***WWSIS@***

***Wrocławska Wyższa Szkoła***

***Informatyki Stosowanej***

Wydział Informatyki

Łukasz Sidorowicz

Nr albumu: 4492

Projektowanie i implementacja aplikacji mobilnej przygotowującej do brązowej odznaki jeździeckiej.

Praca magisterska

Kierunek: Informatyka

Specjalność/specjalizacja: Projektowanie urządzeń mobilnych

Praca wykonana pod kierunkiem:

dr inż. Katarzyna Pentoś

*Wrocław 2016*

1. **Cel i założenia pracy.**

Obecnie w każdym uprawianym sporcie szczególny nacisk kładzie się na bezpieczeństwo. Jeździectwo jest sportem szczególnym, gdyż wymagana jest poprawna współpraca z dużym i silnym zwierzęciem- koniem. Niestety jest to sport urazowy, a olbrzymia ilość wypadków bierze się z niewiedzy jeźdźców. Dążąc do poprawy bezpieczeństwa i poziomu wyszkolenia Polski Związek Jeździecki wprowadził system szkolenia i odznaki jeździeckie. Brązowa odznaka jeździecka jest niezbędna do udziału w zawodach, zaś jej zakres teoretyczny pozwala na znaczne zwiększenie bezpieczeństwa podczas codziennej pracy z koniem.

Jak dotąd nie powstała jednak żadna aplikacja, czy program wspomagający przygotowanie do części teoretycznej egzaminu. Jedynymi materiałami jest wykaz pytań z odpowiedziami i zalecanej literatury.

Dzisiejsza młodzież nie rozstaje się obecnie ze swoimi smartfonami. Tkwi tu duży potencjał na uzupełnienie luki poprzez stworzenie prostego programu, który można by w każdej chwili otworzyć i wrócić do przerabianego materiału.

Celem pracy będzie zaprojektowanie i zaimplementowanie aplikacji mobilnej, w której pytania zostaną podzielone na odpowiednie działy, ułatwiając naukę materiału. Całość zostanie zaimplementowana do bazy danych, a dane z bazy będą pobierane i wyświetlane w programie.

Efektywność działania aplikacji zostanie porównana w badaniach grup osób korzystających z aplikacji z grupami osób korzystających z otrzymanych materiałów drukowanych. W ramach badań zostanie przeprowadzone szkolenie na terenie stajni w Magnicach. Wśród jeźdźców chcących przygotować się do egzaminu zostaną wyłonione grupy o podobnym poziomie wiedzy początkowej i podzielone na takie, które otrzymają materiały drukowane lub aplikację mobilną. Tam zostaną też przeprowadzone ankiety weryfikujące.

Badania będą miały za zadanie pokazać, jakie metody nauczania są najefektywniejsze, czy użycie aplikacji poprawi wyniki uzyskiwane przez zdających.

Wyniki badań zostaną zaprezentowane za pomocą narzędzia Tableau.

1. **Wstęp**
   1. System szkolenia jeździeckiego w Polsce.

W ostatnich latach wzrasta liczba jeźdźców, jak i posiadaczy koni. Niestety nie idzie to w parze ze znajomością zagadnień teoretycznych, takich jak psychologia konia i jego potrzeby wynikające z natury. Prowadzi to do licznych wypadków w stajni i w drodze ze stajni do ujeżdżalni, jak również przykrymi obrazami w postaci okładanych bezmyślnie batami koni przez jeźdźców. Mając to na uwadze od kilku lat wdraża się systematycznie system szkoleniowy mający poprawić obecną sytuację. Podstawą tego systemu jest zmuszenie początkujących adeptów jeździectwa to zapoznania się z podstawowymi zasadami bezpieczeństwa i postępowania z końmi. Chcąc wystartować w zawodach trzeba posiadać minimum brązową odznakę jeździecką[[1]](#endnote-1). Aby startować w konkursach wyższych klas trzeba posiadać odznakę srebrną, zaś aby być szkoleniowcem- złotą.

Rysunek 1 Odznaki jeździeckie[[2]](#endnote-2)



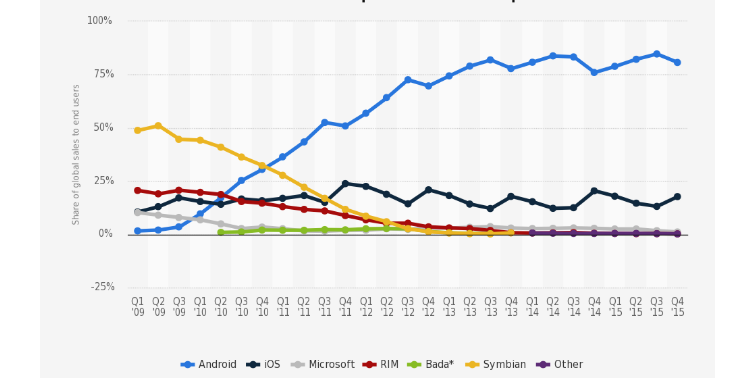
Egzamin na brązową odznakę składa się z części teoretycznej i praktycznej. Zakres części teoretycznej jest dość obszerny i obejmuje praktycznie wszystkie aspekty jeździectwa. Pytania i odpowiedzi można pobrać bezpłatnie ze strony związku. Egzamin praktyczny składa się z części ujeżdżeniowej i skokowej. Jednak dla jeźdźca, który regularnie jeździ konno to tylko formalność- przeszkody są niskie, figury na ujeżdżalni proste. Co innego część teoretyczna. Nie ma możliwości nauczyć tak szerokiego zakresu materiału w trakcie jazdy. Jeźdźcom zaleca się odpowiednie pozycje literaturowe, takie jak „Akademia Jeździecka[[3]](#endnote-3)”, czy też „Zasady Jazdy Konnej[[4]](#endnote-4)”. Jednakże instruktorzy w wielu „ośrodkach” często pomijają aspekty teoretyczne, „instruktorami” są rolnicy nie mający żadnej wiedzy o koniach popełniający skandaliczne ilości błędów w podejściu do młodych adeptów jeździectwa. Odznaki jeździeckie mają za zadanie otworzyć oczy jeźdźcom takich „ośrodków”.

* 1. Systemy operacyjna dla urządzeń mobilnych

Obecnie na rynku systemów operacyjnych dla urządzeń mobilnych liczą się tylko dwaj gracze Apple IOS i Alphabet (wcześniej Google) Android. Do niedawna mówiło się o trzecim poważnym graczu – systemie Microsoftu- Windows Phone.

Jednakże jak przedstawiam na poniższym rysunku praktycznie nie ma on już udziału w rynku nowo sprzedawanych urządzeń:

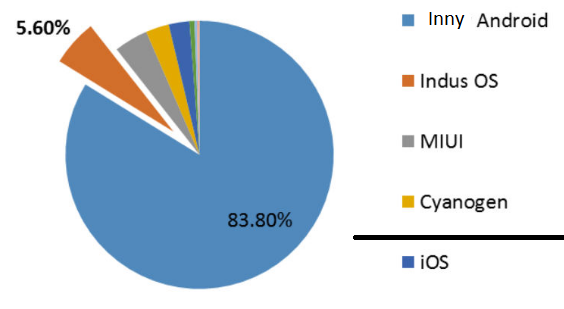
Rysunek 2 Udział procentowy poszczególnych platform mobilnych[[5]](#endnote-5)



Na rysunku przedstawiono procentowy udział sprzedawanych urządzeń wykorzystujących dany system operacyjny. Android wygrywa z kilku względów- wydajny smartfon czy tablet z tym systemem można kupić w cenie poniżej 1000 zł, np. Lenovo Moto G 3 Gen już za 799 zł[[6]](#endnote-6). Aby pozwolić sobie na zakup nowego iPhona z IOSem trzeba liczyć się z wydatkiem co najmniej 3349zł[[7]](#endnote-7).

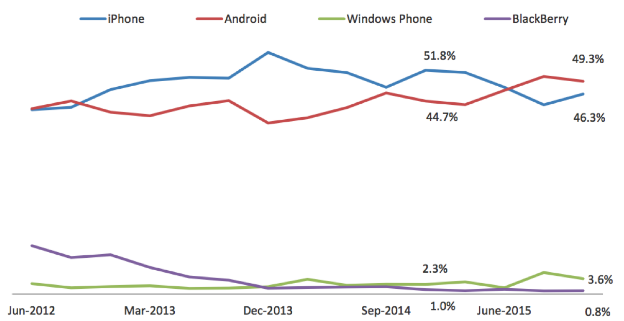
Dysproporcja udziału w rynku pomiędzy Androidem a IOSem jest największa w krajach rozwijających się takich np. jak Indie.

Rysunek 3 Udział Platform Mobilnych w Indiach[[8]](#endnote-8)



W krajach rozwiniętych, takich jak np. Wielka Brytania udziały obu systemów są równe:

Rysunek 4 Udział Platform Mobilnych Wielka Brytania[[9]](#endnote-9)

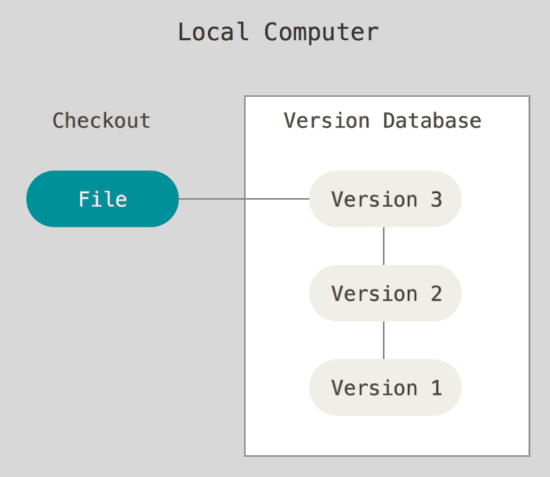


Kolejną ważną zaletą Androida jest darmowy dostęp do zintegrowanego środowiska programistycznego- Android Studio[[10]](#endnote-10) jak i niska cena konta deweloperskiego, która jest jednorazowa i wynosi 25 dolarów[[11]](#endnote-11). W przypadku Apple IOS za wszystko w najtańszej wersji trzeba płacić co roku 99 dolarów[[12]](#endnote-12).

* 1. Systemy kontroli wersji.

System kontroli wersji śledzi zmiany dokonywane na plikach i w folderach. Umożliwia przywrócenie dowolnej wcześniejszej wersji kodu, czy też rysunku jak i projektu. Używanie takiego systemu jest niezbędne w pracy nad każdym większym projektem. Podczas tworzenia aplikacji jedna zła linijka kodu może zniweczyć długą pracę. Dzięki systemowi kontroli wersji można wrócić do wcześniejszego działającego kodu, porównać zmiany i znaleźć dosyć łatwo błąd. Dla wielu osób metodą kontroli wersji jest kopiowanie danego pliku do innego katalogu i nadawanie mu kolejnych numerów, np. draft1, draft2 itd. Nie jest to dobry sposób, po pierwsze można zapomnieć gdzie się kopiowało, ponadto porównanie takich plików jest czasochłonne.

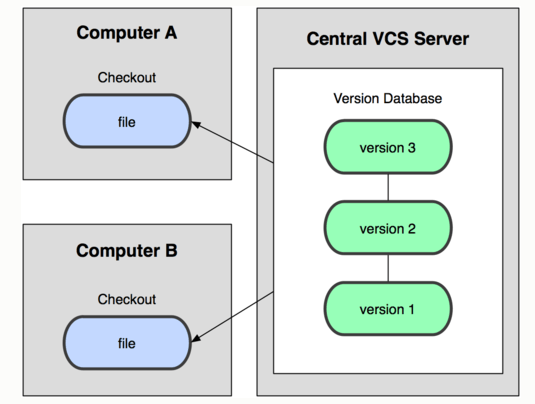
Rysunek 5 Diagram lokalnego systemu kontroli wersji[[13]](#endnote-13)



Aby sobie poradzić z takimi problemami programiści stworzyli lokalne systemy kontroli wersji, który tak naprawdę był lokalną bazą danych, w której przechowywano zmiany dokonane na plikach projektowych. Jednym z pierwszych takich narzędzi był RCS- Revision Control System[[14]](#endnote-14).

Poważnym problemem tego podejścia był brak możliwości dobrej współpracy w zespole, ponadto w przypadku uszkodzenia komputera traciło się całą pracę. Aby przezwyciężyć ten problem powstały centralne systemy kontroli wersji takie np. jak Subversion[[15]](#endnote-15) czy CVS (Concurrent Version System)[[16]](#endnote-16).

Rysunek 6 Diagram scentralizowanego systemu kontroli wersji.

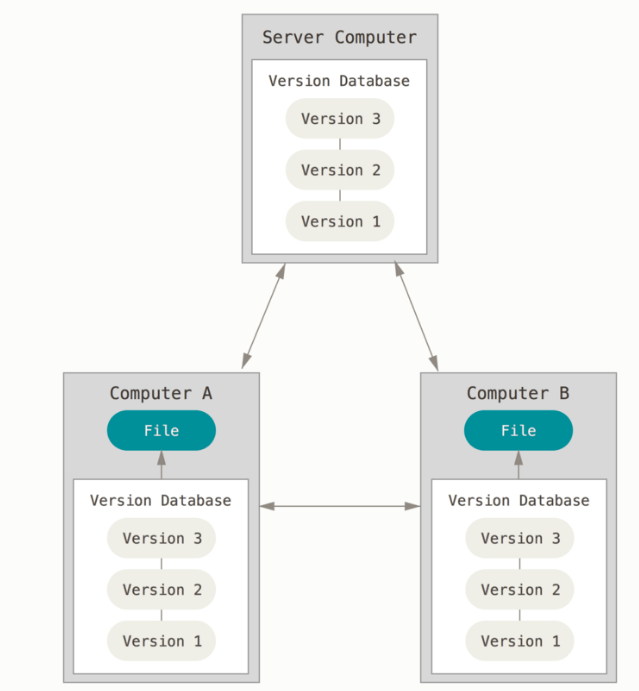


Takie systemy składają się z jednego centralnego serwera, który zawiera wszystkie pliki projektowe. Osoby pracujące w projekcie są klientami i mogą się łączyć z serwerem uzyskując dostęp do aktualnych wersji plików. Ma to olbrzymie zalety w porównaniu z lokalnym systemem- zwiększa się bezpieczeństwo danych, jak również ułatwia współpracę w dużych zespołach. Administratorzy tych systemów mają kontrolę nad uprawnieniami poszczególnych osób w projekcie.

Niestety te systemy mają swoją poważną wadę i słaby punkt- centralny serwer. W przypadku jego awarii, na czas jej trwania zespół traci możliwość współpracy, jak również nikt nie ma możliwość zapisywania zmian, których dokonywał. Co więcej, w przypadku awarii dysku na serwerze można utracić absolutnie wszystko co się robiło od ostatniego backupu.

Tych wad pozbawione są najnowesze systemy kontroli wersji- rozproszone. Do takich systemów należy Git[[17]](#endnote-17), Mercurial[[18]](#endnote-18) i Bazaar[[19]](#endnote-19).

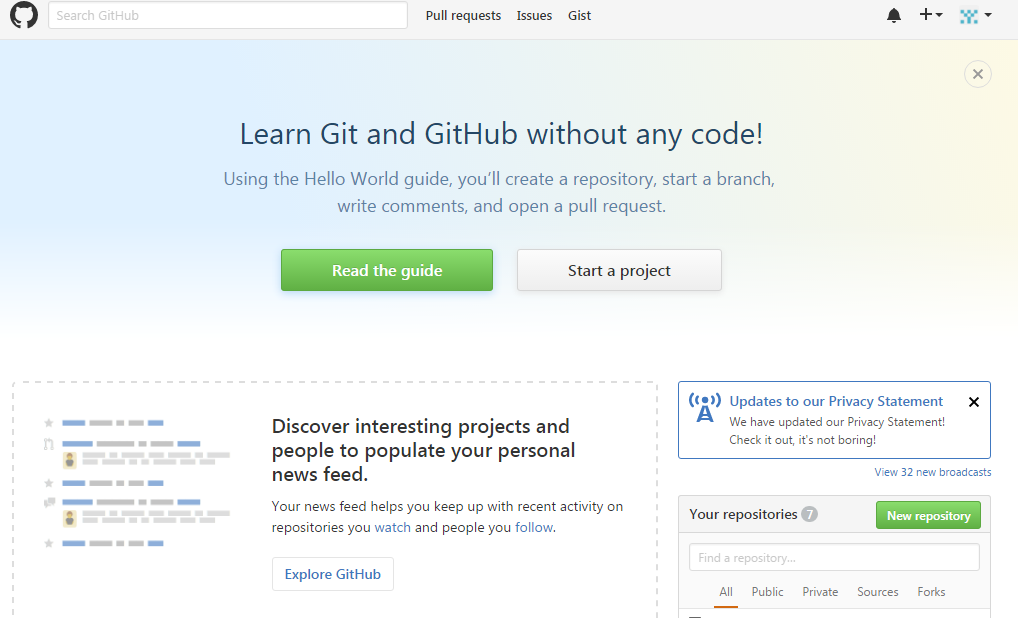
Rysunek 7 Diagram rozproszonego systemu kontroli wersji



W takich systemach klienci kopiują całe repozytorium, jak również mają dostęp do najnowszych wersji plików. W przypadku awarii serwera każdy członek zespołu posiada kopię na swoim komputerze. Ponadto ciągłość pracy zostaje zachowana.

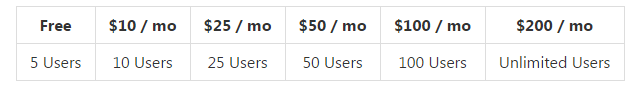
Szczególną popularność zdobył Git, używany od 2005 do utrzymywania wersji jądra Linuxa. Git jest prosty w obsłudze, szybki i wydajny. Z poziomu Gita bardzo łatwo jest skorzystać z internetowego hostingu GitHub[[20]](#endnote-20) jak i Bitbucket[[21]](#endnote-21).

Rysunek 8 Serwis GitHub



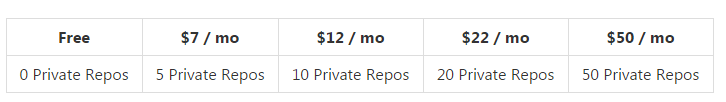
Oba serwisy umożliwiają darmowe tworzenie repozytoriów open source oraz płatne prywatne repozytoria. Wybór, z którego serwisu skorzystać zależy od tego, czy pracujemy w licznym zespole, czy prowadzimy dużo małych projektów. W Bitbucket ma się nielimitowaną liczbę prywatnych repozytoriów, zaś płaci się od liczby użytkowników:

Rysunek 9 Opłaty za korzystanie z serwisu Bitbucket[[22]](#endnote-22)



W GitHubie płaci się od liczby prywatnych repozytoriów niezależnie od liczby użytkowników:

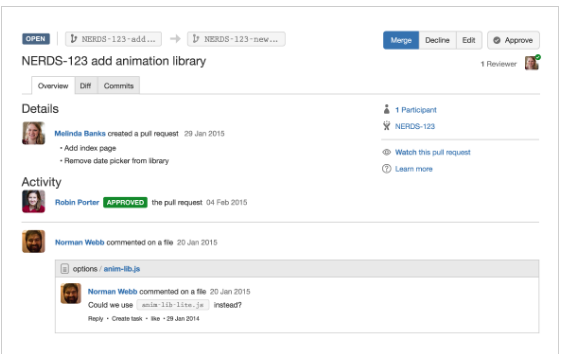
Rysunek 10 Opłaty za korzystanie z serwisu GitHub[[23]](#endnote-23)



Ponieważ funkcjonalność obu serwisów jest podobna, małe zespoły będą decydować się na Bitbucket, ponieważ będą mogły utrzymywać dużą liczbę projektów w niewielkiej cenie, z kolei zespoły wieloosobowe będą preferować GitHub.

Na uwagę zasługuje fakt, iż do użytku na cele np. pracy dyplomowej mogą służyć otwarte repozytoria i wówczas z obu serwisów można korzystać całkowicie za darmo.

Rysunek 11 Wygląd serwisu Bitbucket

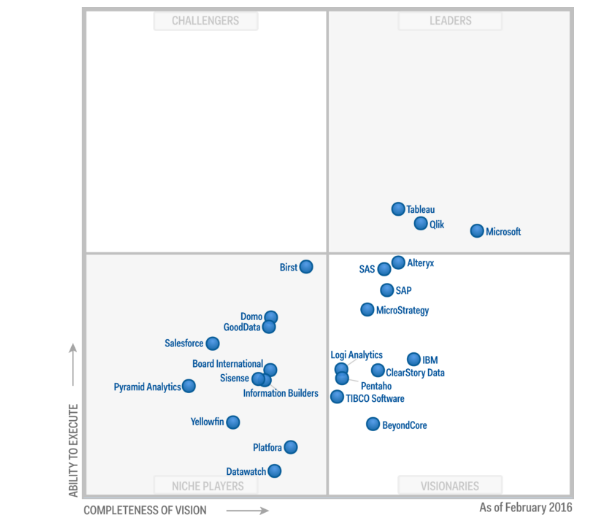


2.4 Narzędzia Business Intelligence

Business Intelligence można najprościej opisać jako proces uzyskiwania wiedzy z danych przekształcanych w informację. Ta wiedza daje przewagę nad konkurencją. Do realizacji tych celów wykorzystuje się raporty bazujące na hurtowniach danych, wydobywanie kluczowych wskaźników działalności przedsiębiorstwa jak również ich wizualizację.

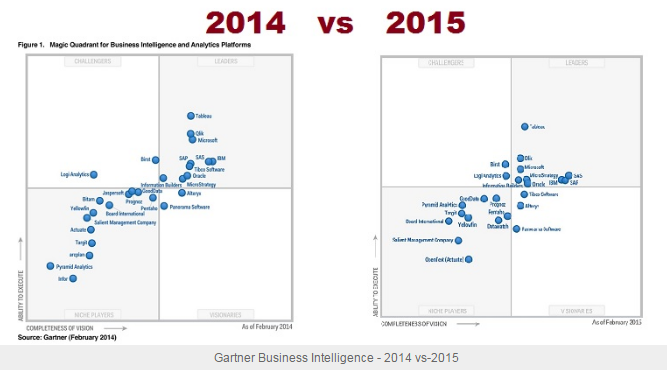
W wielu korporacjach mówi się obecnie, że ten, kto ma dostęp do danych i właściwych narządzi do ich przetwarzania, ten wygrywa. Rynek oprogramowania narzędzi BI jest bardzo dynamiczny, branża silnie się rozwija i zmienia. Siłę danej grupy narzędzi obrazuje magiczny kwadrat Gartnera.

Rysunek 12 Magiczny kwadrat Gartnera 2016



Przedstawia on w zależności od opinii klientów oprogramowania podział na liderów, innowatorów, lubiących wyzwania i małych graczy. Gartner Institute[[24]](#endnote-24) publikuje oprócz tego szczegółowy raport o poszczególnych narzędziach prezentując ich ogólny opis oraz silne i słabe strony. Magiczny kwadrat nie do końca odzwierciedla sytuację rynkową. Niezwykle popularny jest SAP Business Objects, jednak obecnie z całego szerogkiego portfela SAP, Garner do narzędzi BI zalicza tylko SAP Lumira. Uzasadnia to tym, że pozostałe narzędzia takie jak Web Intelligence, Crystal Reports, Dashboards, czy Explorer nie spełniają norm nowoczesnych narządzi BI[[25]](#endnote-25). Nie tylko SAP ma problem aby w porę być wystarczająco innowacyjnym. Z zestawienia Gartnera całkowicie wypadł taki gigant jak Oracle. A jeszcze w latach 2014 i 2015 był zaliczany do liderów.

Rysunek 13 Zmiany w magicznym kwadracie Gartnera w latach 2014-2015[[26]](#endnote-26)

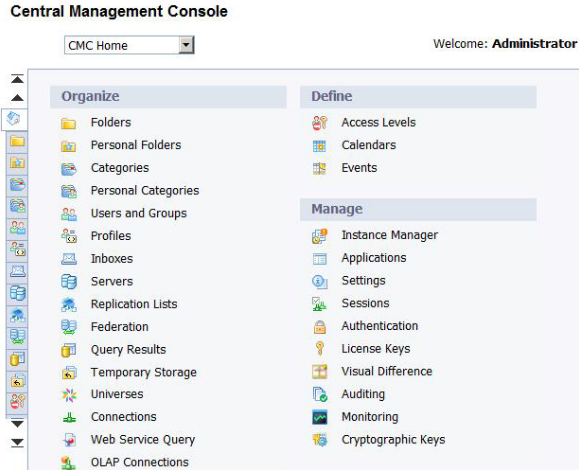


To pokazuje przed jakimi wyzwaniami stoją wytwórcy oprogramowania. Muszą ciągle wdrażać innowacja i uważać na błędy, gdyż wówczas pozycja ich narzędzi drastycznie się zmienia.

Tableau ostatnio umożliwił uzyskanie za darmo klucza na 6 miesięcy dla ich aplikacji desktopowej[[27]](#endnote-27). Tableau Desktop to potężne narzędzie umożliwiające bardzo szybkoą wizualizację wyników wyposażony dodatkowo w liczne algorytmy ułatwiającą pracę z danymi. Utrzymuje się on ponadto na pozycji lidera już wiele lat. Do jego głównych konurentów zalicza się SAP BO[[28]](#endnote-28) i QlikView[[29]](#endnote-29).

Z produktów w kategorii enterprise, zdecydowanie prym wiedzie SAP BO BI Platform. Umożliwia sprawne zarządzanie uprawnieniami, administrację serwerami, kontami urzytkowników, import autentykacji SAP, LDAP, AD.

Rysunek 14 SAP BO BI Platform CMC



Jak widać na rysunku powyżej z jednego miejca można zarządzać połaczeniami, warstwą semantyczną (uniwersami), kluczami licencyjnymi, prowadzić monitoring platformy, harmonogramować zadania.

Jest to dojrzały produkt który posiada bardzo dobrą dokumentację jak i support ze strony SAP. Platforma może być instalowana zarówno pod Linuxem jak i Windowsem. W ostatnich latach nawiązała się intensywna współpraca pomiędzy SUSE Linux i SAP, co prowadzi do dalszej optymalizacji działania. Ostatnio SUSE prowadzi intensywną kampanię swojego Linuxowego serwera oferując darmowy okres próbny 60 dni[[30]](#endnote-30).

1. Projektowanie i implementacja programu przygotowującego do egzaminu teoretycznego na brązową odznakę jeździecką.

Pracę nad tworzeniem aplikację wykonano na komputerze Dell Latitude E5440, wyposażonym w procesor Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz, 16 GB pamięci operacyjnej RAM, dysk twardy 500 GB, system operacyjny Windows 7 Enterprise Service Pack 1 64 bit. Zrzuty wykonano przy pomocy programu Microsoft One Note. Do edycji rysunków wykorzystano program Microsoft Paint.

3.1 Projektowanie Aplikacji

Cele strategiczne

Umożliwienie nauki w dowolnym miejscu i dowolnym czasie.

Ułatwienie przygotowania do egzaminu teoretycznego na brązową odznakę jeździecką.

Grupy użytkowników:

Osoby planujące zdawać egzamin na brązową odznakę jeździecką i jeźdźcy planujący poszerzyć swoją wiedzę teoretyczną:

- wybór działu do nauczenia

- wyświetlanie pytań o odpowiedzi

- możliwość wyświetlenia poprzedniego pytania

Lista wymagań funkcjonalnych:

|  |  |
| --- | --- |
| Nazwa: | Wybór działu do nauki |
| ID: WF-001 | |
| Aktorzy: Osoby planujące zdawać egzamin na brązową odznakę jeździecką i jeźdźcy planujący poszerzyć swoją wiedzę teoretyczną | |
| Przypadek użycia: niezbędny | |
| Priorytet: Wysoki | |
| Opis:  Aplikacja ma umożliwić w prosty sposób wybór działu do nauczenia. | |
| Warunki początkowe:  Posiadanie urządzenia z systemem Android | |
| Kryteria akceptacji:  Aktor klika w odpowiedni przycisk. | |
| Scenariusz główny:   * Aktor dokonuje wyboru działu poprzez kliknięcie w przycisk z nazwą tego działu. | |
| Scenariusze alternatywne i rozszerzenia:   * Brak działu na ekranie, wywołanie kolejnego ekranu z resztą tematów. | |

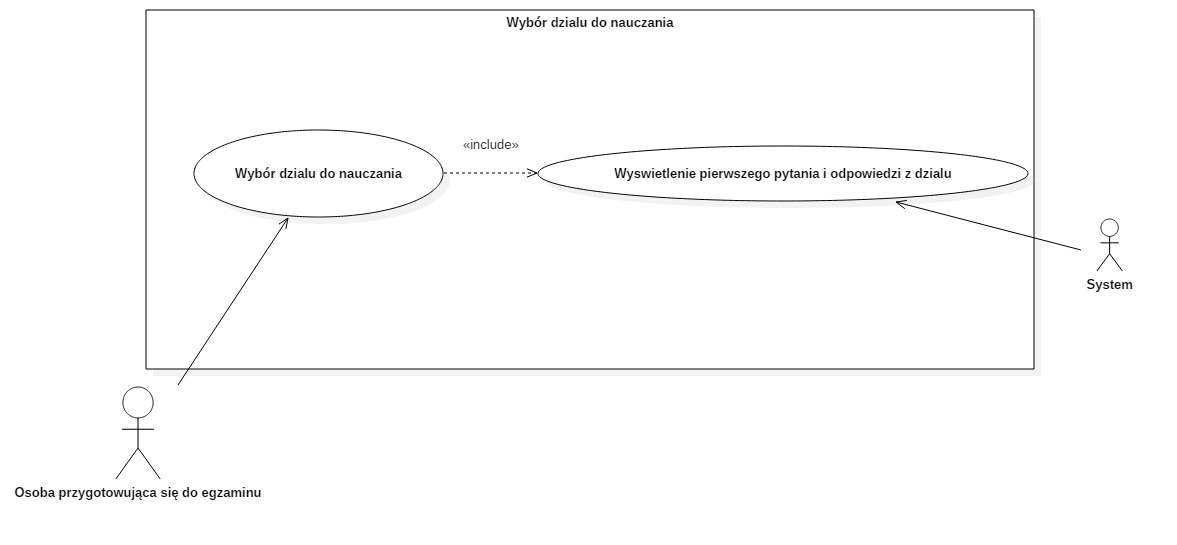
|  |  |
| --- | --- |
| Nazwa: | Wyświetlenie pytań i odpowiedzi |
| ID: WF-002 | |
| Aktorzy: Osoby planujące zdawać egzamin na brązową odznakę jeździecką i jeźdźcy planujący poszerzyć swoją wiedzę teoretyczną | |
| Przypadek użycia: niezbędny | |
| Priorytet: Wysoki | |
| Opis:  Aplikacja ma umożliwić w prosty sposób wyświetlanie pytań i odpowiedzi. | |
| Warunki początkowe:  Wybranie działu do nauczania | |
| Kryteria akceptacji:  Aktor klika w odpowiedni przycisk. | |
| Scenariusz główny:   * Aktor wyświetla kolejne lub poprzednie pytania używając guzików poprzednie i następne. | |
| Scenariusze alternatywne i rozszerzenia:   * Koniec pytań w dziale- wyświetlenie komunikatu o braku większej liczby pytań. * Powrót to listy działów za pomocą przycisku | |

Lissta wymagań niefunkcjonalnych:

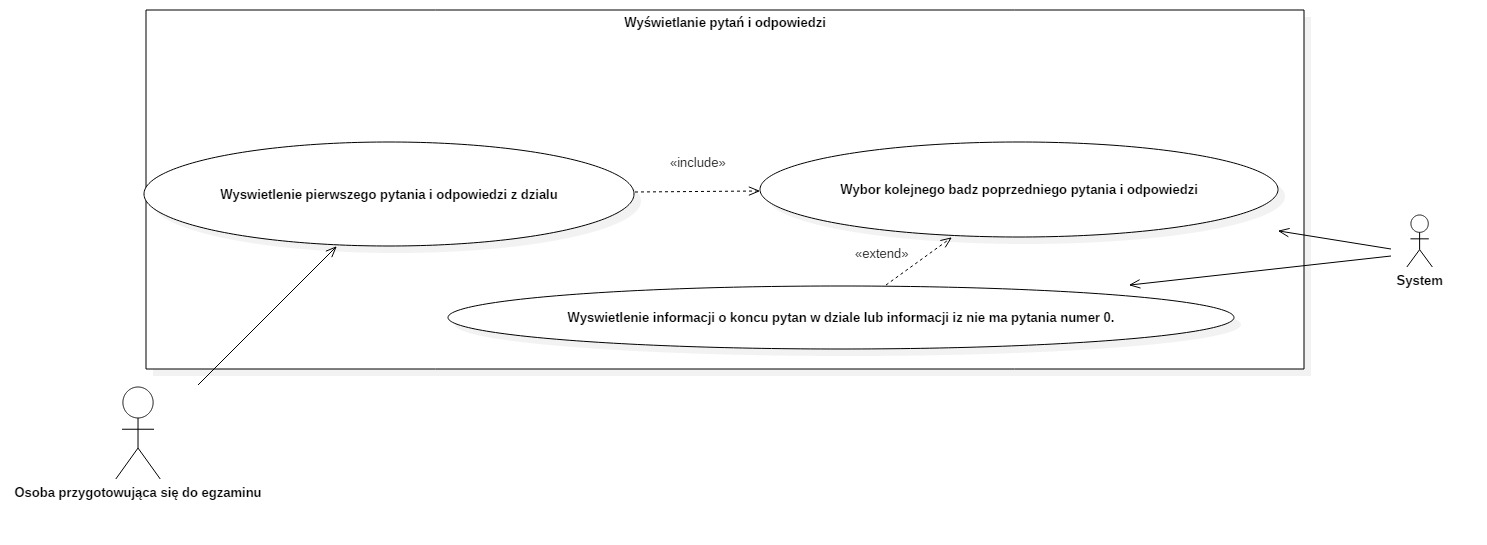
|  |  |
| --- | --- |
| Nazwa: | Skalowalność aplikacji |
| ID: WN-001 | |
| Aktorzy: System | |
| Przypadek użycia: niezbędny | |
| Priorytet: Wysoki | |
| Opis:  System ma mieć rozszerzania o kolejne elementy (nowe działy i odznaki) | |
| Warunki początkowe: | |
| Kryteria akceptacji: | |
| Scenariusz główny: | |
| Scenariusze alternatywne i rozszerzenia: | |

Diagramy przypadków użycia:

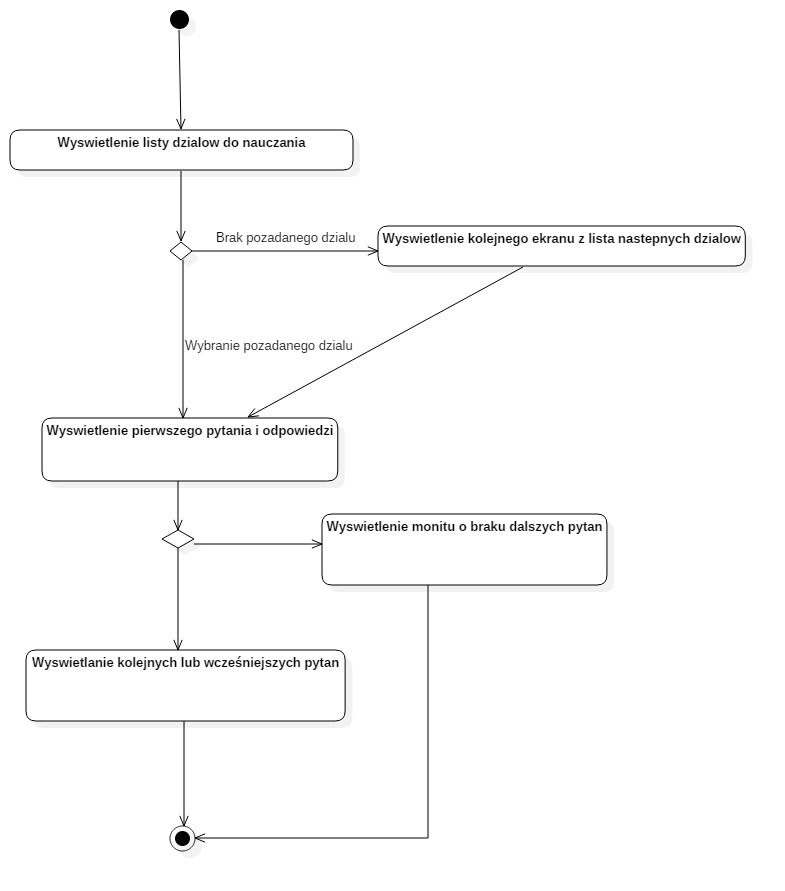
Rysunek 15 Diadram przypadków użycia- wybór działu do nauki



Rysunek 16 Diagram przypadków użycia- wyświetlanie pytań i odpowiedzi



Rysunek 17 Diagram aktywności

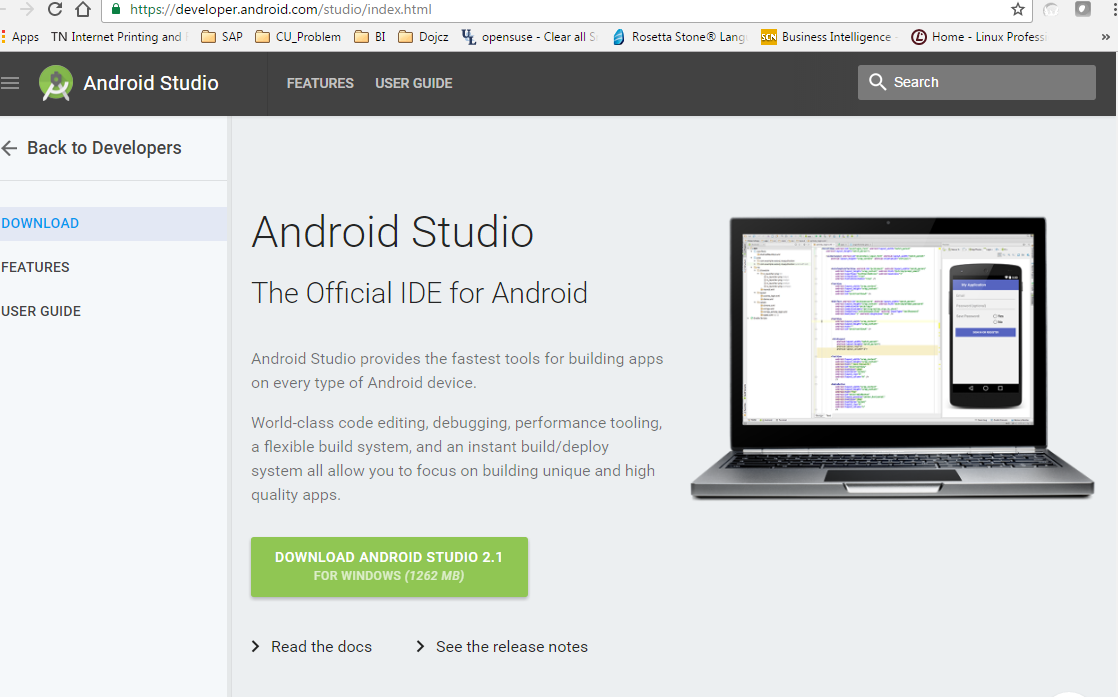


3.xx Pobieranie i instalacja Android Studio IDE, utworzenie pustego projektu.

Ze względu na to, iż do Androida zintegrowane środowisko programistyczne można pobrać i użytkować bezpłatnie, a także, iż wśród rodziny i znajomych jest to najpowszechniejszy system w smartfonach, zdecydowałem się na wybór tej platformy. Umożliwia mi to łatwą instalację aplikacji na starych „zapasowych” telefonach i rozdaniu ich uczestnikom badań.

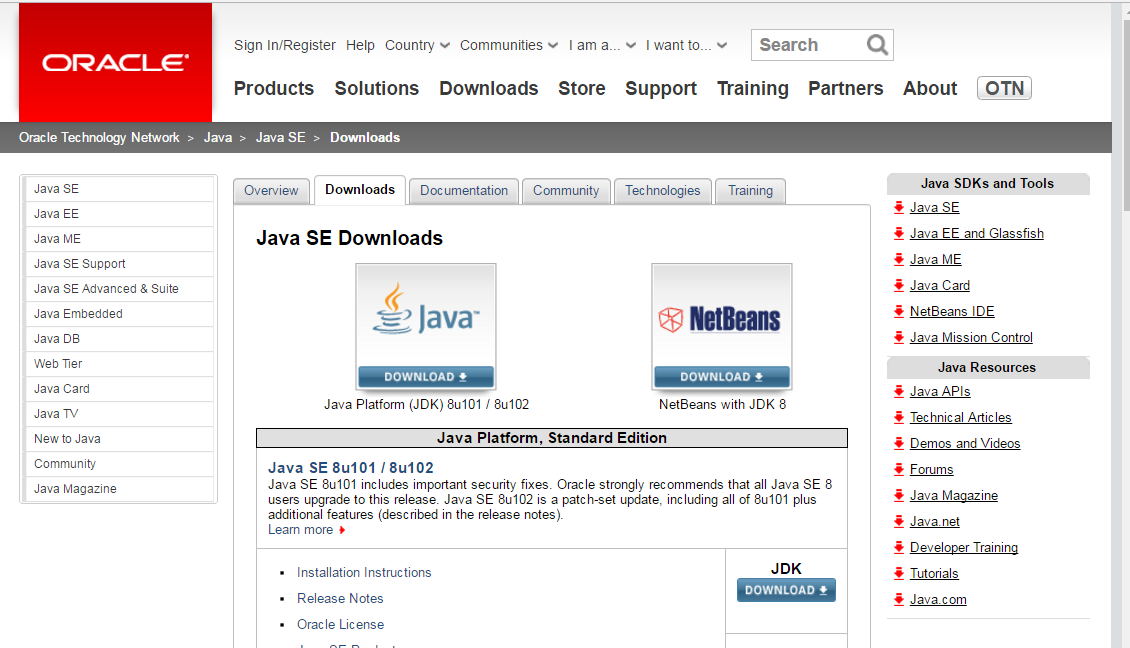
Pracę rozpoczynam od pobrania i instalacji Android Studio[[31]](#endnote-31).

Rysunek 18 Pobieranie Android Studio



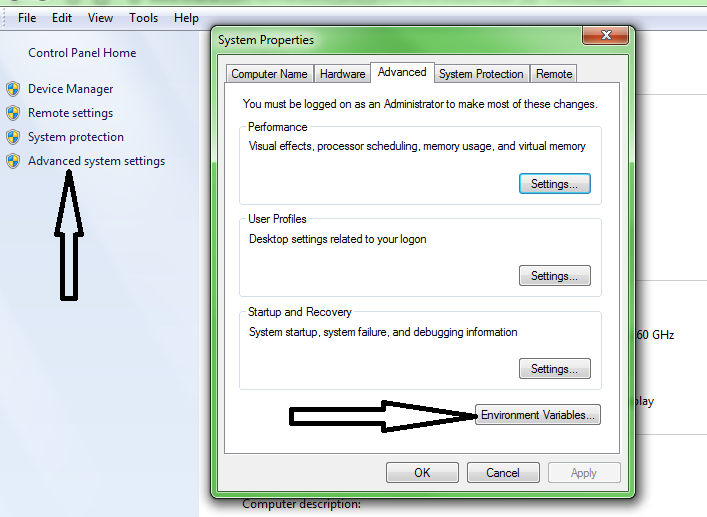
Jednym z prerekwizytów do działania Android Studio jest posiadanie Java Development Kit w wersji minimum 6, którą również pobieram bezpłatnie ze strony Oracle[[32]](#endnote-32). Java Runtime Environment zainstalowany na większości komputerów standardowo nie jest wystarczający do tworzenia programów. Przy pobieraniu należy się upewnić, że ściągamy narzędzia developerskie.

Rysunek 19 Pobieranie Java Development Kit



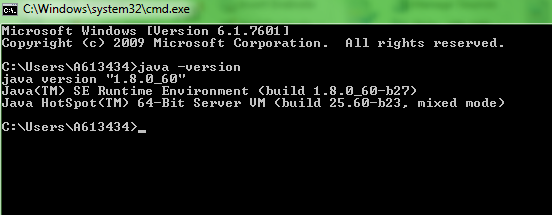
Po instalacji Java SDK należy ustawiam zmienną środowiskową JAVA\_HOME, a także dodaję katalog instalacji Javy do zmiennej PATH. W Windowsie zmienne środowiskowe ustawia się poprzez wejście do panelu sterowania, następnie system i advanced system settings.

Rysunek 20 Ustawianie zmiennnych środowiskowych



Po ustawieniu zmiennych środowiskowych upewniam się, iż zrobiłem to prawidłowo. W tym celu naciskam Super + R, wpisuję cmd i wciskam enter. Następnie w linii komend wpisuję java -version i patrzę czy system „widzi” javę.

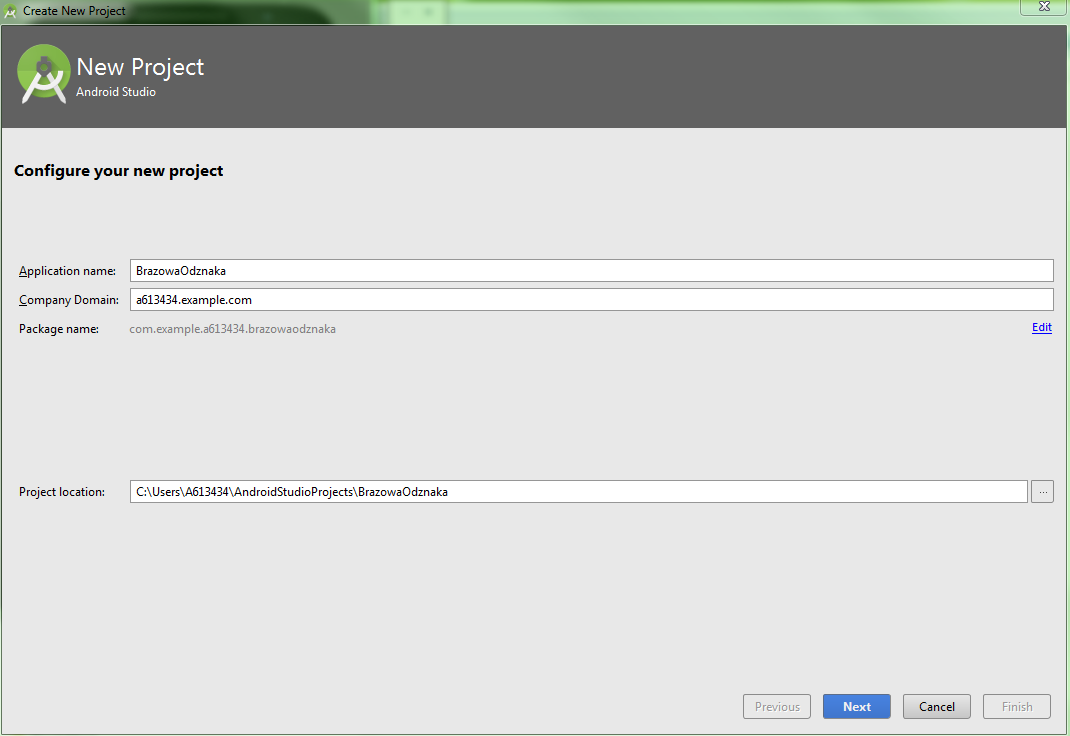
Rysunek 21 Sprawdzanie prawidłowości konfiguracji JAVA



Mając zainstalowaną Javę, mogę już zainstalować i uruchomić Android Studio. Okazuje się, iż jeszcze muszę doinstalować brakujące SDK Androida jak również narzędzia developerskie. Proces ten zabiera dość dużo czasu, gdyż łącznie do pobrania jest ponad 2 GB danych.

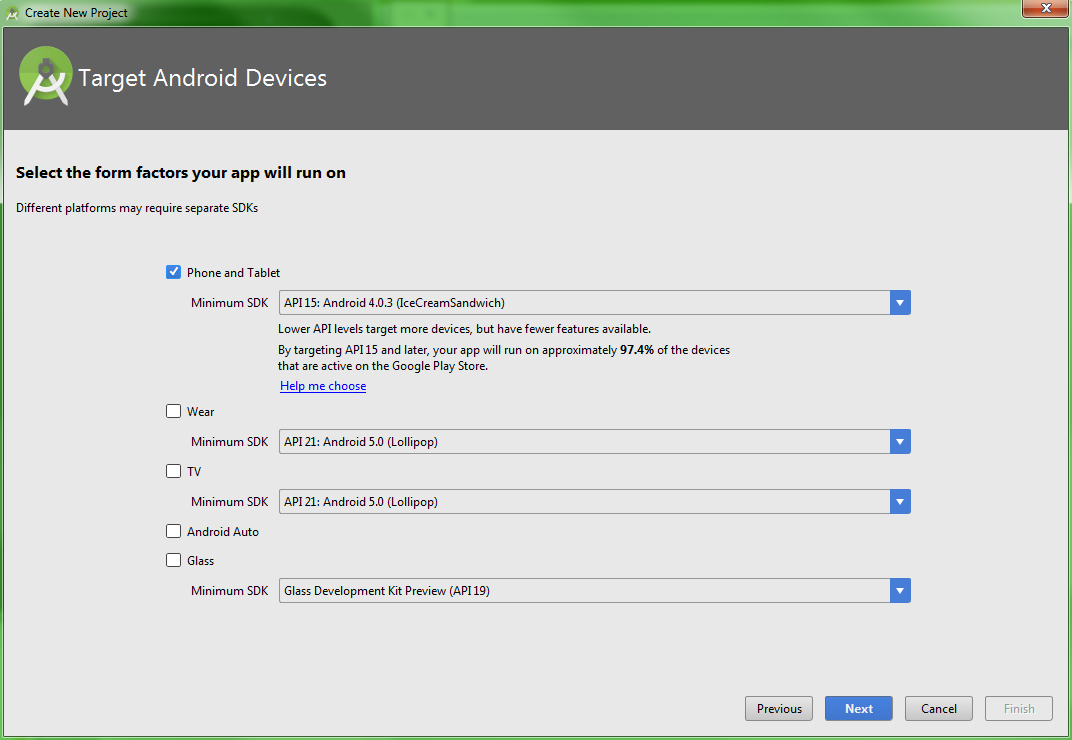
Następnie tworzę szkielet mojej aplikacji. Wybieram File -> New ->New Project. Pojawia się okno dialogowe, gdzie wpisuję nazwę aplikacji:

Rysunek 22 Tworzenie nowego projektu w Android Studio



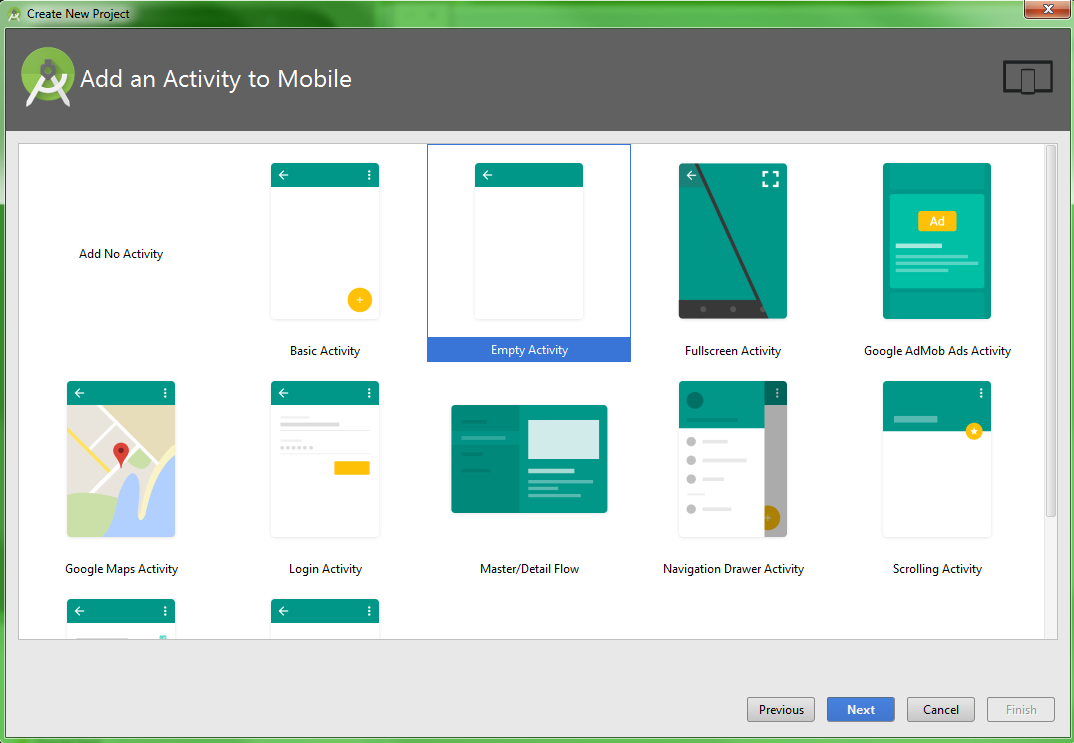
Po kliknięciu na przycisk “Next” pojawia się następne okno dialogowe. Jest ono bardzo ważne, ponieważ wybieram na nim minimalny poziom SDK, na którym będzie pracować moja aplikacja. Sprawdziłem, iż najstarszy telefon ma poziom API 16, więc na wszelki wypadek schodzę jeszcze poziom niżej. Od razu wyświetla się informacja, iż aplikacja będzie działać na 97,4% urządzeń z Androidem.

Rysunek 23 Wybór minimalnego kompatybilnego SDK



Na kolejnym ekranie pojawia się wybór aktywności jaką można wygenerować. Jako, iż na razie jestem zainteresowany utworzeniem tylko szkieletu aplikacji, wybieram Empty Activity, gdyż później sam zdecyduję co, gdzie ma się znajdować.

Rysunek 24 Wybranie jaką aktywność ma być utworzona na początku.



Następnie pojawia się okno dialogowe, gdzie można nadać nazwę aktywności, po czym zostaje utworzona gotowa aplikacja wyświetlająca na ekranie „Hallo World!”.

3.xxxx Utworzenie bazy danych SQLite

Egzamin teoretyczny na brązową odznakę obejmuje zakres 345pytań. Aby go zdać trzeba uzyskać 50% poprawnych odpowiedzi. Taka ilość pytań bez podziału na kategorie jest ciężka do nauczenia. Zdecydowałem podzielić się materiał na 14 kategorii:

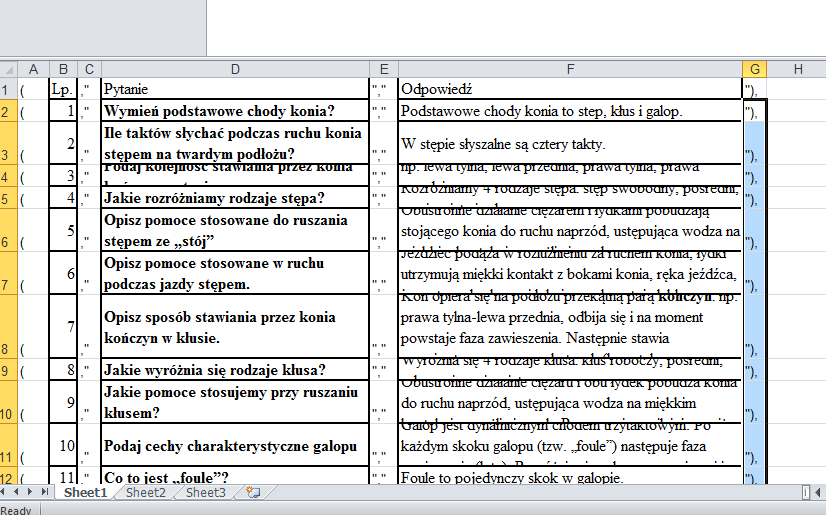
1. Koń z natury
2. Stajnia
3. Podstawowe zasady bezpieczeństwa przy pracy z koniem, wypadki i pierwsza pomoc
4. Pielęgnacja i wyprowadzanie koni.
5. Cechy użytkowe konia:
6. Żywienie koni
7. Zdrowie koni
8. Sprzęt jeździecki
9. Kiełznanie i siodłanie konia
10. Wyprowadzanie osiodłanego konia, wsiadanie i zsiadanie z konia.
11. Podstawowe zasady zachowania na ujeżdżalni
12. Podstawy jazdy konnej.
13. Teoria klasycznej jazdy konnej
14. Chody konia, skoki przez przeszkody, jazda w terenie.

Podziału dokonałem o własną wiedzę i doświadczenie- jestem instruktorem jazdy konnej od 2008 roku, jeżdżę konno od roku 1995. Przez kilkanaście lat aktywnie startowałem w zawodach z dyscypliny skoków przez przeszkody. Następnie utworzyłem tabele w Wordzie i przekopiowałem tam pytania i odpowiedzi z PDF udostępnianego przez PZJ.

To dało mi podstawę do utworzenia bazy SQLite, a jednocześnie dało materiały drukowane, które wykorzystałem później w badaniach.

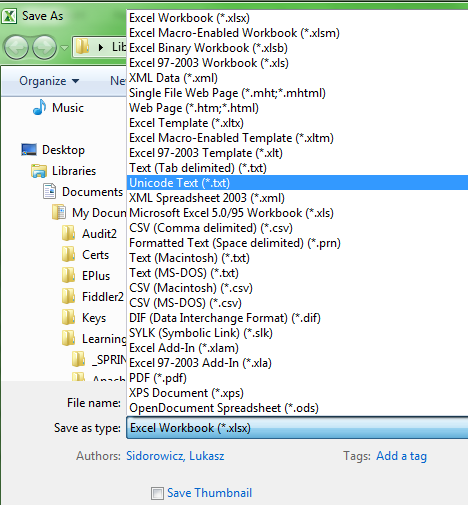
Każdą tabelę w sposób pokazany na ilustracji poniżej przeniosłem do programu Microsoft Excel 2010:

Rysunek 25 Wstawianie danych do Ms Excel 2010



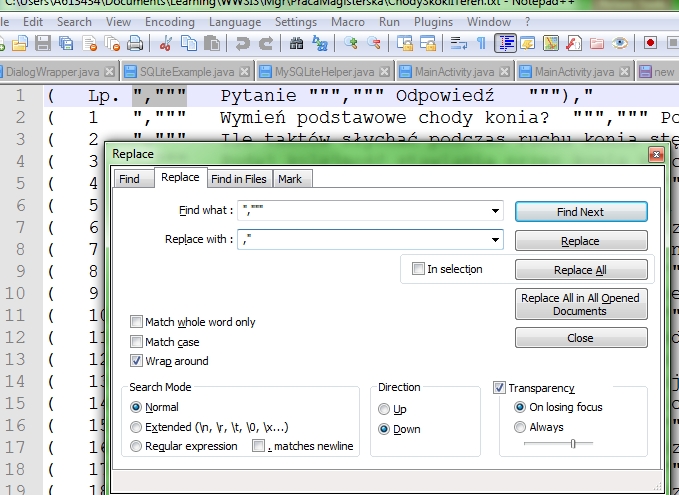
Następnie z poziomu MS Excel zapisywałem plik do formatu Unicode. Umożliwia to poradzenie sobie z polskimi znakami. Próba przekonwertowania do formatu CSV powodowała wyświetlanie krzaczków zamiast polskich liter.

Rysunek 26 Przekonwertowanie arkusza na plik w formacie Unicode



Otrzymany plik w formacie Unicode poprawiałem w programie Notepad++[[33]](#endnote-33) do składni SQLowej. Znając tą składnię, wiedziałem, iż każdy wiersz z excela trzeba przerobić do formatu: „(1, „Pytanie”, „Odpowiedz),”.

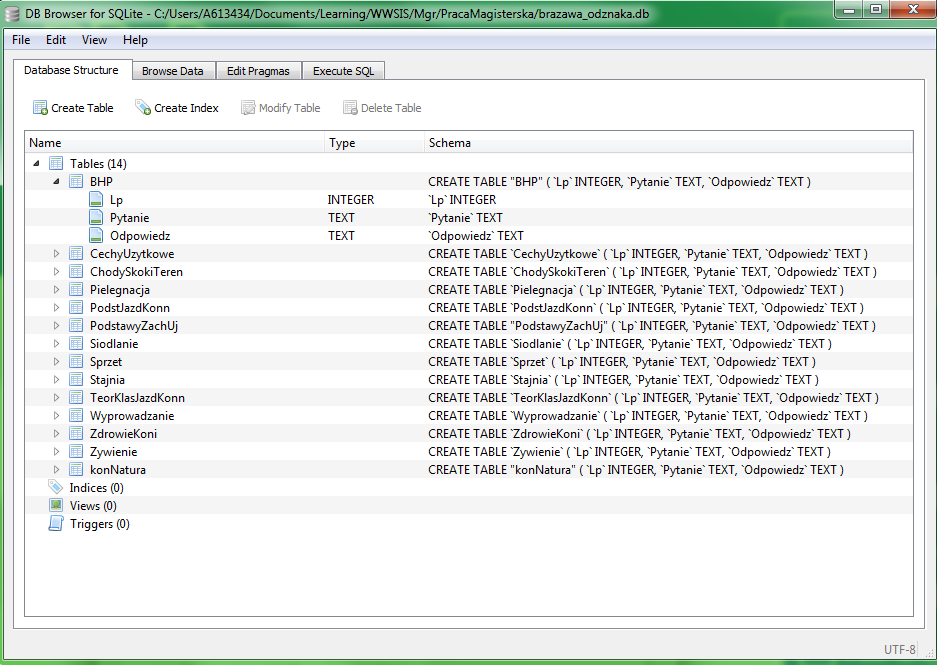
Co umożliwia później szybkie utworzenie kolejnych tabel w bazie.

Rysunek 27 Podmiana znaków w programie Notepad++

Do utworzenia bazy wykorzystałem SQLite3[[34]](#endnote-34). SQLite pomimo pewnych ograniczeń bardzo popularnym, wykorzystywanych przez takich gigantów jak Alphabet, Apple, Facebook[[35]](#endnote-35). Pobranie i korzystanie z bazy jest bezpłatne. Silnik bazy SQLite jest wbudowany w system Android, można od razu korzystać z bazy bez konieczności instalowania czegokolwiek i problemów z kompatybilnością[[36]](#endnote-36). Aby nie tworzyć bazy w niewygodnej linii komend, użyłem programu DB Browser for SQLite. Umożliwia on w prosty sposób utworzenie bazy, dodanie tabel i wykonywanie kodu SQL[[37]](#endnote-37).

Z pomocą tego programu utworzyłem bazę danych, utworzyłem czternaście tabel i wstawiłem do nich odpowiednie dane przygotowane wcześniej. Mając tak przygotowaną bazę mogłem wrócić do tworzenia aplikacji w Android Studio.

Rysunek 28 Tworzenie bazy danych w DB Browser for SQLite



3.xxx Wykonanie aplikacji mobilnej

Dysponując szkieletem aplikacji stworzonym wcześniej i bazą danych SQLite zawierającą czternaście tabel, mogłem rozpocząć implementację aplikacji.

Pierwszą rzeczą było opracowanie ekranu startowego aby użytkownik po uruchomieniu programu widział poszczególne działy do nauczania.

Wszystkie zmiany dokonywane w projekcie dodaję do repozytorium lokalnego Git.

Na początku inicjuję śledzenie wersji poprzez otwarcie wiersza poleceń (CMD). Przechodzę do folderu z projektem wydaję komendę:

git init

Powoduje to powstanie pustego repozytorium. Następnie trzeba dodaję wszystkie pliki i foldery do tzw. staging area poprzez komendę:

git add \*

Aby pliki i foldery zostały dodane do repozytorium wykonuję polecenie

git commit –m „Wersja początkowa”

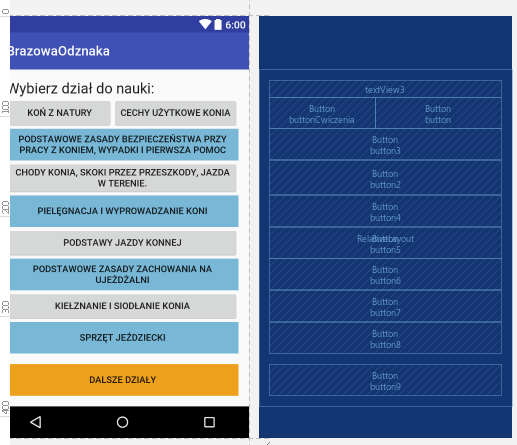
parametr „-m” powoduje możliwość dodania komentarza. W przypadku braku tego parametru otworzy się edytor tekstowy i komentarz trzeba dodać z jego poziomu. Ja preferuję wersję konsolową. Jest to indywidualna preferencja. Po każdych zmianach powtarzam komendy git add i git commit –m, żeby w razie napisania złego nowego fragmentu kodu móc wrócić do wcześniejszej działającej wersji.

Tak zwany laout projektuję w xml, podaję poniżej fragment z przykładem ustawienia etykiety (TextView) i jednego przycisku (Button):

<**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Koń z natury"  
 android:onClick="info"  
 android:id="@+id/buttonCwiczenia"  
 android:layout\_below="@+id/textView3"  
 android:layout\_toLeftOf="@+id/button"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"** />  
  
<**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textAppearance="?android:attr/textAppearanceLarge"  
 android:text="Wybierz dział do nauki:"  
 android:id="@+id/textView3"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentEnd="true"** />

następnie korzystam z możliwości podglądu.

Rysunek 29 Tworzenie layoutu menu głównego aplikacji



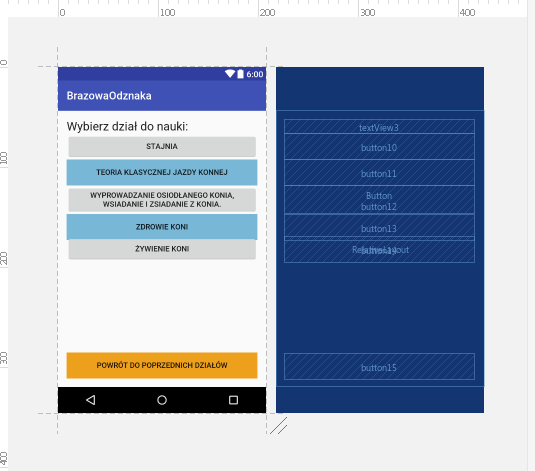
Dla moich przycisków chcę używać własnych zdefiniowanych kolorów. Dodaję nowe wartości w colors.xml:

*<?***xml version="1.0" encoding="utf-8"***?>*<**resources**>  
 <**color name="colorPrimary"**>#3F51B5</**color**>  
 <**color name="colorPrimaryDark"**>#303F9F</**color**>  
 <**color name="colorAccent"**>#FF4081</**color**>  
 <**color name="Czerwony"**>#eda01b</**color**>  
 <**color name="Czarny"**>#ff0a0a0c</**color**>  
 <**color name="Brazowy"**>#ffaf3c00</**color**>  
 <**color name="blue"**>#78b8d6</**color**>  
 <**color name="Zielony"**>#ff20ff00</**color**>  
 <**color name="Morelowy"**>#ffffed36</**color**>

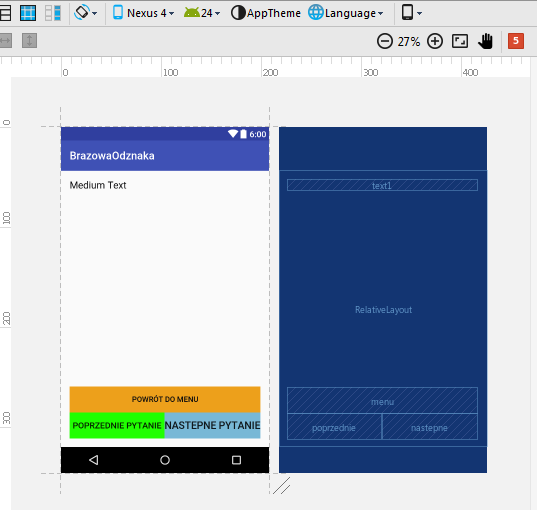
Dodaję również nazwę aplikacji do pliku strings.xml:

<**resources**>  
 <**string name="app\_name"**>BrazowaOdznaka</**string**>  
 <**string name="action\_settings"**>Settings</**string**>  
</**resources**>

Tematy nie mieszczące się na ekranie przesuwam na kolejny:

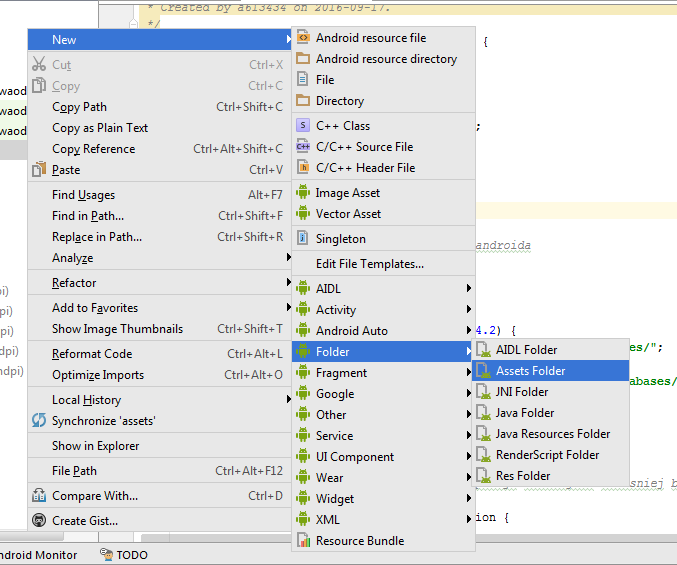
Rysunek 30 Tworzenie layoutu listy pozostałych działów

Projektuję również ekran do wyświetlania pytań i odpowiedzi:

Rysunek 31 Projektowanie layoutu do wyświetlania pytań i odpowiedzi.

Tym sposobem kończę wizualne przygotowanie aplikacji. Pozostaje oprogramować poszczególne funkcje, tak, aby spełnić założenia projektowe. Ponieważ aplikacja będzie wykorzystywać utworzoną bazę SQLite tworzę w z poziomu środowiska Android Studio folder „Assets”. Konieczne jest wykonanie tego kroku z poziomu platformy, utworzenie folderu z poziomu systemu operacyjnego powoduje, iż później nasza aplikacja nie widzi bazy.

Rysunek 32 Dodanie folderu Assets do projektu



Do utworzonego w ten sposób folderu kopiuję utworzoną wcześniej bazę brazawa\_odznaka.db.

Dzięki temu, że SDQLite jest wbudowany w Androida, mogę dostać się do bazy bez wykorzystywania jakichkolwiek sterowników takich jak JDBC.

Dodaję w pliku AndroidManifest.xml wymaganie, aby aplikacja miała prawo zapisu w pamięci zewnętrznej:

<**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />

Pracę z bazą rozpoczynam od utworzenia klasy BazaHelper rozszerzającej SQLiteOpenHelper[[38]](#endnote-38). Na początku definiuję zmienne ze ścieżką do bazy, nazwą bazy i tabel w niej zawartych:

**private static** String *DB\_PATH* = **""**;  
**private static** String *DB\_NAME* = **"brazawa\_odznaka2.db"**;  
**private static** String *BHP* = **"BHP"**;  
**private static** String *konNatura* = **"konNatura"**;  
**private static** String *cachyUzytkowe* = **"CechyUzytkowe"**;  
**private static** String *chodySkokiTeren*= **"ChodySkokiTeren"**;  
**private static** String *pielegnacja*= **"Pielegnacja"**;  
**private static** String *podstJazdKonnej*= **"PodstJazdKonn"**;  
**private static** String *podstZachUj*=**"PodstawyZachUj"**;  
**private static** String *siodlanie* = **"Siodlanie"**;  
**private static** String *sprzet* = **"Sprzet"**;  
**private static** String *stajnia* = **"Stajnia"**;  
**private static** String *teoriaKlasJazdKonn* = **"TeorKlasJazdKonn"**;  
**private static** String *wyprowadzanie* = **"Wyprowadzanie"**;  
**private static** String *zdrowieKoni* = **"ZdrowieKoni"**;  
**private static** String *zywienie*= **"Zywienie"**;  
**private** SQLiteDatabase **myDataBase**;  
**private final** Context **myContext**;

Ponieważ system operacyjny Android zmienia się dosyć mocno z kolejną wersją, mmogą występować problemy ze ścieżką do bazy. Aby rozwiązać ten problem tworzę kontruktor, który dostosowuje ścieżkę do różnych wersji Androida[[39]](#endnote-39):

**public** BazaHelper(Context context) {  
 **super**(context, *DB\_NAME*, **null**, 1);  
 **if** (android.os.Build.VERSION.***SDK\_INT*** >= 4.2) {  
 DB\_PATH = context.getApplicationInfo().**dataDir** + **"/databases/"**;  
 } **else** {  
 DB\_PATH = **"/data/data/"** + context.getPackageName() + **"/databases/"**;  
 }  
 **this**.**myContext** = context;  
}

Aby utworzona przeze mnie baza danych mogła być zainstalowana w aplikacji wykorzystuję metodę createDataBase(), która najpierw sprawdza czy jakakolwiek baza jest już utworzona, gry nie tworzy pustą bazę, a następnie kopiuje bazę z folderu Assets:

**public void** createDataBase() **throws** IOException {

**boolean** dbExist = checkDataBase();  
  
 **if** (dbExist) {  
} **else** {  
 **this**.getWritableDatabase();  
**try** {  
 copyDataBase();  
 } **catch** (IOException e) {  
 **throw new** Error(**"Blad kopiowania bazy"**);  
 }  
 }  
  
}

W celu sprawdzenia, czy baza istnieje implementuję metodę checkDataBase():

**private boolean** checkDataBase() {  
  
 SQLiteDatabase checkDB = **null**;  
 **try** {  
 String myPath = *DB\_PATH* + *DB\_NAME*;  
 checkDB = SQLiteDatabase.*openDatabase*(myPath, **null**, SQLiteDatabase.***OPEN\_READONLY***);  
 } **catch** (SQLiteException e) {}  
 **if** (checkDB != **null**) {  
 checkDB.close();  
 }  
  
 **return** checkDB != **null** ? **true** : **false**;  
}

Kopiowanie bazy odpywa się z wykorzystaniem metody copyDataBase(), która otwiera nowoutworzoną pustą bazę, a następnie kopiuje do niej gotową bazę za pomocą strumienia bajtów:

**private void** copyDataBase() **throws** IOException {  
InputStream myInput = **myContext**.getAssets().open(*DB\_NAME*);  
String outFileName = *DB\_PATH* + *DB\_NAME*;  
  
OutputStream myOutput = **new** FileOutputStream(outFileName);  
**byte**[] buffer = **new byte**[1024];  
 **int** length;  
 **while** ((length = myInput.read(buffer)) > 0) {  
 myOutput.write(buffer, 0, length);  
 }  
myOutput.flush();  
 myOutput.close();  
 myInput.close();  
  
}

Ponieważ rozszerzałem klasę abstrakcyjną, przesłaniam metody onCreate(), closer() i onUpgrade():

@Override  
**public void** onCreate(SQLiteDatabase db) {  
  
}  
  
@Override  
**public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
  
}

@Override  
**public synchronized void** close() {  
 **if** (myDataBase != **null**)  
 myDataBase.close();  
 **super**.close();  
  
}

Jako ostatnią rzecz w klasie BazaHelper tworzę metodę selectWords(), która wykorzystując klasę Cursor odczyta dane z bazy i wstawi do ArrayListy.

**public** List<String> selectWords() {  
 List<String> list = **new** ArrayList<String>();  
 Cursor cursor = **this**.**myDataBase**.query(*BHP*, **new** String[] { **"Lp"**, **"Pytanie"**, **"Odpowiedz"** }, **null**, **null**, **null**, **null**, **null**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 list.add(cursor.getString(0)+**". \n\nPytanie:\n"** + cursor.getString(1) + **"\n\nOdpowiedż:\n"** + cursor.getString(2));  
 } **while** (cursor.moveToNext());  
 }  
 **if** (cursor != **null** && !cursor.isClosed()) {  
 cursor.close();  
 }  
 **return** list;  
}

Metodę tą później kopiuję dla każdej tabeli i tym sposobem mam odczytaną całą bazę danych. Pozostaje opracować wyświetlenie i przewijanie elementów ArrayListy.

Tworzę nową aktywność WyswietlaniePytan. Przy otwarciu planuję wyświetlić pytanie i odpowiedź numer jeden, będę w tym celu pobierał element ArrayListy numer 0, aby tu umożliwić tworzę następujące zmienne:

Integer **myInt**=0;  
String **Pytanie**;  
BazaHelper **myDbHelper**;  
Button **poprzed**;  
Button **nast**;  
TextView **text**;

Ustawiam następnie widok i inicjuję object z klasy BazaHelper, przyporządkowuję także pole tekstowe i przyciski do odpowiednich zmiennych:

setContentView(R.layout.***activity\_wyswietlanie\_pytan***);  
**myDbHelper** = **new** BazaHelper(**this**);  
**text** = (TextView) findViewById(R.id.***text1***);  
**poprzed** = (Button) findViewById(R.id.***poprzednie***);  
**nast** = (Button) findViewById(R.id.***nastepne***);

Po tym inicjuję bazę danych i ją otwieram, abym mógł pobrać i wyświetlić pierwsze pytanie i odpowiedź. Tworzę takżę listę ciągów znaków i ustawiam pierwszy element w polu tekstowym:

**try** {  
 **myDbHelper**.createDataBase();  
} **catch** (IOException ioe) {  
 **throw new** Error(**"Niepowodzenie tworzenia bazy"**);  
}  
  
**try** {  
 **myDbHelper**.openDataBase();  
} **catch** (SQLException sqle) {  
 **throw** sqle;  
}  
  
**final** List<String> names = **myDbHelper**.selectWords();  
**Pytanie** = names.get(**myInt**);  
**text**.setText(**Pytanie**);

Oprogramowuję pierwszy przycisk, mający służyć wyświetlaniu kolejnych pytań. Zabezpieczam program także, poza wyjściem za ostatni element listy poprzez wyświetlenie komunikatu o braku większej ilości pytań w danym dziale:

**nast**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **myInt**++;  
 Integer safe = names.size()-1;  
 **if** (**myInt**<=safe && **myInt** >=0){  
 **Pytanie** = names.get(**myInt**);  
 **text**.setText(**Pytanie**);}  
  
 **else** {  
 Toast.*makeText*(getApplicationContext(),**"Nie ma już więcej pytań w tym dziale."**, Toast.***LENGTH\_LONG***).show();  
 }  
 }  
});

Podobnie oprogramowują przycisk do przewijania listy do poprzednich pytań, zabezpieczając przed zejściem poniżej pierwszego elementu listy:

**poprzed**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **myInt**--;  
 Integer safe = names.size()-1;  
 **if** (**myInt**<=safe && **myInt** >=0){  
 **Pytanie** = names.get(**myInt**);  
 **text**.setText(**Pytanie**);}  
 **else** {  
 Toast.*makeText*(getApplicationContext(),**"Nie ma pytania o numerze 0 i niżej."**, Toast.***LENGTH\_LONG***).show();  
 }  
 }  
 });  
}

Oprogramowuję także przycisk pozwalający na powrót do menu głównego, za pomocą wywołania Intenta, nazwa metody została zdefiniowana wcześniej w xml:

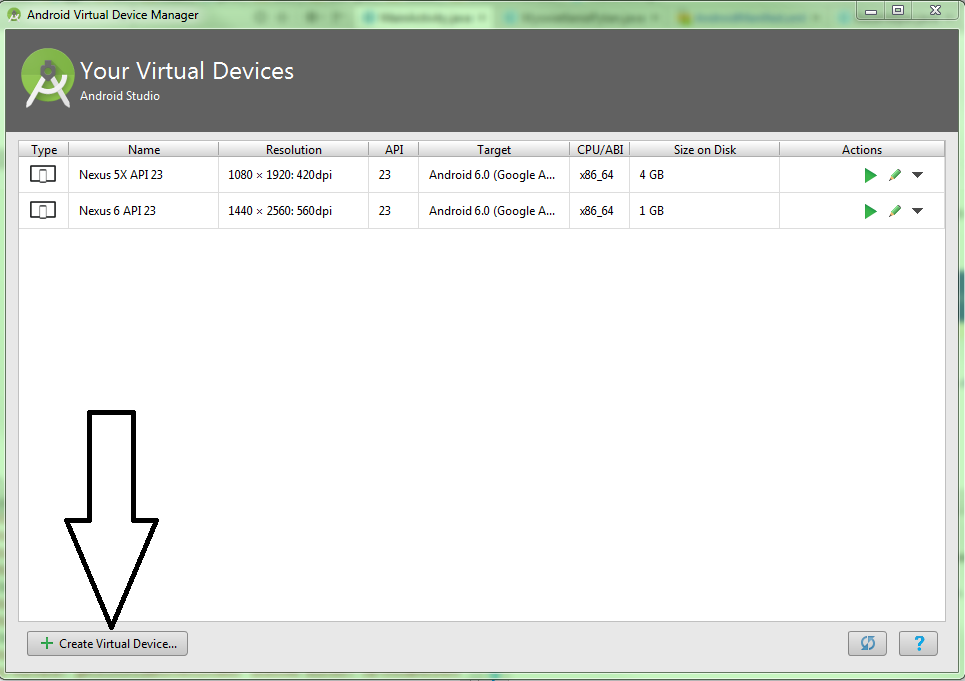
**public void** menu (View view) {  
 Intent menu = **new** Intent (getApplicationContext(), MainActivity.**class**);  
 startActivity(menu);  
}

Aby móc przetestować działanie aplikacji, czy poprawnie pobiera i wyświetla dane, muszę jeszcze oprogramować przynajmniej jeden przycisk w klasie głównej wyświetlającej działy do nauki. Ponieważ w wyświetlaniu pytań ustawiłem tabelkę „BHP”, toteż oprogramowuję na razie tylko przycisk „Podstawowe zasady bezpieczeństwa przy pracy z koniem, wypadki i pierwsza pomoc”. Robię to poprzez wywołanie Intenta do zdefiniowanej wcześniej metody onClick() z widoku xml:

**public void** BHP (View view){  
 Intent BHP = **new** Intent(getApplicationContext(), WyswietlaniePytan.**class**);  
 startActivity(BHP);  
}

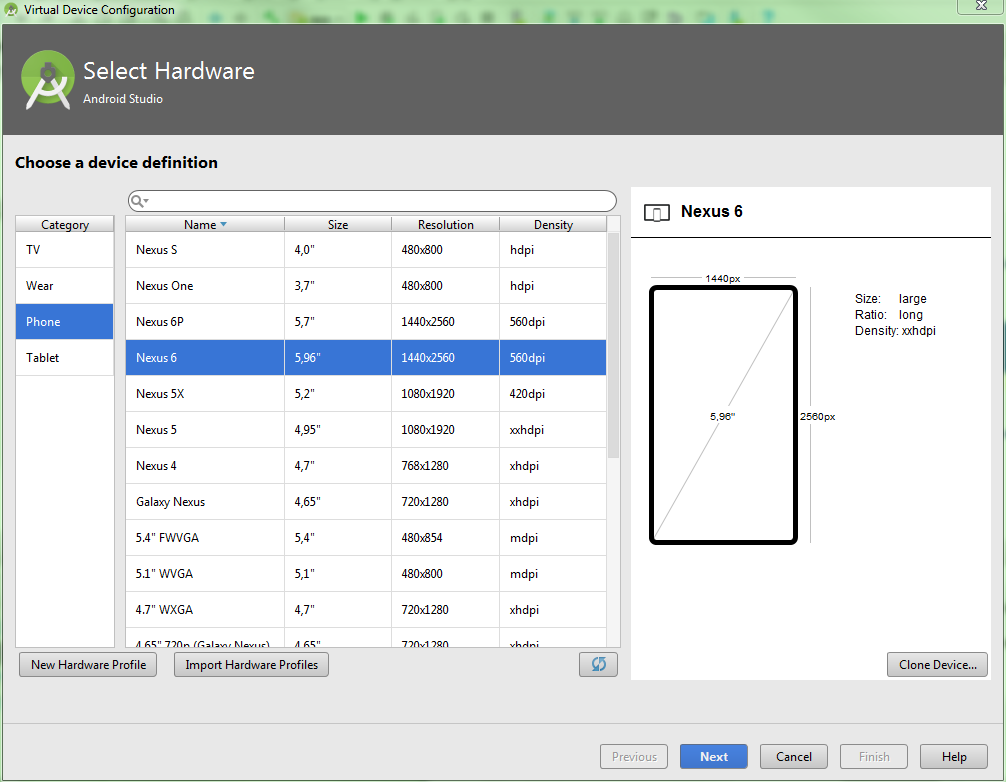
Dysponuję w tym momencie aplikacją, która powinna się uruchomić i wyświetlać pytania. Do testów najlepiej wykorzystać wirtualne urządzenie Android. Zainstalowane środowisko pozwala na utworzenie i używanie takiego urządzenia. Zaletą takiej metody testowania jest brak ingerencji w rzeczywistym urządzeniu, jak również możliwość przetestowania na różnych modelach telefonów i tabletów.Uruchamiam Android Virtual Device Manager i przyciskam Create Virtual Device:

Rysunek 33 Tworzenie wirtualnego urządzenia Android



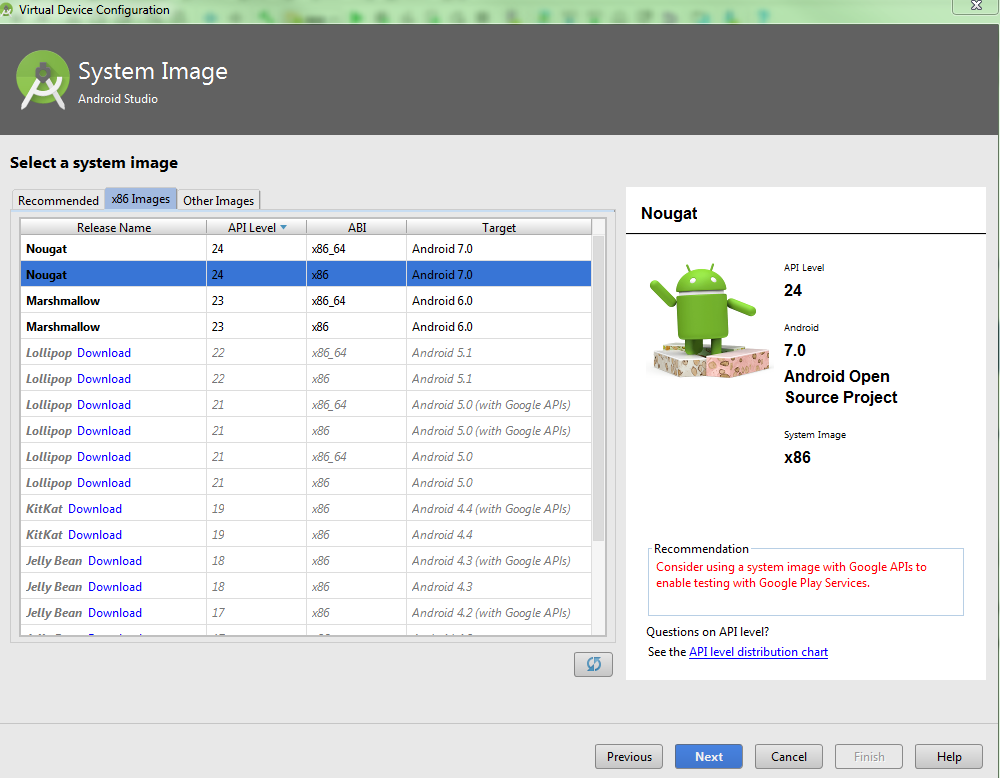
Następnie otrzymuję listę urządzeń, dostępne obecnie kategorie obejmują Phone, Tablet, Wear (Smartwatche) i nawet TV. Wybieram kategorię telefonu i model Nexus6.

Rysunek 34 Wybór tworzonego wirtualnego urządzenia



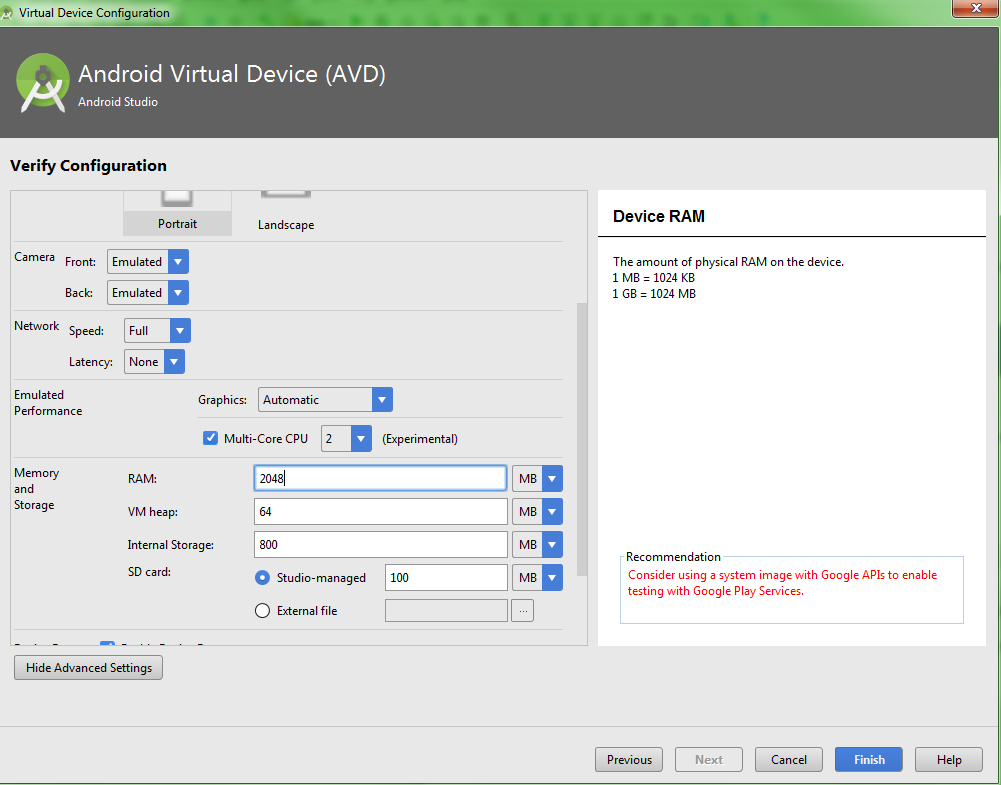
Na kolejnym ekranie wybieram, jaka wersja Androida ma być zainstalowana na urządzeniu wirtualnym. Jest to bardzo istotne w przypadku testów. Dzięki tej funkcji można przejrzeć, czy kompatybilność ze starszymi urządzeniami jest zaimplementowana prawidłowo. Jeśli jakiś poziom API jest niedostępny, można go or razu pobrać.

Rysunek 35 Wybór poziomu API w tworzonym urządzeniu



Na kolejnym i zarazem ostatnim ekranie mamy możliwość przejrzeć i zmienić wybrane wcześniej ustawienia, jak również w opcje zaawansowane. Polecam tę opcję i zwiększenie ilości RAM. Pozwoli to na szybsze uruchamianie aplikacji i jej testowanie. Pozwala to zaoszczędzić dużo czasu w trakcie późniejszej pracy. Należy tylko pamiętać, aby dostosować ten parametr do posiadanych zasobów na komputerze.

Rysunek 36 Wybór ustawień zaawansowanych dla tworzonej maszyny wirtualnej.



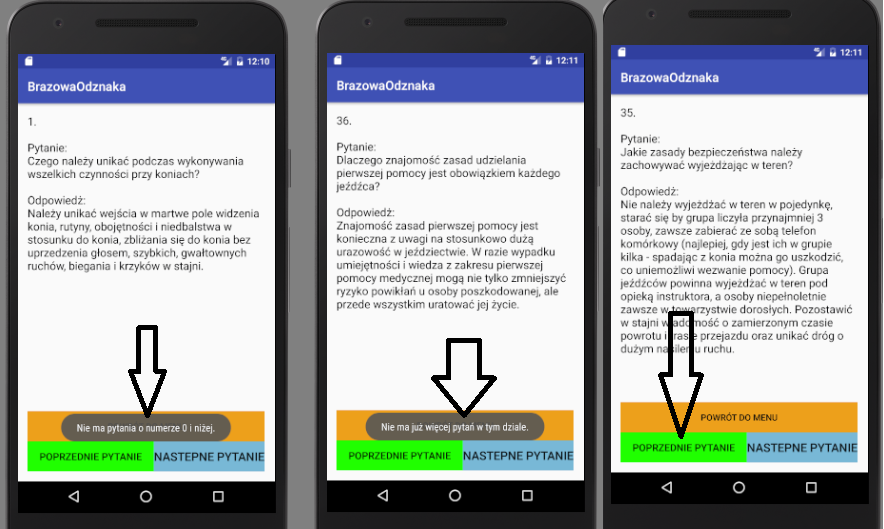
Uruchamiam nowoutworzone wirtualne urządzenie i sprawdzam działanie aplikacji. W szczególności interesuje mnie, czy prawidłowo się otwiera i po naciśnięciu w przycisk „Podstawowe zasady bezpieczeństwa przy pracy z koniem, wypadki i pierwsza pomoc” otworzy nowy ekran i wyświetli pierwsze pytanie i odpowiedź z tego działo. Będzie to oznaczało, iż prawidłowo została zaimplementowana klasa BazaHelper, jak również opcja pobrania pierwszego elementu z listy.

Rysunek 37 Otwarcie aplikacji i wyświetlenie pierwszego pytania z wybranego działu



Kolejnym elementem pozostałym do sprawdzenia pozostaje, czy działa przewijanie pytań i zabezpieczenie przed wyjściem poza tablicę działa i wyświetla komunikat. W tym celu klikam najpierw przycisk „Poprzednie pytanie” i patrzę, czy wyświetla się komunikat. Następnie przewijam kolejne pytania przyciskiem „Następne pytanie” do końca listy i próbuję dalej, następnie naciskam na przycisk „Poprzednie pytanie” Okazuje się, że wszystko działa prawidłowo:

Rysunek 38 Sprawdzanie działania aplikacji.

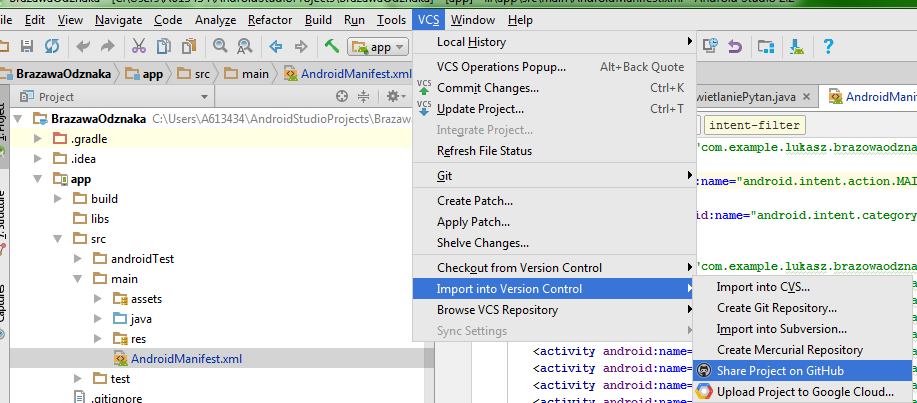


Dzięki sprawdzeniu, iż funkcje wyświetlania działają prawidłowo, przechodzę do implementacji kolejnych przycisków. Dla każdego tworzę nową aktywność rejestrując ją w AndroidManifest.xml:

<**activity android:name="com.example.lukasz.brazowaodznaka.WyswietlaniePytan"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.CechyUzytkowe"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Skoki"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Pielegnacja"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.PodstawyJazdyKonne"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.PodstawyZachowaniaNaUjezdzalni"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Siodlanie"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Sprzet"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Main2Activity"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Stajnia"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.TeoriaKlasJazdKonnej"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Wyprowadzanie"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.ZdrowieKoni"** />  
<**activity android:name="com.example.lukasz.brazowaodznaka.Zywienie"**>

Następnie kopiuję zawartość klasy WyswietlaniePytan.java, podmieniając treść do wyświetlenia. Tym sposobem dysponuję gotową aplikacją. Dla bezpieczeństwa umieszczam ją na platformie GitHub jako repozytorium otwarte. Korzystam, z tego iż w Android Studio jest wbudowana funkcja automatycznego podzielenia się projektem na tej platformie:

Rysunek 39 Umieszczenie projektu na GitHub

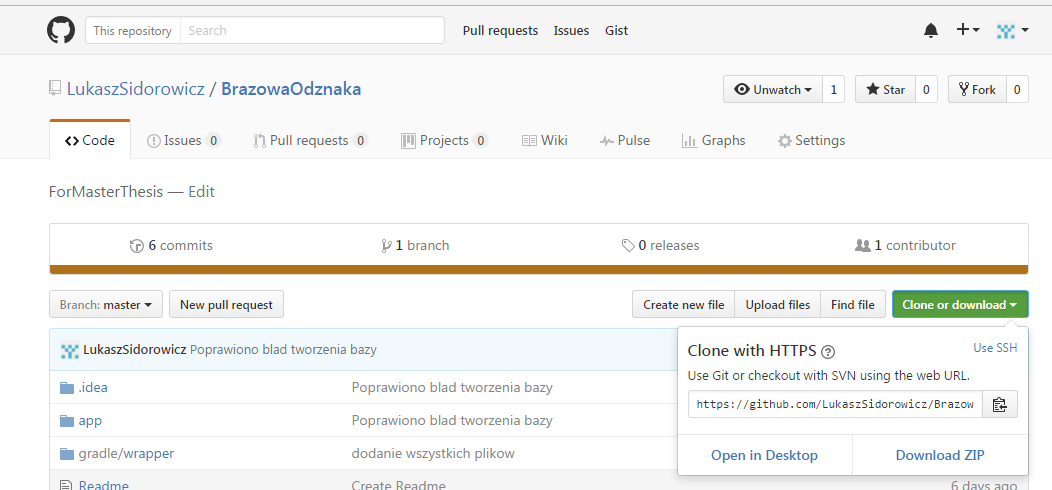


Aplikację można pobrać ze strony:

<https://github.com/LukaszSidorowicz/BrazowaOdznaka>

Aby zainstalować aplikację na telefonie należy pobrać i zainstalować Android Studio i odpowiednie SDK, a następnie wejść na podany adres repozytorium i kliknąć „Clone or Download”

Rysunek 40 Pobieranie projektu z GitHub



Następnie poprany projekt należy zaimportować do Android Studio, podłączyć telefon i zainstalować aplikację. Należy jeszcze pamiętać, aby w telefonie w ustawieniach programisty zaznaczyć opcję debugowania. W komputerach z systemem Windows mogą wystąpić problemy z wykrywaniem telefonu w środowisku programistycznym. Jeśli ktoś dysponuje komputerem z Lunuxem, to zdecydowanie polecam korzystanie z tego systemu do instalacji aplikacji na telefonach i tabletach. Nigdy nie napotkałem tam problemów. Natomiast w komputerach z Windowsem należy upewnić, się czy zainstalowane są sterowniki USB Googla, do pobrania za pomocą SDK menadżera w Android Studio. Czasami pomaga też zmiana trybu połączenia telefonu z komputerem np. na aparat.

1. Badania z użyciem aplikacji.

Dzięki temu, iż od ośmiu lat jestem instruktorem jazdy konnej mam dostęp do jeźdźców zainteresowanych zdaniem egzaminu na odznakę. Z pośród licznej grupy wybrałem piętnaście osób, które podzieliłem na trzy grupy równe wiekowo i podobnym stanie wiedzy początkowej.

Zadaniem postawionym przed grupami było uczenie się przez siedem dni do egzaminu na odznakę przez dwie godziny dziennie, daje to czas po godzinę na dział. Jedna grupa otrzymała smartfony z zainstalowaną aplikacją mobilną, druga grupa otrzymała wydrukowane materiały z pytaniami o odpowiedziami podzielonymi na działy. Trzecia grupa otrzymała obie formy materiałów dydaktycznych.

Grupy zostały tak dobrane, aby reprezentowały podobny wiek, jak również poziom wiedzy. Został przeprowadzony egzamin próbny, gdzie zostało wybrane po jednym pytaniu z każdego działu. Wynik wstępny każdej grupy wynosił 40 %.

1. www.pzj.pl [↑](#endnote-ref-1)
2. Equista.pl [↑](#endnote-ref-2)
3. Pruchniewicz, Wacław; Chaber PR- Akademia Jeździecka; 2003 [↑](#endnote-ref-3)
4. Praca zbiorowa; Akademia Jeździecka; 2008 [↑](#endnote-ref-4)
5. http://digitalfractal.com/2016/05/mobile-operating-systems/ [↑](#endnote-ref-5)
6. http://www.ceneo.pl/43008127 [↑](#endnote-ref-6)
7. http://www.apple.com/pl/shop/buy-iphone/iphone-7 [↑](#endnote-ref-7)
8. http://indianexpress.com/article/technology/tech-news-technology/indus-os-the-indian-operating-system-with-more-installs-that-ios-2876725/ [↑](#endnote-ref-8)
9. http://www.businessinsider.com/apple-ios-v-android-market-share-2016-1?IR=T [↑](#endnote-ref-9)
10. https://developer.android.com/studio/index.html [↑](#endnote-ref-10)
11. https://developer.android.com/distribute/googleplay/start.html [↑](#endnote-ref-11)
12. https://developer.apple.com/support/compare-memberships/ [↑](#endnote-ref-12)
13. https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control [↑](#endnote-ref-13)
14. https://www.gnu.org/software/rcs/ [↑](#endnote-ref-14)
15. https://subversion.apache.org/ [↑](#endnote-ref-15)
16. http://www.nongnu.org/cvs/ [↑](#endnote-ref-16)
17. https://git-scm.com/ [↑](#endnote-ref-17)
18. https://www.mercurial-scm.org/ [↑](#endnote-ref-18)
19. http://bazaar.canonical.com/en/ [↑](#endnote-ref-19)
20. https://github.com/ [↑](#endnote-ref-20)
21. https://bitbucket.org/ [↑](#endnote-ref-21)
22. [https://bitbucket.org/account/user//plans/](https://bitbucket.org/account/user/%3Cusername%3E/plans/) [↑](#endnote-ref-22)
23. <https://github.com/pricing> [↑](#endnote-ref-23)
24. https://www.gartner.com [↑](#endnote-ref-24)
25. https://www.gartner.com/doc/reprints?id=1-2XXET8P&ct=160204 [↑](#endnote-ref-25)
26. http://busitelce.com/bi-analytic-tools-trends/20-gainers-losers-gartner-bi-analytics-2014-vs-2015 [↑](#endnote-ref-26)
27. https://www.coursera.org/learn/analytics-tableau [↑](#endnote-ref-27)
28. http://go.sap.com/product/analytics/bi-platform.html [↑](#endnote-ref-28)
29. http://www.qlik.com/us/ [↑](#endnote-ref-29)
30. https://www.suse.com/products/sles-for-sap [↑](#endnote-ref-30)
31. https://developer.android.com/studio/index.html [↑](#endnote-ref-31)
32. http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html [↑](#endnote-ref-32)
33. https://notepad-plus-plus.org/ [↑](#endnote-ref-33)
34. https://www.sqlite.org/ [↑](#endnote-ref-34)
35. Owens, Mike; The Definitive Guide to SQLite; APRESS; 2006 [↑](#endnote-ref-35)
36. Grant, Allen; Beginning Android; APRESS; 2015 [↑](#endnote-ref-36)
37. http://sqlitebrowser.org/ [↑](#endnote-ref-37)
38. Allen,G.; Owens,M.; *The Definitive Guide to SQLite*, second edition, Apress, 2010 [↑](#endnote-ref-38)
39. MacLean,D.; Komatineni, S.; Allen, G.; *Pro Android 5*, Apress 2015 [↑](#endnote-ref-39)