Rozdział 1

Przegląd zagadnień

Tematem pracy jest stworzenie systemu na platformę Android umożliwiającego pisanie testów na wykładach akademickich i zautomatyzowane przetworzenie wyników testów. W skład systemu planowo mają wejść: aplikacja na system Android oraz aplikacja serwerowa, z którą aplikacja połączy się wysyłając wybrane przez użytkownika odpowiedzi.

Na rynku istnieją już aplikacje częściowo odpowiadające tematowi pracy inżynierskiej. Ich funkcjonalność spełnia pewne aspekty jakimi są: wybór odpowiedzi, przesyłanie odpowiedzi do serwera oraz pobieranie treści z serwera na aplikację. Najlepszym przykładem jest aplikacja nazywająca się Quizowanie [1]. Aplikacja ta pozwala na pobieranie i wyświetlenie pytań z serwera, wyświetlenie możliwych odpowiedzi jako przycisków dla użytkownika oraz wysłanie odpowiedzi do serwera i zweryfikowanie ich. Jej kod źródłowy jest jednak zamknięty co sprawia, że jest nieprzydatny przy tworzeniu systemu będącego tematem tej pracy.

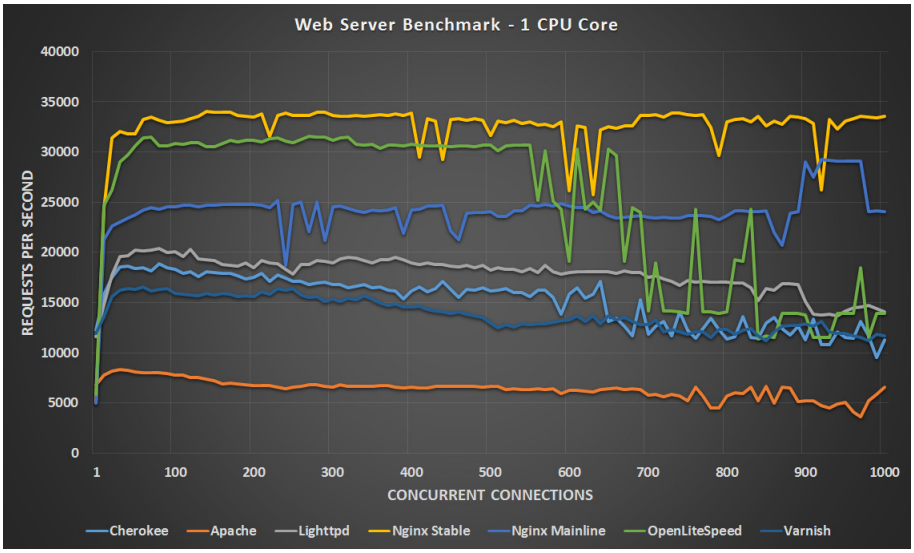
Inne aplikacje, które można wymienić to English Grammar Test[2] lub też sameQuizy[3]. Ich wartość ogranicza się jednak do rozwiązań dotyczących wyglądu interfejsu aplikacji. Nie ma możliwości w legalny sposób uzyskać kodu źródłowego lub opisów rozwiązań w tych aplikacjach z racji ich komercyjnego zastosowania.

W przypadku aplikacji serwerowej jest wiele możliwych technologii, które można zastosować przy jej tworzeniu. Pierwszym etapem był wybór protokołu komunikacyjnego jaki zostanie użyty między aplikacją androidową, a serwerem. Możliwe protokoły to np. TCP, UDP, POP, SMTP, HTTP czy FTP. Wybrany został protokół HTTP z powodu tego, że sieć WiFi na Politechnice Wrocławskiej ma odblokowaną komunikację na portach 80 oraz 8080 oraz prawdopodobnie na tych wymienionych portach sprawdzane są też użyte protokoły i ich zgodność z HTTP. Innego rodzaju protokół wykluczyłby użycie systemu w połączeniu z politechniczną siecią WiFi co znacząco obniżyłoby jego zastosowanie.

Kolejnym etapem był wybór odpowiedniego odpowiedniego serwera web do obsługi zapytań przesłanych z aplikacji mobilnej. Możliwe aplikacje serwerowe jakie można wykorzystać to: Apache, nginx, lighttpd, OpenLiteSpeed. Początkowe wersje aplikacji mobilnej komunikują się z serwerem Apache[4] jednak docelowo ma zostać do tego użyty nginx[5]. Z racji tego, że większość serwerów obsługuje FastCGI pozwalających na napisanie programu obsługującego zapytania w wielu popularnych językach, to przy wyborze serwera kierowano się jego wydajnością obsługi zapytań przychodzących. Wykorzystano tutaj Linux Web Server Performance Benchmark[6] z 2016.

Należało też wybrać środowisko w jakim aplikacja mobilna powinna powstać. W tym przypadku do wyboru jest Android Studio[7] oraz Qt[8]. Oba środowiska oferują duże wsparcie ze strony społeczności programistycznej oraz szeroki zestaw narzędzi. Oba pozwalają na tworzenie interfejsu użytkownika w XML oraz udostępniają klasy obiektów przydatne do wizualizowania i przetwarzania danych. Wybrany został jednak Android Studio z racji jego największej popularności oraz najszerszej gamy narzędzi

[1] https://play.google.com/store/apps/details?id=se.feomedia.quizkampen.pl.lite
[2] https://play.google.com/store/apps/details?id=english.grammar.test.app
[3] https://play.google.com/store/apps/details?id=pl.filing.samequizy
[4] https://httpd.apache.org/
[5] https://www.nginx.com/
[6] https://www.rootusers.com/linux-web-server-performance-benchmark-2016-results/
[7] https://developer.android.com/studio/index.html
[8] https://www.qt.io/



Rysunek 1.1: Wykres przedstawiający ilość obsługiwanych zapytań przy danej ilości otwartych połączeń z Linux Web Server Performance Benchmark

wchodzących w skład tego środowiska. Pozwala na dołączanie bibliotek z repozytoriów Git lub też bibliotek za pomocą wbudowanego narzędzia Maven[9], bieżące sprawdzanie kompatybilności użytych bibliotek z wersjami systemu Android na które aplikacja powstaje oraz tworzenie maszyn wirtualnych do testowania napisanych aplikacji.
Zważywszy na powyżej przedstawione informacje system do rozwiązywania testów musi być napisany od podstaw. Nie znaleziono przykładów otwartych systemów na tyle użytecznych, żeby można było użyć ich przy tworzeniu systemu. Systemy takie jak Quizowanie mają zamknięty kod i można się nimi sugerować jedynie w kwestii tworzenia interfejsu użytkownika.

W trakcie pracy nad systemem do rozwiązywania testów wymagania co do tego systemu zostały doprecyzowane. Aplikacja mobilna musi być przyjazna dla użytkownika i intuicyjna. Użytkownik dalej ma możliwość pisania testów na papierze dlatego forma elektroniczna musi być wystarczająco zachęcająca. Ponadto musi też być niezawodna. Nie powinna tracić przetwarzanych w niej istotnych informacji lub wyłączać się w trakcie rozpoczęcia pisania testu. Aplikacja musi się też w niezawodny sposób łączyć z serwerem w celu przesłania odpowiedzi użytkownika lub też pozwolić na ponowne ich wysłanie po utracie połączenia. Kopie zapasowe testów muszą też być zapisywane w pamięci wewnętrznej telefonu, a ich zawartość zaszyfrowana chroniąc wrażliwe dane przed ingerencją osób nieupoważnionych. Aplikacja powinna też być weryfikowalna przez serwer, żeby nie dopuścić do wysyłania odpowiedzi aplikacji zmodyfikowanych przez osoby trzecie. Musi też informować użytkownika odpowiednimi komunikatami wyświetlanymi na ekranie o zaistniałych problemach na każdym etapie użytkowania, żeby użytkownik wiedział w jaki sposób reagować na nieprzewidziane sytuacje.

Aplikacja serwerowa musi być w stanie obsługiwać sprawnie wysoki ruch, w którym przybywa średnio 10 zapytań na sekundę dochodzących nawet do 100 na sekundę przy 100 osobowej grupie użytkowników piszących na aplikacji test. Docelowo powinna jednak być w stanie obsłużyć nawet 300-400 użytkowników, żeby można było ją zastosować w każdej możliwej grupie studentów na Politechnice Wrocławskiej. Serwer musi też być w stanie weryfikować łączące się aplikacje, tak aby nie przesyłać wrażliwych informacji do nieupoważnionych osób. Jego struktura musi być możliwie prosta, żeby przyszłe modyfikacje były mało uciążliwe dla użytkownika całego systemu.

[9]https://maven.apache.org/

Rozdział 2

Stopień zaawansowaia prac

Aplikacja mobilna przechowuje dane użytkownika takie jak: Imię, Nazwisko, numer indeksu oraz nazwa przedmiotu na którym wykonywany jest test. Umożliwia też zmianę adresu serwera, na który przysyłane są odpowiedzi z testu. Dane te zmienia się w ustawieniach aplikacji. W menu głównym pozwala na wprowadzenie rzędu i miejsca, na którym student siedzi podczas testu, wektora wag służącego do obliczania grupy oraz ID testu jaki jest przeprowadzany. Zostało też zaimplementowane udogodnienie dla studenta w postaci skanera kodów QR umożliwiającego zautomatyzowane wczytanie wektora wag, ID testu i wyliczenie grupy studenta. Do skanowania kodów QR została wykorzystana biblioteka zxing[1] (Zebra Crossing). Po przejściu do testu wyświetlany jest ekran konfiguracyjny aplikacji gdzie student jest informowany o operacjach przeprowadzanych przed rozpoczęciem testu. Na obecnym etapie prac aplikacja sprawdza osiągalność serwera, do którego wysyła odpowiedzi oraz pobiera plik konfiguracyjny zawierający takie dane jak minimalna i maksymalna wersja aplikacji dopuszczona do testu oraz klucz szyfrowania danych w plikach testu przechowywanych na telefonie użytkownika. Po przejściu do testu wyświetlana jest zakładka pytania, a na niej: nr grupy w jakiej student jest, nr pytania, przyciski do wysłania odpowiedzi "tak", "nie", "nie wiem"; przyciski do dodania pytania i do podsumowania testu oraz unikalny identyfikator sesji wygenerowany dla aktualnej sesji testowej otwartej na telefonie. Dodanie pytania wymusza przejście do następnego pytania a UI użytkownika pozwala na przesuwanie ekranu w celu dostania się do innych pytań. Podsumowanie testu wyświetla ilość poprawnie przesłanych odpowiedzi "tak", "nie" oraz "nie wiem" do serwera, oraz ilość odpowiedzi zapisanych w pliku testowym. W razie różnicy w odpowiedziach przesłanych do serwera a zapisanych w pliku aplikacja wyświetla odpowiednie ostrzeżenie sugerujące użytkownikowi powrót do testu i ponowne przesłanie odpowiedzi. Na ekranie podsumowania wyświetlany jest przycisk pozwalający powrót do testu oraz przycisk kończący test powodujący powrót do menu głównego aplikacji.

Obecnie serwerem do obsługi zapytań sieciowych jest Apache. Do przetwarzania danych wykorzystano skrypt napisany w języku AWK. Skanuje on logi serwera Apache w poszukiwaniu zarejestrowanych zapytań przysłanych z aplikacji mobilnych i umożliwia zapisanie ich w formacie użytecznym dla użytkownika całego systemu.

Stworzony został też deszyfrator plików przechowywanych na telefonach studentów pozwalający na odzyskanie odpowiedzi studenta w razie problemów z połączeniem z serwerem gdy nie wszystkie odpowiedzi do niego dotarły.

[1] https://github.com/zxing/zxing

Rozdział 3

Elementy aplikacji

3.1 Widoki aplikacji

Wygląd aplikacji stanowi ważny element aplikacji. W początkowych fazach projektu rozmieszczenie elementów na ekranie telefonu komórkowego jak też ich estetyka wielokrotnie była zmieniana. Potrzebne było wypracowanie przejrzystego oraz przyjaznego dla studenta interfejsu użytkownika. W trakcie tworzenia aplikacji zauważono, że istotnym elementem jest zachowanie kompatybilności z różnymi wersjami systemu Android realizując jednolite i przewidywalne zachowanie się interfejsu aplikacji. Na przykład nie można było zastosować kolorowania przycisków korzystając z metody setColorFilter obecnej we wszystkich wersjach systemu od Android 4.0 do Android 7.0. Wywołanie tej metody w systemach Android 4.0 do 4.4 powodowało zakolorowanie przycisku w sposób ujawniający się dopiero po odtworzeniu widoku. W systemach Android 5.0 i wyższych powodowało natychmiastową zmianę koloru przycisku. Widoki jakie zostały zaimplementowane w aplikacji to:

- Widok główny
  Za jego pomocą użytkownik może wprowadzić swoje miejsce i rząd w jakim się znajduje, wektor wag służący do wyliczenia grupy na teście oraz kod testu. Na tym widoku można uruchomić również skaner kodów QR do automatycznego pobrania wektora wag i kodu testu oraz wyliczenia grupy na teście. Trzy dolne przyciski służą do wyjścia z aplikacji, ręcznego wyliczenia numeru grupy oraz przejścia do testu. Z tego widoku można również otworzyć menu, z którego można dostać się do widoku ustawień i widoku informacji.

- Widok ustawień
  Tutaj użytkownik może wprowadzić i zapisać w aplikacji swoje dane takie jak: imię, nazwisko, numer indeksu oraz nazwa przedmiotu. Opcjonalnie może wprowadzić inny adres serwera, na który aplikacja będzie przesyłać odpowiedzi użytkownika.

- Widok informacji
  W tym widoku użytkownik może się dowiedzieć z jakiej wersji aplikacji korzysta

- Widok konfigurowania testu
  Tutaj użytkownik jest informowany o przebiegu połączenia testowego z serwerem, pobrania pliku konfiguracyjnego, sprawdzeniu poprawności wersji aplikacji oraz sprawdzeniu poprawności otrzymanego klucza szyfrowania.

- Widok testu
  W tym miejscu użytkownik może wybierać odpowiedzi na pytania o danym numerze, dodać kolejne karty, z których może wysłać następne odpowiedzi oraz przejść do podsumowania testu gdzie dowie się w liczbach jakie odpowiedzi wybrał oraz ile z tych odpowiedzi znajduje się w pliku testu na telefonie. Może też wrócić do testu aby go kontynuować lub też go zakończyć wracają co do widoku głównego.

3.2 Interaktywne elementy na widokach

Rozwijane listy z testami w widoku ustawień.

Rozwijane propozycje kodów testu w widoku głównym.

W aplikacji zastosowano również interaktywne elementy sygnalizujące operacje zachodzące w trakcie korzystania z aplikacji. Pierwszym z nich jest przycisk o nazwie Android Circular Progress[1] Button został stworzony przez Danylyk Dmytra i udostępniony na licencji MIT. Przycisk ten w aplikacji zmienia swój kolor oraz wygląd sygnalizując:

- bezczynność - niebieski pusty przycisk
- przetwarzanie operacji - wirujące kółko
- powodzenie przetwarzania operacji - zielony przycisk z checkmark
- niepowodzenie przetwarzania operacji - czerwony przycisk

Bezczynność została początkowo użyta dla jednej nieaktywnej funkcji, którą była walidacja aplikacji. Zielony kolor wykorzystany został do sygnalizowania powodzenia połączenia z serwerem oraz do sygnalizowania poprawnego pobrania pliku konfiguracyjnego z walidacją klucza szyfrowania i dopuszczalnej wersji aplikacji. Czerwony kolor został wykorzystany do sygnalizowania niepowodzenia wyżej wymienionych operacji. Wirujące kółko trwało tak długo jak te asynchronicznie wykonywane operacje nie zakończyły swojego działania.

tutaj będą obrazki pokazujące jak zmienia się przycisk

Drugim z nich są przyciski występujące na kartach z odpowiedziami w aplikacji. Przyciski te sygnalizują swoimi kolorami etap przetwarzania odpowiedzi użytkownika. Biały przycisk sygnalizuje brak wybranej do tej pory odpowiedzi. Szary przycisk sygnalizuje poprawnie zapisaną odpowiedź do pliku testowego. Niebieski przycisk sygnalizuje poprawnie wysłaną odpowiedź na serwer. Jeżeli operacja zapisywania do pliku się nie powiedzie to przycisk pozostaje biały. Jeżeli operacja wysyłania do serwera się nie powiedzie to przycisk zakolorowywany jest jedynie na szaro informując użytkownika o tym, że jego odpowiedzi przechowywane są jedynie lokalnie na telefonie.

tutaj będą obrazki pokazujące przykładowo wybraną odpowiedź

3.3 Powiadomienia

Istotnym elementem aplikacji okazały się być powiadomienia pozwalające na zrozumienie użytkownikowi wydarzeń występujących w aplikacji. Dzięki wypracowaniu zestawu chmurek oraz monitów student jest informowany o powodzeniu w edycji ustawień aplikacji, błędach występujących przy konfigurowaniu testu oraz różnych innych zdarzeń takich jak utrata połączenia z serwerem albo brak poprawnego testu, na który użytkownik chce wysyłać odpowiedzi.

3.4 Szyfrowanie plików

Początkowo pliki były szyfrowane algorytmem DES z kluczem generowanym na podstawie łańcucha znaków. Szyfrowało to tekst w dość niezadowalający sposób ujawniając regularności w zakodowanym tekście. Ponadto klucz szyfrowania można było uzyskać po zdekompilowaniu aplikacji. Pierwszym ulepszeniem zaimplementowanym w aplikacji było zastosowanie metody "security by obscurity", gdzie klucz szyfrowania zmieniał się w trakcie działania programu. W ten sposób sprawa jego uzyskania została utrudniona. W dalszym etapie wykorzystano szyfrowanie algorytmem RSA. Klucz publiczny jest wykorzystywany w aplikacji do szyfrowania odpowiedzi i zapisywania ich do pliku testu. Uzyskano w ten sposób zadowalający nieregularny wygląd odpowiedzi w pliku testu. Aplikacja została też zmodyfikowana w taki sposób, że klucz publiczny jest pobierany w pliku konfiguracyjnym z serwera pozbywając się niebezpiecznej metody z kluczem bezpośrednio zapisanym w aplikacji.

3.5 Skaner QR

W aplikacji wykorzystano również skaner kodów QR. Jak wcześniej o tym wspomniano, wykorzystano do tego bibliotekę zxing. Pozwala ona nie tylko na skanowanie kodów QR ale również kodów

[1] https://github.com/flavioarfaria/circular-progress-button

kreskowych, kodów Aztec i innych. Skaner skanuje kod QR wyświetlany na rzutniku pozwalając na automatyczne wpisanie kodu testu, wektora wag i wyliczenie grupy na teście. Ułatwia to pracę z aplikacją potencjalnemu użytkownikowi oraz pozwala na zmniejszenie prawdopodobieństwa popełnienia błędu przez użytkownika przy wyliczaniu grupy na teście.

Rozdział 4

Elementy serwera

4.1 Generator konfiguracji

4.2 Deszyfrator

4.3 Generator kodów QR