządzenie z systemem Android w wirtualnego osobistego asystenta. Oczywiście, jeśli możesz to sobie wyobrazić, możesz go zaprogramować na platformie Android. Ważnym punktem środowiska aplikacji na Androida jest to, że aplikacje na Androida były od dawna pisane w języku programowania Java. Wspomnieliśmy już, że Android działa teraz na stosunkowo nowym języku programowania o nazwie Kotlin. Jeśli jednak dopiero zaczynasz, Java jest bezpiecznym miejscem do odkrywania z kilku powodów. Po pierwsze, w sieci dostępne są zasoby Androida z dekady poświęcone Javie. Po drugie, Java to język, który wciąż ma potencjał dla innych platform - w szczególności technologii internetowych po stronie serwera. Jako klasyczny język programowania obiektowego, umiejętności w języku Java nadal mają znaczenie. Chociaż w tym momencie może to nie mieć większego znaczenia, istnieje kilka interesujących szczegółów dotyczących tego, jak aplikacje na Androida powstają z ich podstawowego kodu źródłowego. Kod źródłowy Java jest ustalany i formatowany na kod bajtowy, który reprezentuje logikę aplikacji, ale nie specyficzne wymagania dla konkretnego urządzenia sprzętowego. Podobnie jak tradycyjne środowiska Java, wczesne wersje Androida działały poprzez konwersję tak zwanych kodów bajtowych na kody wykonywalne specyficzne dla sprzętu. Ta procedura konwersji jest zwykle osiągana przy użyciu kompilatora Just-In-Time (JIT) i odbywa się za każdym razem, gdy aplikacja jest wdrażana. Gdy kompilator JIT przekonwertuje kod, jest on wykonywany na maszynie wirtualnej znanej jako Dalvik VM. Ale począwszy od wersji Androida 4.4 (KitKat), maszyna wirtualna Dalvik została usunięta. Kompilator JIT konwertuje kody bajtowe na kod wykonywalny bezpośrednio na urządzeniu w czasie wykonywania, przy każdym uruchomieniu aplikacji. To nowe podejście nazywa się kompilacją z wyprzedzeniem (AOT). W przypadku kompilacji AOT kod bajtowy jest konwertowany tylko raz podczas instalacji aplikacji. Spowalnia to jednorazową czynność instalowania aplikacji, ale daje korzyść w postaci szybszej implementacji w czasie wykonywania, gdy jest to naprawdę konieczne. Dzięki AOT możemy cieszyć się szybszym środowiskiem wykonawczym przy niższych wymaganiach dotyczących pamięci RAM kosztem większej ilości pamięci. Biorąc pod uwagę powolny spadek kosztów przechowywania, wydaje się to dobrym kierunkiem dla aplikacji mobilnych. Jednocześnie, niezależnie od ruchomych elementów

konwersji aplikacji z kodu źródłowego na działający, z perspektywy programowania istnieją pewne podstawowe zasady, które należy zrozumieć przed rozpoczęciem programowania na Androida. Po pierwsze, aplikacja na Androida składa się z co najmniej jednej z następujących czterech klasyfikacji:

• Działania: aplikacja, która ma widoczny interfejs użytkownika, jest implementowana za pośrednictwem działania. Po wybraniu aplikacji na ekranie głównym lub w programie uruchamiającym aplikację rozpoczyna się aktywność.

• Usługi: Możesz użyć usługi dla dowolnej aplikacji, która musi działać przez długi czas, takiej jak monitor sieci lub aplikacja do sprawdzania aktualizacji.

• Dostawcy treści: Najłatwiejszym sposobem myślenia o dostawcach treści jest postrzeganie ich jako serwerów baz danych. Obowiązkiem dostawcy treści jest zarządzanie dostępem do utrwalonych danych, takich jak kontakty w telefonie. Jeśli twoja aplikacja jest bardzo prosta, niekoniecznie musisz tworzyć dostawcę treści, ale jeśli budujesz większą aplikację lub taką, która udostępnia dane wielu czynnościom i/lub aplikacjom, dostawca treści jest zabronionym środkiem dostępu do twoich danych.

• Odbiorniki transmisji: możesz uruchomić aplikację na Androida, aby przetworzyć określony element danych lub odpowiedzieć na zdarzenie, takie jak otrzymanie wiadomości tekstowej.

Aplikacja dla systemu Android jest wdrażana na urządzeniu w pakiecie z plikiem o nazwie AndroidManifest.xml. Ten plik jest wymagany dla każdej aplikacji na Androida i jest w zasadzie arkuszem przepisów, który dokładnie dyktuje systemowi operacyjnemu sposób interakcji z Twoją aplikacją. Plik AndroidManifest.xml zawiera zarówno wymagane nazwy klas, jak i typy zdarzeń, które aplikacja może wykonać, a także wymagane uprawnienia, które aplikacja musi uruchomić. Aby zilustrować na przykładzie, jeśli aplikacja wymaga dostępu do sieci w celu pobrania pliku, uprawnienie to powinno być wyraźnie określone w pliku manifestu. Lub na przykład aplikacja musi uzyskać dostęp do aparatu. Użytkownik również musi to zatwierdzić. Takie podejście polegające w szczególności na regulowaniu użytkownika w zakresie tego, z czego aplikacja będzie korzystać, ma kluczowe znaczenie w dzisiejszych czasach rosnących problemów związanych z prywatnością i bezpieczeństwem. Chociaż możesz szybko zaakceptować licencje na oprogramowanie, ważne jest, aby zwracać szczególną uwagę na to, do czego prosi aplikacja.

ODKRYWANIE STUDIO ANDROID

Najłatwiejszym sposobem rozpoczęcia tworzenia aplikacji na Androida jest zapoznanie się z pakietem aplikacji Android Studio. Możesz pobrać kopię Android Studio na preferowaną platformę (Windows, Mac OS X lub Linux) ze strony programisty Androida. Android Studio zawiera narzędzia do zarządzania wieloma funkcjami specyficznymi dla platformy za pomocą sdkmanager oraz możliwość testowania aplikacji na rzeczywistym urządzeniu lub emulatorze. Istnieją starsze generacje narzędzi programistycznych, w tym ADT, który jest poprzednikiem Android Studio i Eclipse z wtyczką ADT, który był poprzednikiem ADT. Mówiąc dokładniej, Android Studio to trzecia generacja środowiska Android IDE. Istnieją również narzędzia wiersza poleceń i różne łańcuchy narzędzi ciągłej integracji, które umożliwiają tworzenie aplikacji na Androida. Możesz rozważyć rozpoczęcie od kodu Java. Skrypty w języku Java w Android Studio są intuicyjne, ponieważ zapewniają bogate środowisko Java, w tym pomoc kontekstową i uwagi dotyczące sugestii kodu. Po czystym skompilowaniu kodu Java komponenty Android Studio upewnią się, że cała aplikacja jest poprawnie ustawiona, w tym plik AndroidManifest.xml. Wystarczające tło w tym momencie. Rzućmy okiem na proces instalacji i środowisko programistyczne wymagane do zbudowania aplikacji na Androida.