Uniwersytet Jagielloński w Krakowie
Wydział Fizyki, Astronomii i Informatyki Stosowanej

WITOLD KAROL BOBROWSKI

NUMER ALBUMU: 1115454

CZYTNIK KSIĄŻEK ELEKTRONICZNYCH NA iOS

PRACA LICENCJACKA NA KIERUNKU INFORMATYKA

PRACA WYKONANA POD KIERUNKIEM:
dra Karola Przystalskiego
Zakład Technologii Informatycznych

Kraków 2017

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Kraków, dnia Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Kraków, dnia Podpis kierującego pracą

podziękowania

i więcej podziękowań

Streszczenie

streszczenie

Abstract

abstract

Spis treści

1. Wstęp

Mobilny system od Apple, iOS niedługo będzie obchodził 10 lat od wprowadzenia go na rynek. W 2014 roku Apple ogłosiło, że jest ponad miliard aktywnych urządzeń z tym właśnie systemem a dziś jest ich z pewnością znacznie więcej. Każdy kolejny model telefonu komórkowego marki Apple, prezentowany z roczną częstotliwością cieszy się coraz większym powodzeniem. Oprócz nowych urządzeń dostajemy w pakiecie nową odsłone iOS która jest nie tylko udoskonaleniem poprzedniej wersji, ale również jej pełnoprawnym następcą wprowadzając nowy zbiór elementów zarówno funkcjonalnościowych jak i wizualnych. Tak dynamicznie rozwijający się system jest bardzo atrakcyjny dla użytkownika, który dzięki darmowym aktualizacjom dla starszych urządzeń wciąż może cieszyć się najnowszym oprogramowaniem.

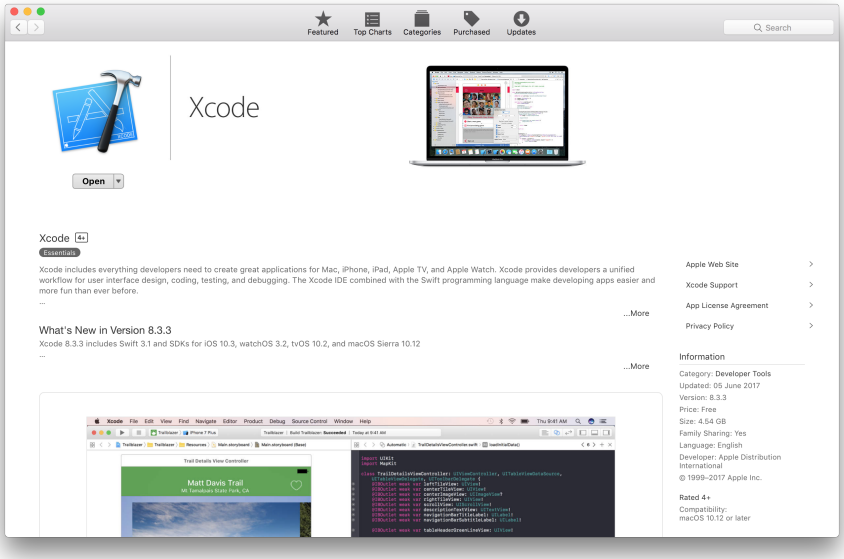
Dla programisty, iOS jest również atrakcyjnym miejscem. Język Swift nad którym prace zostały rozpoczęte w 2010 roku przez Chrisa Lattnera oraz wielu innych programistów z Apple, a w 2014 roku miał swój debiut, dzisiaj jest już głownym językiem programowania mobilnych aplikacji na platformę iOS oraz aplikacji desktopowych na MacOS i wyparł dotychczas używany w tych celach język Objective-C, który swoją historię ma również ściśle związaną z Apple. Korzenie fundamentalnych frameworków z iOS (CocoaTouch) sięgają lat 80 poprzedniego stulecia, a przez ten czas były bardzo silnie rozwijane i wykorzystywane w desktopowym systemie od Apple, MacOS. Nowoczesny język oraz potężne SDK (Software Development Kit) stanowią dziś podstawę pracy z aplilacjami na te platformy. Z roku na rok, wraz z nową wersją systemu, Apple wypuszcza uaktualniego do instniejących API (Application Programming Interface) oraz dostarcza nowe biblioteki zapewniające dostęp do najnowyszych elementów systemu.

Ta praca dokumentuje framework "EPUBKit" którego zadaniem jest pełna obsługa (parsowanie oraz wyświetlanie) książek elektronicznych w formacie EPUB (Electronic Publication) a następnie wykorzystanie go w aplikacji. Rozpoczynając od dokładnego opisu środowiska, wykorzystanych narzędzi, scharakteryzowano format EPUB i jego spycyfikację techniczą, opisano proces tworzenia frameworku, jego strukturę oraz możliwości dystrybucji biblioteki jako moduł gotowy do wykorzystania przez developerów. Następnie w celu demonstracji funkcjonalności frameworku opinasno proces tworzenia aplikacji z jego wykorzystaniem. Celem pracy jest stworzenie prostego i lekkiego narzędzia w języku Swift, które uprości pracę innym programistom tworzącym aplikacje na iOS rozwiązując problem jakim jest obługa formatu EPUB. Na dzień dzisiejszy natywne biblioteki iOS nie zapewniają programistom takiego narzędzia, a publicznie istnieje niewiele rozwiązań, które cieszą się mniejszą lub większą popularnością.

2. Wykorzystane technologie

2.1. Xcode i Developer Tools

Xcode jest IDE (Integrated development environment) stworzonym przez Apple i dostępnym za darmo do pobrania z App Store, sklepu z aplikacjami do którego dostęp mają wyłącznie użytkownicy komputerów z systemem MacOS. Jest wyposażony w pakiet wszystkich narzędzi (Developer Tools) potrzebnych dla developerów aby tworzyć aplikacje na iOS. Główną aplikacją pakietu jest Xcode IDE który wraz z wspomagającymi aplikacjami dostępnymi w pakiecie takimi jak Simulator czy Instruments czyni pracę przy tworzeniu aplikacji płynną i efektowną. W tym rozdziale przedstawię właśnie te narzędzia ze względu na ich rolę w procesie tworzenia aplikacji.

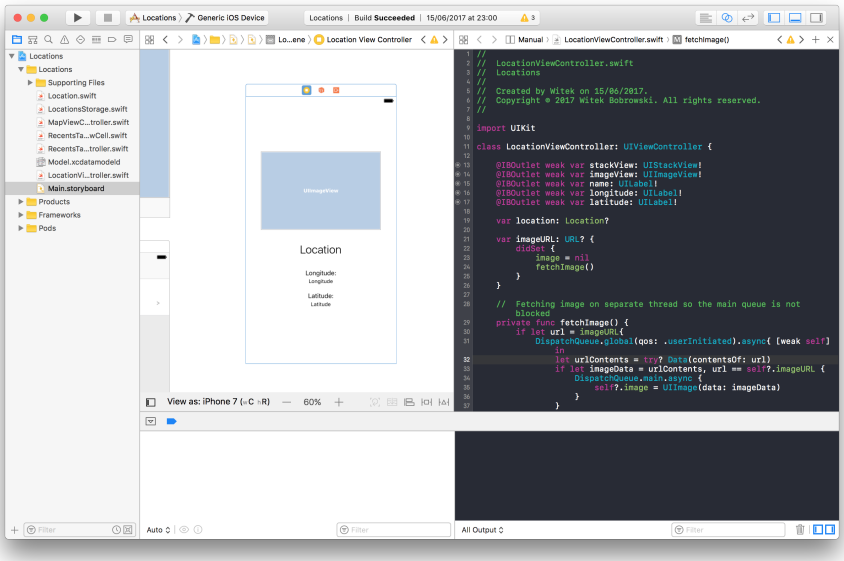


Rysunek 2.1: Xcode w App Store

Xcode IDE

Xcode jako nowoczesne, produktywne środowisko jest miejscem w którym programista aplikacji na iOS spędza większość swojego czasu. Całość prac wykonywanych przy produkcji aplikacji może zostać wykonana właśnie tutaj. Najbardziej podstawowy element jakim jest edytor tekstu dobrze współgra z takimi narzędziami jak Interface Buildier, który pozwala w prosty sposób zaprojektować stronę wizualną aplikacji przy użyciu Storyboardów a następnię stworzyć referencję w kodzie do wybranych przez nas

elementów przez proste przeciągnięcie myszką. Storyboardy są opcjonalnym aczkolwiek bardzo pożytecznym narzędziem szczególnie dla programistów stawiających swoje pierwsze kroki na tej platformie. Zapewniają one wizualne wyobrazenie interfejsu aplikacji nad którą wykonywana jest praca, a projektowanie dowolnego widoku który będzie wyglądał dobrze na każdym urządzeniu w dowolnej orientacji, jest relatywnie proste po zapoznaniu się z kilkoma elementarnymi zasadami.



Rysunek 2.2: Xcode pokazujący "Assistant editor"

Ponieważ Storyboardy tworzy się w jednym pliku o formacie .storyboard, często w profesjonalnej produkcji rezygnuje się z nich ze względu na konflikty w systemach kontrolii wersji. Konflikty te powstają w wyniku pracy wielu programistów, a ponieważ plik .storyboard jest w rzeczywistości plikiem XML, który został wygenerowany automatycznie, rozwiązywanie konfilków bywa kłopotliwe, a przy dużych projektach problematyczne. Dlatego rezygnuje się z nich na rzecz tworzenia widoków tylko przy użyciu kodu, oraz niezależnych plików XIB. Pliki te pozwalają na ustawienie elementów w stylu znanym ze Storyboardów lecz w przeciwieństwie do nich reprezentują pojedyńczy widok, dzięki czemu problem z konfliktami zostaje uniknięty a jednocześnie tworzenie bardziej skomplikowanych widoków pozostaje znacznie ułatwione. Xcode zapewnia wsparcie dla systemu kontroli wersji git. Przy tworzeniu nowego projektu, gdy jest o to poproszony, inicjalizuje nowe repozytorium. Dodatkowo w nawigatorze projektu, w którym widać strukturę projektu, Xcode oznaczy literą "M" pliki które git oznacza jako pliki w których dokonano zmian (modified) a literą "A" pliki które zostały dodane (new file) od czasu poprzedniego zachowania zmian. W najnowszej wersji 9.0, Xcode zyskał nową funkcjonalność - Source Control Navigator, który pozwala na eksplorowanie poszczególnych gałęzi repozytorium i podglądu dowolnego momentu w jego historii.

Simulator

Simulator pozwala na uruchomienie zbudowanej aplikacji na dowolnym urządzeniu z iOS które jest w stanie zasymulować. Gdy chce się przetestować aplikację wystarczy w pasku narzędzi Xcode wybrać

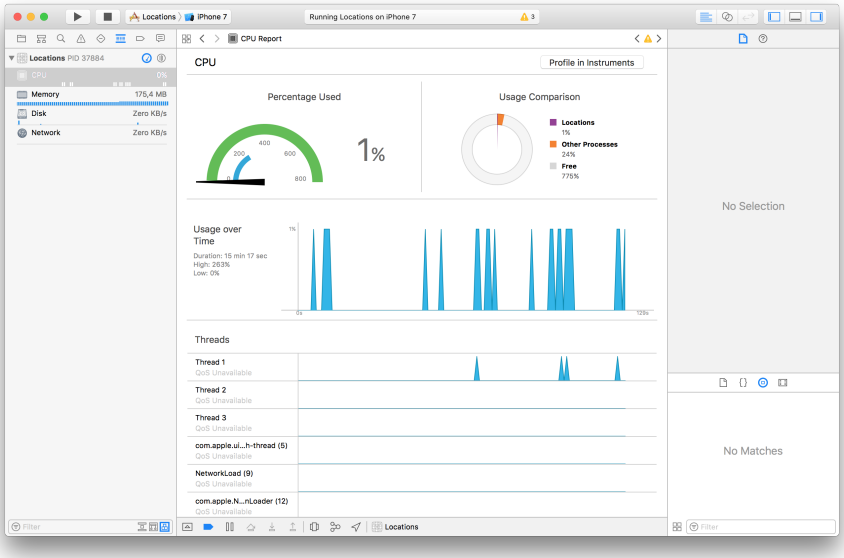
dowolny model urządzenia (oprócz telefonów iPhone znajdują się tam rówież tablety iPad) które chcemy zasymulować (jeżeli podłączymy do komputera fizyczne urządzenie, Xcode rówiez je wykryje i pozwoli na zainstalowanie i uruchomienie na nim aplikacji), a Xcode przejdzie to etapu budowania aplikacji w którym kompiluje pliki źródłowe, a następnie umieści aplikację w symulatorze wybranego przez nas urządzenia.

(a) Wybór docelowego urządzenia

(b) Uruchomiona aplikacja na iPhone 7

Rysunek 2.3: Po wybraniu urządzenia w Xcode, Simulator uruchamia na nim aplikację

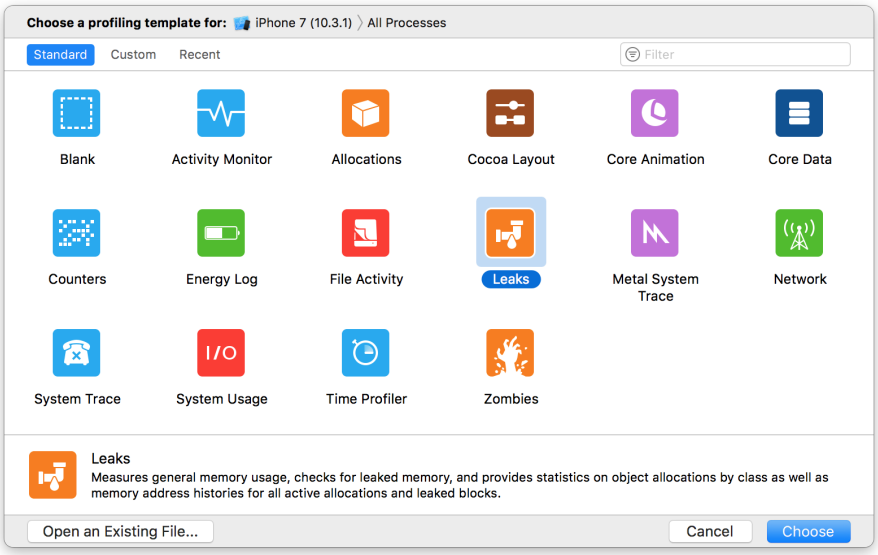
Podczas gdy aplikacja jest uruchomiona i testowana na symulatorze, Xcode pozwala na podgląd użycia zasobów takich jak procesor, pamięć RAM, pamięć dyskowa oraz sieć. Po wybraniu dowolnego z wyżej wymienionych zasobów z nawigatora Debuggera ukazje nam się bardziej szczegółowy podgląd na to w jakim stopniu aplikacja obciąża urządzenie co pozwala na szczegółowe testowanie.



Rysunek 2.4: Użycie CPU na symulatorze odświeżane na bieżąco i wyświetlne w Xcode

Instruments

Widok poboru zasobów w Xcode jest bardzo pomocny na poglądową ocenę wydajności naszej aplikacji, jeżeli jednak chcemy poddać ją prawdziwej próbie, musimy uruchomić kolejne narzędzie jakim jest Instruments. Instruments jest aplikacją dzięki której dokonamy pomiaru nie tylko każdego zasobu na urządzeniu, ale również dostajemy możliwość nadzoru takich aktywności jak alokowanie pamięci dla obiektów, zmiany layoutu widoków czy zmiany w Core Data, natywnej bazie danych dla iOS.



Rysunek 2.5: Instruments, menu główne

Niezależnie od tego jak szybkie i wydajne są w dzisiejszych czasach telefony, optymalny kod nadal jest podstawą prawidłowego działania aplikacji i niedbałe projektowanie architektury może być fatalne w skutkach. Cykle referencji mogą powodować niechciane wycieki pamięci a problemy powstające podczas przerysowywania się widoków spowodują nieczytelny interfejs. Jeżeli takie błędy nie zostaną wykryte na etapie produkcyjnym, a dogłębne testy aplikacji nie zostaną przeprowadzone przed oddaniem jej do recenzji (Aby nasza aplikacja mogła znaleźć się w sklepie AppStore, i być dostępna dla każdego, musi ona otrzymać pozytywną recenzje Apple) aplikacja najprawdopodobniej zostanie odrzucona, co wiąże się z dodatkowymi opóźnieniami. Pakiet narzędzi dostarczanych przez Apple spełnia swoje zadanie i dla większości developerów są one wystarczające. Istnieją rozwiązania firm trzecich, JetBrains dostarcza alternatywne IDE do produkcji aplikacji - AppCode, które cieszy się bardzo dobrą reputacją wśród użytkowników.

2.2. Swift

Hello, world!

Gdy w 1996 roku Apple przejęło NeXT wraz z ich oprogramowaniem, Objective-C stało się głównym językiem do tworzenia oprogramowania na najnowszy system operacyjny OS X. Wraz z OpenStepem Apple znalazło się w posiadaniu takich narzędzi jak Interface Buildier oraz Project Buildier które z

biegiem czasu ewoluowały w Xcode który znamy dzisiaj. Objective-C służył programistom przez wiele lat, na początku tym tworzącym aplikacje desktopowe, a od ukazania się pierwszego iPhone'a, programistom mobilnym. Objective-C jest bardzo mocno rozwiniętym językiem, co zwarzając na jego wiek niepowinno nikogo dziwić. Lecz jako język którego początki sięgają wczesnym latom osiemdziesiątym dwudziestego wieku, z biegiem lat postarzał się, co bardzo upraszczając można by przedstawić stwierdzeniem, iż nie jest wystarczająco nowoczesny.

Twórca LLVM (Low Level Virtual Machine) Chris Lattner, w 2010 roku rozpoczął prace nad nowym językiem który z założenia miał czerpać inspirację z takich języków jak Objective-C, Haskell, Python czy Ruby. Cztery lata później, Apple na WWDC (Worldwide Developer Conference) zaprezentowało Swifta i wypuściło wersję beta. Swift jest bezpiecznym, szybkim językiem, który agreguje wiele paradygmatów znanych z innych nowoczesnych języków. Jego najważniejszymi cechami są automatycznie zarządzana pamięć, typ Opcjonalny który zapewnia wsparcie dla nullowych wskaźników, zaawansowana obługa błędów która zapewnia kontolowane wyjście z niespodziewanych niepowodzeń, zmienne, które zawsze są zainicjalizowane przed użyciem oraz silne typowanie które po połączeniu z nowoczesnym, lekkim syntaxem czyni go językiem zarówno potężnym jak i łatwo dostępnym. Dynamiczny rozwój zawdzięcza inżynierom z Apple oraz społeczności open-source, którzy bardzo intensywnie pracują dodając nowe elementy, naprawiając niedoskonałości oraz optymalizując jego architekturę. Swift dzisiaj jest jednym z najbardziej lubianych języków programowania. Jego popularność wciąż rośnie, a odkąd stał się projektem open-source'owym oprócz systemów Apple, zaczyna być uzywany do takich celów jak tworzenie serwerowych aplikacji. Zdominował platformę iOS i zdetronizował Objective-C, stając się jezykiem pierwszego wyboru. Wszystkie powstające aplikacje są tworzone przy jego użyciu, i wiele napisanych w Objective-C jest dzisiaj przepisywanych na nowy język. Warto wspomnieć, że kod w Objective-C i kod w Swifcie może być użyty w tej samej aplikacji, dzięki czemu może się tam znaleźć również C i C++. Jednak aby użyc kodu napisanego w C czy w C++ w Swifcie musimy go najpierw opakować aby był dla niego dostępny. W przypadku Objective-C, Swift ma dostęp do jego obiektów oraz może dziedziczyć z jego klas. A to zapewnia nam dostęp do wszystkich natywnych bibliotek, czyniąć Swift pełnoprawnym językiem i narzędziem gotowym do użytku już w dniu jego premiery.

Do projektu wykorzystano Swift w wersji 3.1.

2.3. iOS SDK

Jak już wspomniano wcześniej, iOS dziedziczy wiele po desktopowym systemie, MacOS. W skład iOS SDK wchodzą biblioteki znane już od wielu lat oraz biblioteki stworzone z myślą o urządzeniach mobilnych. W tej sekcji omówiono po krótce framework CocoaTouch, który dostarcza elementy interfejsu będące podstawą każdej aplikacji i zapewniajce wizualną spójność platformy.

CocoaTouch

CocoaTouch jest potomkiem frameworku Cocoa dostępnego na systemach MacOS który jest rozszeżony o interfejs obsługi narzędzi dostępnych w urządzeniu mobilnym takich jak rozpoznawanie gestów,

serwis lokalizacji czy obsługa kamery. W skład CocoaTouch wchodzą między innymi biblioteki Foundation, UIKit, MapKit, EventKit i wiele innych. Dzięki temu pakietowi Apple zdefiniowało jak powinny byc tworzone aplikacje na iOS. Dostarczony jest zbiór wielu elementów interfejsu uzytkownika które można dowolnie rozszeżać i modyfikować aby stworzyć unikalny wygląd aplikacji trzymając się wytycznych wyznaczonych przez Apple. Dostęp do gestów zapewni naszej aplikacji lekkość obsługi oraz intuicyjność, a niezliczona ilość innych biblitotek wchodzących w skład CocoaTouch sprawi że aplikacja nabierze życia. Implementowanie funkcjonalności staje się bardzo proste dzięki wysoko poziomowym interfejsom dającym dostęp do poszczególnych elementów systemu oraz fizycznego urządzenia. CocoaTouch jest najbardziej elementarnym frameworkiem na iOS ponieważ to on zapewnia na poziomie postawowym to co potrzebne do stworzenia funkcjonalnej aplikacji. Zakładając że aplikacja powinna agregować, w jakiś sposób przetwarzać a następnie wyświetlić w odpowiedniej formie zbiór pewnych danych, CocoaTouch w parze ze Swiftem to zagwarantują, oczywiście do pewnego stopnia. W większości przypadków gdy potrzebujemy wykonać jakieś specyficzne zadanie, lub chcemy je poprostu uprościć unikając implementacji własnego rozwiązania problemu co było by czasochłonne, będziemy skłaniać się ku zewnętrznym bilbiotekom.

2.4. Zewnętrzne biblioteki

Biblioteki te, tworzone przez środowiska open-sourcowe, pojedyńczych programistów czy wielkie korporacje dadzą nam to czego nie otrzymaliśmy wraz z natywnymi frameworkami. Przykładem jest projekt który opisuje ta praca. Rozwiazuje problem programisty tworzącego własną aplikację króry nie chce tracić czasu na implementowanie funkcjonalości która zajęła by mu relatywnie dużo czasu, a która niekoniecznie jest głównym celem jego aplikacji. Istnieje wiele sposobów na rozszeżenie naszej aplikacji o wybrane moduły, które bardziej szczegółowo zostaną opisane pod koniec czwartego rozdziału podczas omawiania możliwościach dystrybucji framewoku. W tej sekcji przedstawiono framework'i wykorzystane do projektu EPUKit, które nie wchodzą w skład iOS SDK a zostały udostępnione do publicznego użytku.

Zip

Zip jest swift'ową biblioteką open-source dostępną na licencji MIT której kod źródłowy znajdziemy na GitHubie[1]. Zip jest narzędziem do archiwizowania plików oraz rozpakowywania archiwów. Wspiera on formaty archiwów .zip, .cbz oraz daje możliwość dodania rozszeżenia pliku który chcemy rozpakować. W przypadku tego projektu dodano format .epub, aby rozpakować książkę elektroniczną w tym właśnie formacie. Zip bazuje na narzędziu minizip napisanym w języku C, również na wolnej licencji, oraz dostępnym na portalu GitHub. Dzieki bibliotece zip, parser frameworku EPUBKit może w prostu sposób otrzymać dostęp do struktury plików publikacji elektronicznej. Szczegółowe opisanie wykorzystania frameworku Zip znajduje się w czwartym rozdziale podczas omawiania kodu źródłowego parseru.

[1] Adres url: github.com/marmelroy/Zip

AEXML

Kolejną wykorzystaną biblioteką jest AEXML, prosty i lekki pareser XML, dostępny publicznie na licencji MIT, również udostęptniony na GitHubie[2]. Ze względu na strukturę elektronicznej publikacji EPUB, parser xml jest niezbędny do rozpoznania zawartości całej struktury dokumentu, identyfikacji elementów oraz określenia ich lokalizacji. AEXML jest swiftowym frameworkiem a jego API jest czytelne i proste w wykorzystaniu dzięki czemu idealnie spełnia swoje zadanie w projekcie który opisuje ta praca.

[2] Adres url: github.com/tadija/AEXML

3. Charakterystyka EPUB

3.1. Omówienie

EPUB jest standardem formatu dystrybucji cyfrowych publikacji i dokumentów opartych na standardach technologii webowej. EPUB definiuje formę reprezentacji, organizacji struktury oraz kodowania określonej zawartości webowej, na co składają się XHTML, CSS, SVG, obrazy i inne zasoby sprowadzone do formy pojedyńczego pliku. EPUB daje wydawcom możliwość stworzenia cyfrowej publikacji a następnie dystrybuowania go, a odbiorcy łatwy dostęp do pliku niezależnie od urządzenia jakim operuje. Jako następca OEB (Open eBook Publication Structure), zaprezentowanego w 1999 roku, EPUB 2 został ustandaryzowany w roku 2007 a aktualną jego wersją jest EPUB 3.1 (styczeń 2017). Dzisiaj jest on standardem wykorzystywanym na szeroką skale przez wszystkich wydawców. Obok MOBI oraz PDF dominuje rynek, dzięki jego popularności wsród wydawców oraz wsparciu urządzeń. W przeciwieństwie do MOBI które zostało spopularyzowane przez Amazon, właściciela sklepu amazon.com, giganta dystybucji książek elektronicznych oraz producenta czytników elentronicznych marki Kindle, EPUB jest standardem uniwersalnym, nieograniczonym do jednej platformy. EPUB zarówna jak i MOBI charakteryzuje się tym, że jego zawartość nie jest statyczna, co oznacza, że ilość która jest wyświetlana dopasowana jest do wielkośći ekranu urządzenia dzięki czemu jest ona bardziej przyjazna dla odbiorcy (PDF natomiast jest już podzielony na strony których nie da się podzielić). To co najbardziej różni EPUB od MOBI, to wsparcie EPUB dla multimediów (od wersji 3.0) oraz CSS, który stulizuje cały dokument. Dzięki temu jest znacznie bardziej elastyczny i nowoczesny. Popularną praktyką wśród dystrybutorów książek elektronicznych, jest dostarczanie książki klientowi który ją zakupił we wszystkich trzech wcześniej wymienionych formatach. EPUB jako standard jest szeroko udokumentowany dzięki International Digital Publishing Forum[1], grupie która nadrozuje rozwój formatu. W następnej sekcji zostanie szczegółowo opisana struktura formatu EPUB.

3.2. Specyfikacja

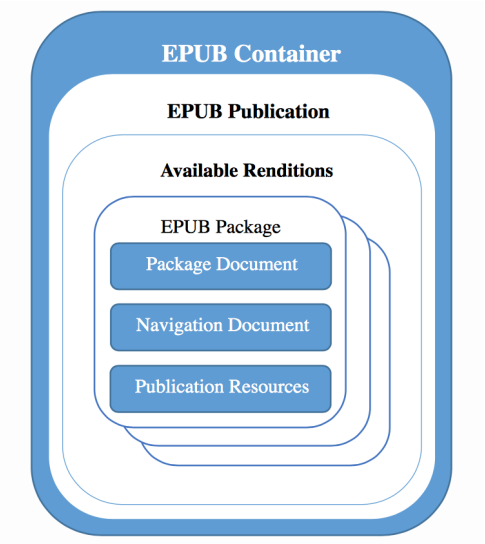
Poniższa specyfikacja jest podzbiorem najważniejszych informacji wyselekcjonowanych ze specyfikacji EPUB 3.1 z dnia 5 stycznia 2017 roku dostępnej na stronie International Digital Publishing Forum[2]. Przedstawiono tutaj najbardziej istotne elementy formatu EPUB w celu zrozumienia problemu jakiego dotyczy projekt EPUBKit.

[1] Adres url: idpf.org

[2] Adres url najnowsej wersji specyfikacji: http://www.idpf.org/epub3/latest

EPUB Open Container Format

Format EPUB definiuje jego najwyższy abstrakcyjny model jakim jest EPUB Publication. Model ten składa się z interpretacji jego zawartości. Interpretacja która jest przedstawiona za pomocą EPUB Package zawiera już bezpośredniozawartość dokumenu, oraz poboczne zasoby mające na celu wspomagać system czytający (z ang. ze specyfikacji "Reading System", jakim jest EPUBKit). Kluczowym elementem jest Package Document który zawiera wszystkie metadane które są używane następnie przez system czytające do zaprezentowania publikacji użytkownikowi. Zawiera on również kompletny manifest zasobów publikacji oraz "kręgosłup" (z ang. ze specyfikacji "Spine"), który reprezentuje sekwencję w jakiej system czytający ma wyświetlać poszczególne elementy. EPUB Package zawiera również Navigation Document pełniący rolę spisu treści, przeznaczony dla użytkownikowa do poruszania się po dokumencie. Wszystko to opakowane jest archiwum ZIP z rozszeżeniem ".epub". Rozszeżenie informuje o charakterze pliku, oraz dostarcza infromacje o archimuw w ZIPowskim stylu za pomocą pliku "mimetype", oraz zapewnia system o posiadaniu przez niego folderu "/META-INF" w krórym dostępny jest plik container.xml, niezbędny systemowi do określenia lokalizacji zawartośći publikacji.



Rysunek 3.1: Wizualna reprezentacja struktury formatu EPUB. źródło: EPUB 3, Recommended Specification 5 January 2017, rozdział 2.2. Road Map

To w jaki sposób zawartość publikacji jest zorganizowana określa standard EPUB Open Container Format (OPF), który definiuje reguły enkapsulacji zasobów w pojedyńczym kontenerze abstrakcyjnym (EPUB Container) zawartym w archiwum ZIP. Struktura OPF to tylko jedna część składająca się na EPUB Publication, druga część to zawartość przedstawiona urzytkownikowi która jest oparta o XHTML oraz SVG. Zawrtość ta jest rozszeżona o wiele dodatkowych zasobów potrzebnych do prawidłowego wyświetlenia publikacji jakimi mogą być obrazy, pliki audio lub video, dodatkowe czcionki, skrypty oraz style nazywane w oficjalnej specyfikacji "EPUB Content Documents".

EPUB Content Documents

EPUB Core Media Types

4. Framework EPUBKit

4.1. Tworzenie frameworku na iOS

4.2. Model

4.3. Parser

4.4. Widok

4.5. Dystrybucja

5. Aplikacja demonstracyjna

5.1. Tworzenie aplikacji na iOS

5.2. Wykorzystanie frameworku w aplikacji

5.3. Publikacja

6. Podsumowanie

Bibliografia