Pytanie 34: Jak obsługiwane jest zarządzanie pamięcią w systemie iOS?

Każdy programista iOS, który jest w grze od jakiegoś czasu, powinien swobodnie porozmawiać na ten temat. Słaba wiedza na temat zarządzania pamięcią może prowadzić do wycieków pamięci, słabej wydajności i rozczarowania menedżerów, inwestorów, programistów i użytkowników. Odpowiedź można zacząć od wskazania, że Swift używa automatycznego liczenia referencji (ARC), co jest zasadniczo takie samo jak w Objective-C. Domyślnie wszystkie odwołania są silnymi odniesieniami. Dlatego mogą wystąpić silne cykle referencyjne, co uniemożliwia ARC zwolnienie pamięci. Można to rozwiązać za pomocą słabych odniesień. Ta rozmowa może rozszerzyć się na omówienie nieznanych referencji. Nieposiadane referencje są używane w wartościach, które zawsze mają być inne niż zero i dlatego muszą być zdefiniowane jako nieobowiązkowe. Innym możliwym aspektem rozmowy może być dyskusja na temat zamknięć.

Pytanie 35: Co wiesz o singletonach? Gdzie byś użył, a gdzie nie?

Singleton to po prostu klasa, która zezwala tylko na jedną instancję. Nie możesz ponownie uruchamiać instancji klasy pojedynczej. Jednym z powodów, które często pojawiają się w Internecie, jest klasa „logowania". W takim przypadku można użyć singletona zamiast pojedynczego wystąpienia klasy, ponieważ klasa rejestrowania zwykle musi być używana w kółko przez każdą klasę w projekcie.

Pytanie 36: Jak zazwyczaj tworzysz networking?

Ta rozmowa może być bardzo interesująca i ujawnić wiele spostrzeżeń na temat podejścia programisty. Deweloper powinien wspomnieć o architekturach, których używał i wzorcach, na których się opiera. Mogą to być warstwy usług, MVVM, powiązanie danych interfejsu użytkownika, iniekcja zależności lub funkcjonalne programowanie reaktywne. Z jakich bibliotek korzysta programista? Deweloperzy pochodzą z różnych środowisk i mogą używać AFNetworking, ReactiveCocoa…

W jaki sposób zapewnia możliwość zapisywania danych z sieci? Jak wygląda proces od momentu kliknięcia przez użytkownika składnika interfejsu użytkownika do momentu zapisania danych lokalnie na urządzeniu? Jakie zajęcia dotyczą wybranej architektury? W jaki sposób deweloper przygotowałby aplikację do użytku w trybie offline i czy podchodzi do buforowania? Jak zdefiniowałby idealny interfejs API na urządzenia mobilne. Czy zna wszystkie metody HTTP (PUT, POST, GET, DELETE) oraz kiedy i jak ich użyje? Nie ma dobrych / złych odpowiedzi na poprzednie pytania. Stanowią raczej doskonałą okazję do przedyskutowania z doświadczonym potencjalnym współpracownikiem podejść, które również mogą Ci się przydać, porównując własne procesy z procesami potencjalnego pracodawcy.

Pytanie 37: Jak pobrać JSON z serwera WWW, serializować go i zapisać w lokalnej pamięci masowej?

To pytanie częściowo wywodzi się z poprzedniego. Tutaj programista mógłby porozmawiać o platformach, które można wykorzystać. NSJSONSerialization to framework dostarczony przez Apple, ale ma kilka błędów i ograniczeń. W szczególności ma pewne problemy z walidacją i konwersją danych. Jeszcze lepszą odpowiedzią byłoby wspomnieć o bibliotekach innych firm (myśląc tutaj o ObjectMapper lub Mantle). Ponadto istotna byłaby dyskusja na temat tego, jak logicznie oddzielić proces JSON do jednostki logicznej i jednostki logicznej do magazynu.

Pytanie 38: Jakie wzorce projektowe są Ci znane w systemie iOS i których używasz?

Każdy programista, z którym rozmawiasz, powinien wiedzieć o MVC. To jest paradygmat, na którym zbudowany jest iOS. Im wyższy staż pracy ma programista, tym więcej frameworków będzie mógł omówić. Jej e, możesz dołączyć MVVM, który pomaga programistom zapobiegać kontrolerom Massive View. Programista mógłby również wyjaśnić różnice, zalety i wady różnych frameworków.

Pytanie 39: Jak radzisz sobie z zadaniami asynchronicznymi?

W celu obsługi zadań asynchronicznych iOS udostępnia mechanizm znany jako Grand Central Dispatch. Aby z tego skorzystać, musisz utworzyć kolejkę (w tym kontekście jest to podobne do wątku) i przekazać blok do metody dispatch_async (), która zostanie wykonana w tle. Jednak w iOS można zastosować kilka innych mechanizmów.

• Oddzwaniania

• Globalne kolejki

• Pamięć

• Wiele zadań / bloków

Deweloper może chcieć omówić, kiedy można zastosować każdą z tych alternatyw oraz jakie są jej zalety i wady

Pytanie 40: Co to jest kontekst obiektu zarządzanego i jakie funkcje zapewnia?

Kontekst obiektu zarządzanego jest reprezentowany za pomocą wystąpienia klasy NSManagedObjectContext. Kontekst obiektu zarządzanego można rozumieć jako tymczasowy notatnik dla powiązanej kolekcji obiektów. Ten zestaw obiektów reprezentuje spójny widok kilku stałych magazynów. Pojedyncze wystąpienie obiektu zarządzanego istnieje w jednym i tylko jednym kontekście, ale wiele kopii obiektu może istnieć w różnych kontekstach. Kluczowe funkcje zapewniane przez kontekst obiektu zarządzanego obejmują

• Zarządzanie cyklem życia

• Powiadomienia

• Konkurencja

Pytanie 41: Czy możesz porównać i porównać różne sposoby osiągania współbieżności w systemach OS X i iOS?

Zasadniczo istnieją trzy sposoby osiągnięcia współbieżności w systemie iOS.

• Korzystanie z wątków

• Kolejki wysyłkowe

• Kolejki operacji

Jeśli jesteś w stanie zidentyfikować je wszystkie, możesz również omówić ich różne aspekty i różnice. Problem z wątkami polega na tym, że za zaprojektowanie skalowalnego systemu odpowiedzialny jest programista. Musi zdecydować, ile wątków ma zostać utworzonych i zadbać o ręczne dostosowanie tej liczby w zmieniających się warunkach. Za pomocą GCD ta odpowiedzialność za zarządzanie jest delegowana na poziom systemu. Deweloper jest odpowiedzialny za zdefiniowanie zadania, które ma zostać wykonane i dodanie ich do kolejki wysyłkowej. Na ogół może to ułatwić życie. Kolejka operacji jest odpowiednikiem Cocoa równoległej kolejki wysyłania i jest implementowana przez klasę NSOperationQueue. W przeciwieństwie do kolejek wysyłkowych, kolejki operacyjne nie ograniczają się do wykonywania zadań w kolejności FIFO i wspierają tworzenie złożonych wykresów kolejności wykonywania zadań.

Pytanie 42: Jakie są różne tryby tła w iOS?

Poniżej przedstawiono możliwe tryby tła dostępne w iOS:

• Lay audio: aplikacja może odtwarzać dźwięk w tle.

• Aktualizacje lokalizacji: wywołania zwrotne są wyzwalane za każdym razem, gdy zmienia się lokalizacja urządzenia.

• Wykonywanie zadań o skończonej długości: jest to ogólny przypadek „pracy w tle" w systemie iOS, w którym aplikacja działa i wykonuje operację.

• Voice over IP (VoIP): aplikacja działa i wykonuje VoIP w tle.

Pytanie 43: Czy możesz wymienić i wyjaśnić różne typy stanów aplikacji iOS?

• Nie działa: aplikacja nie została uruchomiona lub działała, ale została zamknięta przez system.

• Nieaktywna: aplikacja działa na pierwszym planie, ale obecnie nie odbiera zdarzeń. (Może jednak wykonywać inny kod). Aplikacja zwykle pozostaje w tym stanie tylko przez krótki czas, ponieważ przechodzi do innego stanu.

• Aktywny: aplikacja działa na pierwszym planie i odbiera zdarzenia. Jest to normalny tryb dla aplikacji na pierwszym planie.

• Tło: aplikacja działa w tle i wykonuje kod. Większość aplikacji przechodzi w ten stan na krótko przed zawieszeniem. Jednak aplikacja, która żąda dodatkowego czasu wykonania, może pozostać w tym stanie przez pewien czas. Ponadto aplikacja uruchamiana bezpośrednio w tle przechodzi w ten stan zamiast w stan nieaktywny. Aby uzyskać informacje o tym, jak wykonywać kod w tle, zobacz Wykonywanie w tle https://developer.apple.com/library/archive/documentation/ iPhone / Conceptual / iPhoneOSProgrammingGuide /TheAppLifeCycle / TheAppLifeCycle.html.

• Zawieszona: aplikacja działa w tle, ale nie wykonuje kodu. System automatycznie przenosi aplikacje do tego stanu i nie powiadamia ich wcześniej. Po zawieszeniu aplikacja pozostaje w pamięci, ale nie wykonuje żadnego kodu. Gdy wystąpi stan małej ilości pamięci, system może wyczyścić zawieszone aplikacje bez powiadomienia, aby zrobić więcej miejsca dla aplikacji pierwszego planu.

Pytanie 44: Jakie są różnice między kopiowaniem a zachowaniem?

Ogólnie rzecz biorąc, zatrzymanie obiektu zwiększy liczbę zatrzymań o jeden. Pomoże to zachować obiekt w pamięci i zapobiegnie jego zdmuchnięciu. Oznacza to, że jeśli posiadasz tylko zachowaną wersję obiektu, udostępniasz tę kopię każdemu, kto ci ją przekazał. Kopiowanie obiektu, jakkolwiek to zrobisz, powinno stworzyć kolejny obiekt ze zduplikowanymi wartościami. Pomyśl o tym jak o klonie. Nie udostępniasz klona nikomu, kto ci go przekazał. Szczególnie w przypadku NSStrings możesz nie być w stanie założyć, że ktokolwiek daje ci NSString, naprawdę daje ci NSString. Ktoś może przekazać Ci podklasę (w tym przypadku NSMutableString), co oznacza, że może potencjalnie zmodyfikować wartości pod okładkami. Jeśli Twoja aplikacja zależy od przekazanej wartości i ktoś ją zmieni, możesz mieć kłopoty.

Pytanie 45: Co może wymusić zniszczenie obiektu za pomocą ARC?

Po prostu ustaw zmienne odnoszące się do tych obiektów na zero. Kompilator następnie zwolni obiekty w tym momencie i zostaną one zniszczone, jeśli nie ma innych silnych odniesień do nich.

Pytanie 46: Co się dzieje, gdy wywołujesz metodę na wskaźniku zerowym?

Wiadomość wysłana do obiektu zerowego jest całkowicie akceptowalna w Objective-C. To jest traktowane jako no-op. Nie ma sposobu, aby oznaczyć to jako błąd, ponieważ to nie jest błąd. W rzeczywistości może to być bardzo użyteczna cecha języka.

Pytanie 47: Kiedy syntetyzowanie właściwości jest obowiązkowe?

Kiedy są zadeklarowane w protokołach.

Pytanie 48: Co to jest NSAssert?

Funkcja NSAssert służy do upewnienia się, że wartość jest tym, czym ona jest i miała być. Jeśli asercja się nie powiedzie, oznacza to, że coś poszło nie tak i aplikacja zostaje zamknięta. Jednym z powodów używania NSAssert może być sytuacja, w której masz jakąś funkcję, która nie będzie działać lub będzie powodować bardzo złe efekty uboczne, jeśli jeden z przekazanych parametrów nie jest dokładnie jakąś wartością (lub zakresem wartości). W takim przypadku możesz umieścić NSAssert, aby upewnić się, że wartość jest taka, jakiej oczekujesz. Jeśli tak nie jest, coś jest naprawdę nie tak i aplikacja zostaje zamknięta. NSAssert może być bardzo przydatny do debugowania / testowania jednostkowego, a także gdy dostarcza się frameworki powstrzymujące użytkowników przed robieniem „złych" rzeczy.

Pytanie 49: Co to jest kategoria w systemie iOS?

Kategorie Objective-C pozwalają na rozszerzenie istniejącej klasy za pomocą dowolnej metody, którą uznasz za stosowną. Jest to bardzo przydatne do dodawania metod pomocniczych do innych klas, które pomagają analizować datę w sposób, którego klasa być może nie zamierzała. Możesz również użyć go do podzielenia własnych klas, więc jeśli masz jakąś dużą omni-klasę, która robi kilka rzeczy, możesz wybrać tylko te części, które chcesz, włączając tylko te kategorie.

Pytanie 50: Czego mógłbyś użyć, aby dodać nową metodę do NSString?

W odniesieniu do poprzedniego pytania odpowiedzią byłyby kategorie. Aby dodać nową metodę do NSString, odpowiedzią byłoby użycie kategorii. Na przykład:

@interface NSString (CategoryName)

- (NSString *) aNewMethod;

@end

Pytanie 51: Jakie są Twoje preferencje podczas pisania interfejsów użytkownika: pliki XIB, storyboardy lub programowy UIView?

Istnieją argumenty przemawiające za każdym z różnych modeli. Byłoby arogancją udawać, że mam ostateczną odpowiedź. Zamiast tego, to pytanie ma na celu omówienie z kandydatem jego / jej myślenia i powodów, dla których preferuje tę lub inną alternatywę (lub scenariusz, który pasuje lepiej). Możliwa odpowiedź mogłaby odnosić się do zalet i wad każdego mechanizmu, na przykład:

Zalety plików XIB:

• Możesz szybko stworzyć interfejs użytkownika.

• Zapewnia prostą implementację dla małych aplikacji z minimalną liczbą ekranów.

• Możesz mieć oddzielne pliki XIB dla różnych lokalizacji (tj. Języków lub krajów).

• Świetnie radzą sobie z układaniem elementów i wizualnym wykrywaniem nierówności. Ułatwiają wprowadzanie niewielkich zmian w układzie.

Wady plików XIB:

• Trudno jest scalać konflikty podczas pracy w zespole (trudne do porównywania, łączenia i odczytywania).

• Bardzo dynamiczne widoki są niemożliwe do opisania jako XIB.

• Jeśli chodzi o wydajność, są wolniejsze niż tworzenie widoków za pomocą kodu, ponieważ XIB musi zostać odczytany z dysku i przeanalizowany / przeanalizowany.

• W plikach XIB brakuje dostosowań, które można wykonać w kodzie, takich jak elementy kwarcowe (cienie, zaokrąglone rogi).

• Są trudniejsze do debugowania (tj. Jeśli zapomnisz nawiązać połączenie w programie Interface Builder lub wykonasz niewłaściwe połączenie).

Zalety scenorysów:

• Storyboardy są dobre dla aplikacji z małą lub średnią liczbą ekranów i stosunkowo prostymi wymaganiami dotyczącymi nawigacji między widokami.

• Można stworzyć makietę przepływu aplikacji bez konieczności pisania dużej ilości kodu.

Wady scenorysów:

• Scenorysy nie są kompatybilne z wersjami wcześniejszymi niż iOS 5, więc uniemożliwiają obsługę iOS 4.3.

• Trudno jest pracować równolegle w środowisku zespołowym, ponieważ wszyscy modyfikują ten sam plik.

• Zgodnie z tymi samymi liniami łączenie sprzecznych scenariuszy w GIT będzie uciążliwe.

• W przypadku scenorysów ludzie napotkali błędy w Xcode (np. Konieczność częstego opróżniania folderu DerivedData z powodu niespójności).

Zalety widoków tworzonych programowo:

• Łatwiej jest scalać konflikty i linie różnicowe kodu niż w przypadku pliku XIB.

• Możesz śledzić kod podczas debugowania i nie musisz patrzeć na Interface Builder.

• Pod względem wydajności zapewnia szybsze tworzenie widoków niż pliki XIB.

• Tworzenie widoków za pomocą kodu zapewnia większą kontrolę i swobodę.

Wady widoków tworzonych programowo:

• Trudniej jest wyobrazić sobie interfejs użytkownika i uzyskać wyobrażenie o tym, jak będzie wyglądał, jeśli całe tworzenie interfejsu użytkownika nie odbywa się w jednym miejscu w kodzie.

• Nie możesz wizualnie pozycjonować elementów, co sprawia, że układanie widoków jest bardziej czasochłonne.

• Zrozumienie przepływu aplikacji i nawigacji zajmie nowicjuszom w zespole projektowym więcej czasu.

Pytanie 52: Jak bezpiecznie przechowywać prywatne dane użytkownika offline na urządzeniu? Jakie inne najlepsze praktyki w zakresie bezpieczeństwa należy zastosować?

Nie ma właściwej odpowiedzi na te pytania, ale są one świetnym sposobem sprawdzenia, jak bardzo dana osoba weszła w zabezpieczenia iOS. Tego rodzaju pytań należy się spodziewać, zwłaszcza w przypadku rozmów z bankami lub innymi podmiotami, w których bezpieczeństwo jest najważniejsze. Nie mam ścisłych wytycznych dotyczących odpowiedzi, ale przynajmniej kilka z poniższych tematów powinno zostać omówionych:

• Jeśli dane są wyjątkowo wrażliwe, nigdy nie powinny być przechowywane w trybie offline na urządzeniu, ponieważ wszystkie urządzenia można złamać.

• Pęk kluczy to jedna z opcji bezpiecznego przechowywania danych. Jednak jego szyfrowanie opiera się na kodzie PIN urządzenia. Użytkownicy nie są zmuszani do ustawiania kodu PIN, więc w niektórych sytuacjach dane mogą nawet nie zostać zaszyfrowane. Ponadto kody PIN użytkowników można łatwo zhakować.

• Lepszym rozwiązaniem jest użycie czegoś takiego jak SQLCipher, który jest w pełni zaszyfrowaną bazą danych SQLite. Klucz szyfrowania może być wymuszony przez aplikację i niezależnie od kodu PIN użytkownika.

• Komunikować się tylko ze zdalnymi serwerami przez SSL / HTTPS.

• Jeśli to możliwe, zaimplementuj przypinanie certyfikatów w aplikacji, aby zapobiec atakom typu man-in-the-middle na publiczne Wi-Fi.

• Usuń poufne dane z pamięci, nadpisując je.

• Upewnij się, że wszystkie weryfikacje przesyłanych danych są również uruchamiane po stronie serwera.

Pytanie 53: Czy kiedykolwiek pracowałeś z NSOperationQueue? Możesz to wyjaśnić?

Jednym ze sposobów jednoczesnego wykonywania operacji w systemie iOS jest użycie klas NSOperation i NSOperationQueue. NSOperationQueue reguluje równoczesne wykonywanie operacji. Działa jako kolejka priorytetowa, tak że operacje są wykonywane w przybliżeniu w sposób FIFO, przy czym operacje o wyższym priorytecie (NSOperation.queuePriority) wyprzedzają te o niższym priorytecie. NSOperationQueue może również ograniczyć maksymalną liczbę jednoczesnych operacji do wykonania w dowolnym momencie, korzystając z właściwości maxConcurrentOperationCount.

Pytanie 54: Jak serializowałbyś macierz na dysk?

Klasy kluczy, które mają być używane, to klasa abstrakcyjna o nazwie NSCoder wraz z protokołem o nazwie NSCoding. Razem mają one na celu ujednolicenie procesu konwertowania klas do iz serializowanych formatów danych. Prawdopodobnie natknąłeś się na NSCoding w pewnym sensie, jeśli kiedykolwiek korzystałeś z plików NIB lub scenorysów, ponieważ NSCoding jest podstawowym mechanizmem używanym przez Cocoa do ładowania widoków i kontrolerów widoku z NIB.

Pytanie 55: Jak działa instancetype i do czego jest użyteczny?

instancetype to kontekstowe słowo kluczowe, które może być używane jako typ wyniku do sygnalizowania, że metoda zwraca powiązany typ wyniku. Na przykład:

@interface Person

+ (instancetype) personWithName: (NSString *) nazwa;

@koniec

Aby ocenić krzywą uczenia się kandydata, kolejnym pytaniem może być omówienie podobieństw i różnic między używaniem typu instancji i identyfikatora. instancetype, w przeciwieństwie do id, może być używany tylko jako

typ wyniku w deklaracji metody.

Pytanie 56: Co oznacza termin refleksja?

Odbicie odnosi się do kodu, który jest w stanie sprawdzić siebie lub inny kod w systemie. Cel-C ma refleksję (i klasy abstrakcyjne). Kolejne pytanie do potencjalnego kandydata brzmiałoby: Kiedy refleksja odpowiada naszym potrzebom?

Pytanie 57: Co to są obiekty warstw i co one reprezentują?

Obiekty warstwy to obiekty danych, które reprezentują zawartość wizualną. Obiekty warstw są używane przez widoki do renderowania ich zawartości. Do interfejsu można również dodawać obiekty warstw niestandardowych, aby zaimplementować złożone animacje i inne rodzaje wyrafinowanych efektów wizualnych.

Pytanie 58: W wyliczeniach Swift, jaka jest różnica między wartościami surowymi a powiązanymi wartościami?

Surowe wartości dotyczą sytuacji, gdy każdy przypadek w wyliczeniu jest reprezentowany przez wartość ustawioną w czasie kompilacji. Są podobne do stałych, na przykład:

enum E: Int {

case A // if you don't specify, IntegerLiteralConvertible-based

enums start at 0

case B

}

W tym przypadku A będzie miało wartość 0, a B będzie miało wartość 1. Wartości skojarzone bardziej przypominają zmienne, powiązane z jednym z przypadków wyliczenia.

enum E {

case A(Int)

case B

case C(String)

}

Pytanie 59: Co robi @synthesize?

@synthesize tworzy metodę pobierającą i ustawiającą dla zmiennej. Pozwala to na określenie niektórych atrybutów dla zmiennych, a kiedy @synthesize tę właściwość do zmiennej, generujesz metodę pobierającą i ustawiającą dla zmiennej. Nazwa właściwości może być taka sama jak nazwa

zmiennej. Czasami ludzie chcą, aby był inny, aby użyć go w init lub dealloc lub gdy parametr jest przekazywany z tą samą nazwą zmiennej.

Pytanie 60: Jakie typy kolekcji są dostępne w Swift?

W Swift typy kolekcji są dostępne w dwóch odmianach: Array i Dictionary. Tablica: Możesz utworzyć tablicę jednego typu lub tablicę z wieloma typami. Szybki zwykle woli to pierwsze. Poniżej znajduje się przykład tablicy jednego typu:

var cardName: [String] = ["Robert", "Lisa", "Kevin"]

Aby dodać tablicę, musisz użyć indeksu dolnego

println (CardNames [0]).

Słownik: jest podobny do tablicy skrótów w innym języku programowania. Słownik umożliwia przechowywanie par klucz-wartość i uzyskiwanie dostępu do wartości przez podanie klucza, jak w poniższym przykładzie:

var cards = ["Robert": 22, "Lisa": 24, "Kevin": 26]

Pytanie 61: Co to jest operator niestandardowy w Swift?

Niestandardowe operatory w Swift są przydatne, gdy potrzebujesz zupełnie nowego operatora, zazwyczaj dlatego, że funkcjonalność jest tak wyjątkowa, że żaden z obecnych operatorów nie ma sensu. Kolejnym interesującym pytaniem dla potencjalnego kandydata jest to, jakie typy niestandardowych operatorów istnieją w Swift. Powinien umieć wymyślić następującą listę:

Prefiks: operator jednoargumentowy (działający na jednej wartości), który znajduje się tuż przed wartością, na której działa.

Postfix: jednoargumentowy operator znajdujący się tuż po wartości, na której działa.

Infix: operator binarny (działający na dwóch wartościach) umieszczony między dwiema wartościami, na których działa.

W ramach ćwiczenia z kodowania kandydat może napisać niestandardowego operatora dla konkretnego problemu.

Pytanie 62: Z jakich problemów zdajesz sobie sprawę podczas pracy z blokami?

Bloki w iOS mogą wprowadzać cykle zachowania w określonych okolicznościach. Może się tak zdarzyć, zwłaszcza jeśli nie uda Ci się uniknąć cykli silnych odniesień podczas przechwytywania siebie.

Pytanie 63: Co to jest rozszerzenie iOS?

Szybkie rozszerzenia zasadniczo dodają metody do istniejących typów (klas, struktur i wyliczeń), tylko w kilku różnych „smakach”. Poniższy kod tworzy rozszerzenie dla klasy UIColor. Zauważ, że użyte tutaj słowo kluczowe to rozszerzenie, a nie class, struct lub enum.

extension UIColor: ImportantProtocol,

MoreImportantCustomProtocol {

}

Pytanie 64: Co to jest piaskownica aplikacji?

App Sandbox to technologia kontroli dostępu dostępna w systemie OS X, wymuszana na poziomie jądra. Zaprojektowano go tak, aby zawierał uszkodzenia systemu i danych użytkownika w przypadku naruszenia bezpieczeństwa aplikacji. Aplikacje dystrybuowane za pośrednictwem Mac App Store muszą być zgodne z App Sandbox. Aplikacje podpisane i rozpowszechniane poza sklepem Mac App Store za pomocą identyfikatora programisty mogą (iw większości przypadków powinny) również korzystać z piaskownicy aplikacji. Ze względów bezpieczeństwa iOS umieszcza każdą aplikację (w tym jej preferencje i dane) w piaskownicy podczas instalacji. Piaskownica to zestaw precyzyjnych elementów sterujących, które ograniczają dostęp aplikacji do plików, preferencji, zasobów sieciowych, sprzętu itd. W ramach procesu piaskownicy system instaluje każdą aplikację w swoim własnym katalogu piaskownicy, który działa jako strona główna aplikacji i jej danych.

Pytanie 65: Jak tworzy się aplikacje na iPada i iPhone'a?

Jeśli wybierzesz rodzinę urządzeń jako uniwersalną, podczas tworzenia nowego projektu otrzymasz opcje przygotowania różnych plików XIB lub scenorysów na iPhone'a i iPada. Na poziomie kodu możesz albo sprawdzić urządzenie, albo użyć odpowiedniego nazewnictwa podczas przydzielania go. W czasie wykonywania możesz chcieć załadować niektóre zasoby, w zależności od tego, czy urządzenie to iPhone czy iPad. Z perspektywy kodu programistycznego można postępować w następujący sposób (w języku Swift):

```
switch UIDevice.currentDevice().userInterfaceIdiom {

case .Phone:

// It's an iPhone

case .Pad:

// It's an iPad

case .Unspecified:

//

}
```

Pytanie 66: Jakie są różne tryby tła w iOS?

• Odtwórz dźwięk: aplikacja może kontynuować odtwarzanie i / lub nagrywanie dźwięku w tle.

• Otrzymuj aktualizacje lokalizacji: aplikacja może nadal otrzymywać oddzwonienia, gdy zmienia się lokalizacja urządzenia.

• Wykonywanie zadań o skończonej długości: jest to ogólny przypadek „cokolwiek”, w którym aplikacja może uruchamiać dowolny kod przez ograniczony czas.

• Przetwarzanie pobierania zestawu dla Kiosku: specyficzne dla aplikacji Kiosku, aplikacja może pobierać zawartość w tle.

• Zapewnij usługi Voice over IP (VoIP): aplikacja może uruchamiać dowolny kod w tle. Oczywiście Apple ogranicza jego użycie, więc Twoja aplikacja musi również zapewniać usługę VoIP.