Tadeusz NIEDZIELA, Sebastian CZERWATY, Kacper DYGAS

APLIKACJA INTERNETOWA LISTSBOOK

Opracowano oraz zaimplementowano rozbudowaną aplikacje internetową posiadającą własny interfejs programistyczny API (ang. application programming interface). Jej zadaniem jest magazynowanie oraz prezentacja danych zapisanych w postaci spersonalizowanych grup zwanych „listami". Listy będzie można współdzielić pomiędzy użytkownikami aplikacji bądź też samo organizującymi się grupami użytkowników.

WSTĘP

Ogrom napływających informacji, aktualnie stwarza potrzebę bezpiecznego ich magazynowania, analizy, czy też możliwości dzielenia się nimi. Jesteśmy przyzwyczajeni do wygody i prostoty. Dodatkowo chcemy, aby w "cyfrowym świecie" wszystko było do natychmiastowego wykorzystania. Wychodząc naprzeciw potrzebom dzisiejszego świata, autorzy niniejszej pracy proponują aplikację internetową "ListsBook" do tworzenia, udostępniania i przedstawiania spersonalizowanych list z danymi.

"ListsBook" jest rozbudowaną aplikacją internetową, która ma na celu w prosty sposób gromadzić dane zamieszczane przez użytkowników w postaci grup zwanych "listami". Każda taka "lista" jest spersonalizowanym zbiorem danych, który możemy dowolnie modyfikować: zmieniać nazwę listy, opis, kolor, obraz, dodawać dane do listy, edytować je a także udostępniać. Wszystko po to aby ułatwić użytkownikowi dopasowanie swoich danych do jego potrzeb.

Dane umieszczane w listach mogą być dowolnego typu (tekst, obraz, dźwięk, plik). Wiele z nich da się otworzyć za pomocą proponowanej aplikacji bądź samej przeglądarki internetowej. Dane, które nie są możliwe do otwarcia przy pomocy "ListsBook" są pobierane na urządzenie, z którego użytkownik korzysta z aplikacji. Daje to możliwość pełnej ich obsługi i zamieszczania w serwisie. Wszystko co zostaje umieszczone w serwisie musi posiadać swoją nazwę, która pomoże w ewentualnym odszukaniu. Pojedynczy element listy (czyli np. plik) można spersonalizować za pomocą wyżej wspomnianej nazwy czy też koloru. Każdy element listy można następnie modyfikować, czy też usuwać z listy. Dane umieszczane w listach można przedstawić na kilka różnych sposobów.

Każdą z list można udostępnić poszczególnym użytkownikom, bądź też samo organizującym się grupom użytkowników. Daje to możliwość łatwego udostępnienia danych, a także zrzeszania się osób według ich własnych preferencji. Przykładem mogą być np. miłośnicy muzyki Chopina, którzy mogą utworzyć swoją grupę i udostępniać w niej własne listy z ulubionymi utworami.

Ponieważ Internet jest siecią ogólnoświatową, postanowiono zadbać, aby osoby pochodzące z rożnych krajów mogli skorzystać z aplikacji w ich własnym języku. Z tego względu "ListsBook" dostępny jest w trzech wersjach językowych: polskim, angielskim oraz hiszpańskim. Zapewnia to większą dostępność i czytelność aplikacji, a tym samym większe zadowolenie użytkowników. W dalszych etapach rozwoju aplikacji można w prosty sposób dodać kolejne języki.

Interface aplikacji został zaprojektowany w taki sposób, aby użytkownik mógł korzystać z niego w sposób intuicyjny. Aby dodatkowo pomóc użytkownikom w korzystaniu z aplikacji jest ona wyposażona w dodatkowe komunikaty, które ułatwiają obsługę. Design oparty jest na rozwiązaniach dostarczanych przez firmę Google (Material Design), który jest miły dla oka, przejrzysty i dobrze znany użytkownikom. Aby wzbogacić doświadczenia użytkowników przy korzystaniu z aplikacji jej interface ma dwie szaty kolorystyczne, które można w prosty sposób zmieniać.

Wybór technologii [1-16] w których wykonano aplikację nie był przypadkowy. Stos technologiczny został wybrany jako rozwiązanie najbardziej optymalne i dające naszym zdaniem najlepsze rezultaty. Umożliwia on nie tylko sprawną implementację ale także rozwój i "wyjście" projektu poza przeglądarkę internetową. Mamy tu na myśli aplikacje na platformy mobilne czy też aplikacje desktopowe. Są to najnowocześniejsze narzędzia dostępne na świecie, poza tym za rozwój tych technologii odpowiedzialne są firmy takie jak Google czy Microsoft, co gwarantuje ich stabilność oraz rozwój. Na dodatek skupiona jest wokół nich ogromna społeczność programistów z całego świata, dzięki czemu ewentualne problemy z implementacją można omówić z innymi użytkownikami, co pozwoli na szybsze rozwiązanie problemu.

Bezpieczeństwo jest w dobie Internetu aspektem kluczowym, dlatego też informacje przesyłane przez użytkownika są zabezpieczone nie tylko podczas ich przesyłania, ale także po umieszczeniu ich w szyfrowanej bazie danych. Aby zabezpieczyć użytkownika przed nieautoryzowanym dostępem w bazie danych zapisywane są dane jego ostatniego logowania (czas i miejsce) tak by później móc wychwycić próby ewentualnego przechwycenia danych.

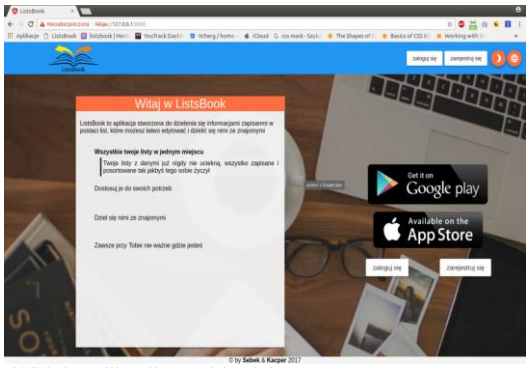
Współpraca pomiędzy programistami bywa czasem bardzo ciężka, dlatego postanowiono posłużyć się narzędziami i metodami, które ułatwiają organizację pracy.

Nadzieją autorów jest to aby projekt w przyszłości nadal ewoluował, podlegał rozwojowi, a tym samym był chętnie wykorzystywany przez użytkowników.

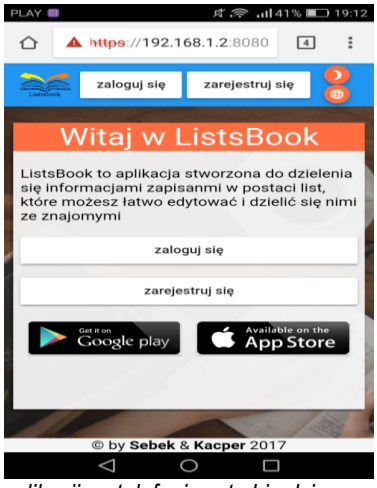
1. IMPLEMENTACJA APLIKACJI

1.1. Strona główna

W pierwszej kolejności zostanie zaprezentowana implementacja aplikacji ListsBook przy użyciu aktualnych technologii. Aby zapewnić przejrzystość, omawianie implementacji podzielone zostanie na poszczególne strony (widoki) aplikacji na które może wejść użytkownik, bowiem każdy z widoków reprezentuje odrębną funkcjonalność aplikacji.



Rys. 1. *Widok aplikacji w trybie dziennym*



Rys. 2. *Widok aplikacji na telefonie w trybie dziennym*

Rysunki powyższe przedstawiają stronę główną proponowanej aplikacji ListsBook. Jest to pierwszy widok przez który przechodzi użytkownik odwiedzając stronę. Jest to responsywna dynamiczna strona internetowa która ma za zadnie poinformować i zachęcić użytkownika do skorzystania z aplikacji [1-3].

Aby uatrakcyjnić user experience, strona główna (jak i inne widoki aplikacji) ma dwa tryby - dzienny i nocny. Tryb dzienny aktywowany jest od godziny od 6:00 do 22:00. Jest on złożony z jasnych kolorów (odcień pomarańczy i niebieskiego), które naturalnie kojarzą się z dniem. Tryb nocny aktywuje się automatycznie w godzinach wieczornych i nocnych (od 22:00 do 6:00). Kolory w nim nawiązują do nocy (odcień czarnego/szarości oraz kontrastujący żółty). Użytkownik może zmienić tryb według własnego uznania niezależnie od godzin klikając na przycisk zmień tryb. Funkcjonalność tę zapewnia Angular Material Design [9], który zmienia zdefiniowany schemat kolorystyczny aplikacji. W praktyce odbywa się to za pomocą dodania klas do obiektów DOM co z kolei powoduje zmianę ich styli CSS [4].

Informacje i komunikaty wyświetlane na stronie dostosowane są do języka użytkownika. Definiowanie języka który będzie używany w aplikacji jest zrealizowane w oparciu o język aktualnie ustawiony w przeglądarce [6-8].

Oczywiście nic nie stoi na przeszkodzie aby użytkownik samodzielnie dostosował język. Odbywa się to za pomocą przycisku zmień język a następnie kliknięciu na flagę państwa w którym dany język jest językiem urzędowym.

Technicznie zrealizowane jest to przy pomocy dodatkowego modułu Frameworka Angular o nazwie "ngx-translate" [12, 15]. Pliki z tłumaczeniami są zapisane w formacie JSON a nazwą jest kod państwa (pl,en,es). Na podstawie języka przeglądarki wybierany jest plik o nazwie kodu państwa w którym zawarte są tłumaczenia. Np. jeśli język przeglądarki jest językiem polskim wybrany kod to "pl" a plik z tłumaczeniem "pl.json" (i odpowiednio dla innych języków). Tłumaczenia zawarte w pliku są pobierane i na podstawie kluczy podstawiane są w odpowiednie miejsca strony. Warto tu zaznaczyć, że zmiana języka nie wymaga przeładowania strony.

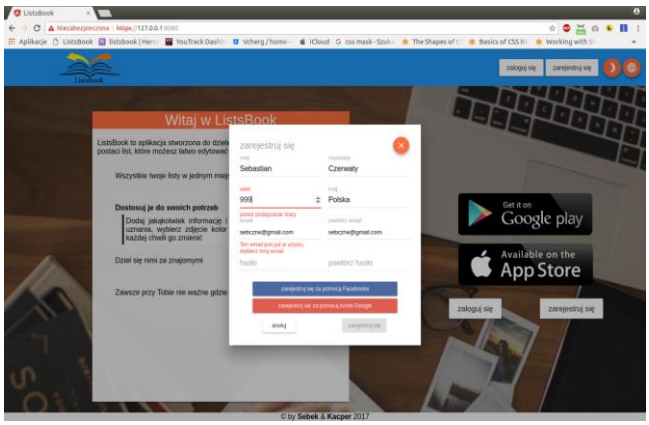
Po zmianie języka aplikacja pobiera jedynie plik z danym tłumaczeniem i podstawia słowa i zwroty w odpowiednie miejsca.

Przyciski na stronie głównej jak i w innych widokach aplikacji są animowane tak aby polepszyć user experience i zachęcić do kliknięcia. Ponieważ niektóre z nich nie zawierają napisów (a zamiast nich ikony-piktogramy), które mówią do czego dany przycisk służy, zachodzi obawa, że mogą być one nie czytelne dla niektórych użytkowników. Aby rozwiązać ten problem Angular Material [9] oferuje wyskakujące "dymki" z napisami, popularnie nazywane tooltipami. Jak sama nazwa wskazuje służą one do informowania do czego służy dane narzędzie a w naszym przypadku przycisk.

Rolę informacyjną spełnia animowany slider po lewej stronie przypominający listę. Informuje on o podstawowej funkcjonalności i celu aplikacji. Animacje mają przykuć uwagę użytkownika a jeśli użytkownik zechce przeczytać dokładniej dany punkt wystarczy, że najedzie na niego myszą a animacja zostanie wstrzymana natomiast punkt rozwinięty. Slider nie jest widoczny na urządzeniach o małych ekranach (np. telefony komórkowe) z uwagi na oszczędność miejsca. Jedyną jego "pozostałością" jest zdanie wprowadzające.

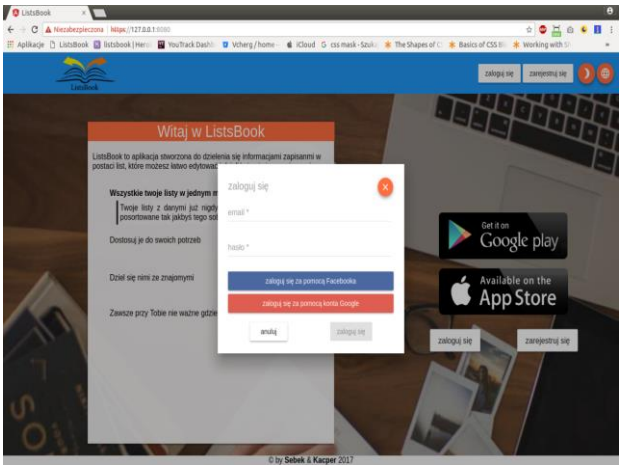
Z myślą o dalszym rozwoju aplikacji dodano przyciski, które docelowo będą przenosić użytkownika do sklepu Google Play lub AppStore ze stroną naszej aplikacji na urządzenia mobilne którą będzie można zainstalować (bez stosownej opłaty) na naszym urządzeniu.

Najważniejszymi funkcjami dostępnymi na stronie głównej jest logowanie i rejestracja. Obydwa przyciski powodują wyświetlenia okien modalnych z odpowiednim formularzem logowania bądź rejestracji.



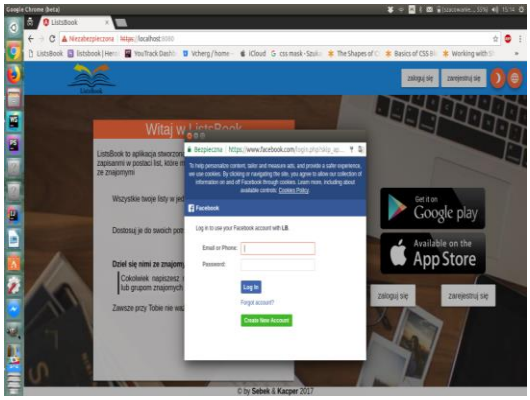
Rys. 3. *Formularz rejestracji*

Każdy z formularzy jest walidowany według ściśle określonych kryteriów, tak aby zapobiec wprowadzaniu błędnych danych bądź też danych w błędnym formacie. W walidacji formularza pomaga framework Angular [16], który dostarcza gotową funkcjonalność do walidacji formularzy w dwóch rodzajach Template Driven i Reactive Driven. Oba podejścia zastosowano w proponowanej aplikacji.

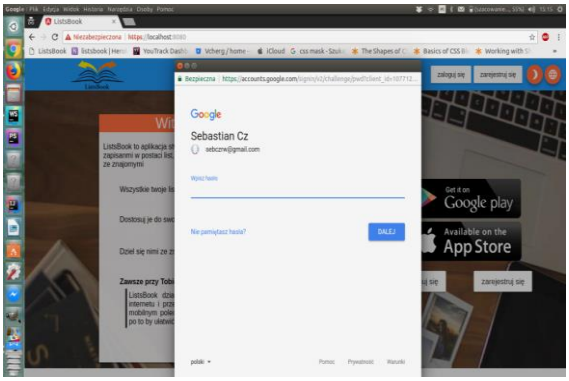


Rys. 4. *Formularz logowania*

Dla wygody użytkownika wprowadzono możliwość zalogowania się przy użyciu konta Facebook bądź Google [13]. Wymagało to zarejestrowania aplikacji w platformach deweloperskich obu serwisów oraz napisaniu i podłączeniu specjalnych funkcji oraz bibliotek które pozwalają na pozyskanie danych z ww. serwisów. Jest to tak zwane O-Auth. Oznacza to, że zewnętrzny serwis odpowiedzialny jest za walidację, potwierdzanie i dostarczanie danych użytkownika.



Rys. 5. *Logowanie przez Facebook*



Rys. 6. *Logowanie przez Google*

2. SYSTEM AUTORYZACJI

System autoryzacji (inaczej system logowania/rejestracji) służy do uzyskania dostępu do swojego konta na danej stronie, aplikacji internetowej. Dzięki systemowi autoryzacji użytkownicy po zalogowaniu mają dostęp do danych widocznych tylko dla nich. Dzięki logowaniu możemy określać dokładnie z jakiego miejsca i o której godzinie użytkownik wszedł na swoje konto. Służy to następnie do zbierania informacji o danej osobie i o jej preferencjach. Najczęściej akceptując regulamin danego serwisu podczas rejestracji użytkownicy dobrowolnie zgadzają się na to że po podaniu swoich danych osobowych i zarejestrowaniu się serwis może wykorzystywać te dane, np. sprzedawać je.

W przedstawionym projekcie są 3 możliwości autoryzacji:

- zwykłe logowanie/rejestracja
- logowanie przez Facebooka
- logowanie przez Google

Dane o użytkownikach przechowywane są w bazie danych mongodb [14] do której stworzyliśmy odpowiedni dynamiczny model „tabeli" (obiekt JSON) do przechowywania tych danych.

2.1. Zwykłe logowanie /rejestracja

Zaproponowaliśmy autorski system rejestracji i logowania, korzystając z możliwości jakie daje NodeJs pod względem zabezpieczeń kryptograficznych. Są to bardzo zaawansowane zabezpieczenia. Klucze autoryzacyjne (tokeny) są inne przy każdym logowaniu tego samego użytkownika. Hasło które podaje użytkownik jest poddawane kodowaniu funkcją kryptograficzną sha512 [10]. Użyliśmy tego rozwiązania ponieważ jest bezpieczne i raz zakodowanych danych nie można odkodować, aby sprawdzić podczas logowania czy wpisane hasło zgadza się z wcześniejszym. Zakodowanym hasłem korzysta się z klucza jakim zakodowaliśmy hasło w naszej aplikacji jest to pole „salt". Funkcja do kodowania hasła, jak i zapisu klucza została przedstawiona na rysunku 2.11.

```
UserSchema.methods.setRandomSalt = function
(length) {
   return crypto.randomBytes(Math.ceil(length/2))
      .toString('hex') /** convert to hexadecimal format
*/
      .slice(0,length);   /** return required number of
characters */
}

UserSchema.methods.hashPassword = function (pass-
word,salt) {
 var hash = crypto.createHmac('sha512', salt); /** Hash-
                ing algorithm sha512 */
   hash.update(password);
var value = hash.digest('hex');
   return {
      salt:salt,
      passwordHash:value
   };
}
```

Rys. 7. *Funkcja ustalająca salt, oraz metoda kodująca hasło*

Dane jakie podaje użytkownik podczas rejestracji i logowania przetrzymywane są w bazie danych MongoDB [11, 14], w tablicy Users. Dzięki temu możemy je w łatwy sposób odczytywać. Warto również zwrócić uwagę na funkcje która sprawdza czy nie mamy do czynienia z włamaniem się na konto „isAuthorized". Funkcja ta sprawdza czy została stworzona sesja dla użytkownika po stronie serwera. Sprawdza czy token przysyłany z widoku jest taki sam jak token zapisany w sesji po stronie serwera (bo samo sprawdzanie danych zapisanych w pamięci przeglądarki po stronie klienta nie wystarczy aby określić czy jest on zalogowany i czy informacje które są mu wyświetlane należą do niego). Jeśli wszystko przebiega prawidłowo, po wcześniejszym sprawdzeniu danych przez funkcje „isAuthorized", czynność jaką wybrał użytkownik jest wykonywana. Jeśli natomiast klucz autoryzacyjny token nie zgadza się z tokenem

zapisanym na serwerze to użytkownik dostaje komunikat o błędzie i musi się ponownie zalogować.

2.2. Logowanie przez Facebooka

Kolejną opcją logowania w naszej aplikacji jest opcja logowania przez Facebooka. Użytkownicy nie muszą tworzyć kolejnego konta, tylko za pomocą Facebooka mogą uzyskać dostęp do naszego serwisu. Logowanie przez konta społecznościowe do różnych aplikacji jest możliwe dzięki autoryzacji oAuth. Oauth jest popularnym standardem autoryzacji współczesnego Internetu, jest to zezwolenie udzielanie konkretnej stronie na korzystanie informacji zawartych w koncie innej witryny albo po prostu w celu weryfikacji naszej tożsamości [13].

Aby była możliwość logowania przez Facebooka zarejestrowaliśmy naszą aplikacje na stronie Facebooka uzyskując w ten sposób klucz dla naszej strony www. Gdy jakiś użytkownik się loguje za pomocą Facebooka, nasz serwis jest sprawdzany z danym kluczem i jeśli klucz się zgadza następuje autoryzacja użytkownika który się chce zalogować. Przy pierwszym logowaniu musi zezwolić naszej aplikacji na pobieranie danych o nim (dane wybieramy sami, w późniejszym celu ulepszenia naszej aplikacji). Facebook udostępnia bardzo wiele danych które możemy pobierać o danym użytkowniku np. jego zdjęcia, listę znajomych, wszystkie informacje o nim, można również publikować posty w imieniu użytkownika.

Po kliknięciu opcji „Zaloguj przez Facebook", jesteśmy przekierowani na stronę Facebooka i po wpisaniu swoich danych oraz zaakceptowaniu danych pobieranych przez nasz serwis zostajemy przekierowani z powrotem oraz zalogowani do serwisu. Dane przesyłane są najpierw do kodu frontedu dokładnie do kontrolera (Angular2) zajmującego się Autoidentyfikacją. Następnie jeśli logowanie przebiegło poprawnie Facebook zwraca nam dane o użytkowniku takie jak np. jego id co identyfikuje go w naszym serwisie. W kolejnym kroku wysyłane są do serwera (NodeJS), aby zapisać nowo zalogowanego użytkownika w naszej bazie, jeśli jeszcze nie istnieje lub aby wyświetlić spersonalizowane dane, jeżeli jest to jego kolejne logowanie na naszej stronie. Rozwiązaliśmy tym samym problem, w jaki sposób przypisać konto użytkownika na danej stronie gdy loguje się przez portal społecznościowy. Logowanie to jest wygodną opcją ponieważ większość ludzi korzysta z Facebooka.

2.3. Logowanie przez Google

Opcja autoryzacji (logowania) przez Google zaimplementowana została w naszym projekcie dla większej wygody użytkowników. Działa bardzo podobnie do logowania przez Facebooka. W pierwszej kolejności musimy zarejestrować naszą stronę w narzędziach dla programistów w Google. Wtedy otrzymujemy nasz identyfikator aplikacji, który jest kluczem dostępu, dzięki któremu Google wie że łączy się z naszą aplikacją. Dane zalogowanych użytkowników przechowywane są w bazie danych MongoDB [14]. Google udostępnia szereg danych jakie możemy pobierać o danej osobie gdy zaakceptuje połączenie z naszym serwisem.

Przy tworzeniu widoku (jak i wszystkich innych w naszej aplikacji) od strony backendu skorzystaliśmy z REST API, a dokładniej operacji CRUD (Create, Read, Update, Delete). Komunikacja pomiędzy stroną frontendu i stroną backedu jest zaimplementowa za pomocą adresów URL tzn. widok wysyła poprzez połączenie http do strony serwera dane za pomocą metod post lub get w zależności od potrzebnej operacji. Uzyskaliśmy tym samym uniwersalny system API który można w późniejszym czasie wykorzystać do aplikacji mobilnej. Taki uniwersalny system komunikacji po stronie serwera otwiera drogę, aby w późniejszym czasie udostępnić nasze API osobą publicznym, dzięki temu dane z naszej strony mogły by być widoczne na innych stronach www.

Po zalogowaniu użytkownika widać wszystkie jego listy jakie stworzył. Uzyskaliśmy to prostym poleceniem „db.find" z mongodb, które służy do wybierania wszystkich elementów z danej tabeli.

Udostępnienia listy jest to opcja gdzie można udostępnić innemu użytkownikowi lub danej grupie wybranej listy. Działa to w ten sposób że po kliknięciu przycisku „udostępnia", zostaje wysyłane zapytanie do API serwera o pobranie wszystkich użytkowników oraz grup jakim lista została udostępniona. Jeśli w odpowiedzi dostaniemy ze lista nie jest udostępniana to stosowny komunikat zostanie wyświetlony użytkownikowi, natomiast gdy lista posiada udostępnienia zostanie zwrócony obiekt z tymi danymi, który jest później formatowany przez stronę frontendu Angular 2 i następnie wyświetlany dla użytkownika. Gdy dana lista zostanie usunięta przez jej twórcę, znika automatycznie u osób, oraz grup jakim była udostępniona.

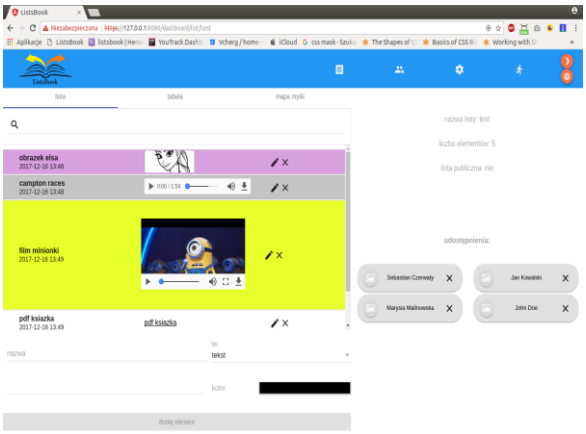
Edycja listy - funkcjonalność jak sama nazwa wskazuje jest edytowaniem listy. Może tego dokonać tylko jej właściciel i nigdy żadne inne osoby, nawet jeśli lista jest im udostępniona. Uzyskaliśmy to sprawdzając przy każdym zapytaniu czy użytkownik jest zalogowany i jest właścicielem listy, poprzez wysłania tokena (zapisanego w pamięci podręcznej) ze strony frontendu do backendu. Każda lista ma w sobie pole creator_id (id użytkownika, który stworzył listę) dzięki czemu łatwe jest określenie kto jest jej twórcą. Odpowiednio edytowana lista po kliknięciu przycisku „gotowe" jest przesyłana na serwer, a tam jej dane są aktualizowane i zapisywane w bazie danych mongodb w tabeli Lists.

Usuwanie listy jest bardzo przydatną opcją jeżeli dana lista nie będzie już potrzebna. Użytkownik może w dowolnej chwili usunąć stworzoną przez siebie listę. Lista zostaje usunięta ze wszystkich grup w jakich była dostępna oraz u osób które również ją widziały. Uzyskaliśmy to w łatwy sposób wysyłając id danej listy do odpowiedniego adresu API po stronie serwera. Tam następuje walidacja czy użytkownik wysyłający zapytanie jest jej właścicielem i czy dana lista istnieje. Jeśli wszystko przebiegło dobrze lista jest usuwana.

Otwórz listę, opcja ta jak sama nazwa wskazuje otwiera daną listę i wczytuje jej elementy. Odbywa się to dynamicznie dzięki adresowi URL, z nazwą danej listy. Informacja wysyłana jest na serwer, metoda „getItems" przeszukuje tablice „Lists" i wybiera z niej odpowiedni index oraz wysyła ją użytkownikowi.

2.4. Strona pojedynczej listy

Jest to jeden z ważniejszych widoków w naszej aplikacji, ponieważ odpowiada on za obsługę zawartości listy. Gdy klikniemy odnośnik "otwórz" na karcie z listą naszym oczom ukaże się pokazana na rysunku poniżej strona.



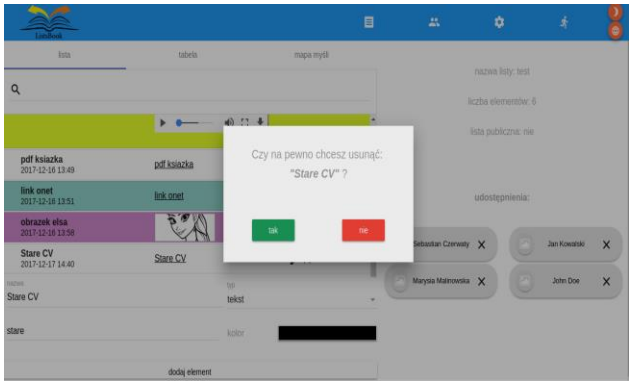
Rys. 8. *Strona widoku listy*

Znajdziemy na niej panel z zakładkami (tabs), który pozwala na zmianę sposobu wyświetlania listy. Poniżej tego panelu znajduje się wyszukiwarka pozwalająca na znalezienie elementu listy bazując na jej nazwie. Wyszukiwanie odbywa się natychmiastowo po wprowadzeniu jakiegokolwiek znaku w polu wyszukiwarki. Lista jest filtrowana wskutek czego wyświetlone zostają wszystkie elementy o wpisanej nazwie bądź elementy zawierające w swojej nazwie wpisaną frazę.

Pod wyszukiwarką widzimy właściwe elementy listy. W ukazanym na obrazku widoku "lista" mają one postać kolorowych wierszy zawierających (od lewej) - pogubioną nazwę elementu poniżej którego znajduje się data dodania elementu. Na środku znajduje się zawartość danego elementu. W zależności od typu przechowywanej wartości sposób wyświetlania jest inny.

Widok listy pozwala na usuwanie oraz edycję danego elementu za pomocą odpowiednich przycisków. Po kliknięciu w edycję element listy zamienia się w uzupełniony pierwotnymi danymi formularz zawierający te same pola co formularz dodawania nowego elementu listy.

Usuwanie odbywa się przez kliknięcie przycisku usuń element po czym należy swoją decyzję potwierdzić w ukazanym oknie modalnym.



***Rys. 9**. Okno confirm usuwania elementu listy*

2.5. Aplikacja hybrydowa na platformy mobilne

Obecnie omówimy technologie oraz przedstawimy zarys powstającej aplikacji hybrydowej na platformy mobilne.

Po omówieniu implementacji projektu przyszedł czas na planowanie jego dalszego rozwoju. Ponieważ większość ruchu sieciowego pochodzi z urządzeń mobilnych (telefony komórkowe), wybór padł na stworzenie aplikacji hybrydowej na najpopularniejsze systemu mobilne (Android i iOS). Aplikacja ta ma docelowo udostępniać te same funkcje, które dostępne są w aplikacji internetowej rozszerzając je o elementy które dostępne są w praktycznie każdym telefonie (takie jak np. aparat czy galeraia zdjęć) komórkowym z wymienionymi powyżej systemami operacyjnymi. Ponieważ logika i wygląd zostały zaprogramowane i zaplanowane przy użyciu aktualnych technologii i metod, postanowiliśmy pójść dalej i aplikację hybrydową wykonać także w tych technologiach nieco je wzbogacając.

Ionic framework

Ponieważ popularne obecnie technologie nie są wystarczające do budowy aplikacji hybrydowych zaszła potrzeba wzbogacenia ich przy jednoczesnym zachowaniu jak największej liczby z nich. Wybór padł na Ionic SDK, który bazuje na technologiach webowych a konkretniej na HTML, CSS TypeScript oraz Angularze.

Można powiedzieć, że Ionic "rozszerza" Angulara o gotowy zestaw komponentów mobilnych i styli dopasowanych do wyglądu i możliwości Androida iOS oraz Windows Phone. Jest to bardzo rozbudowane narzędzie podobnie jak Angular. Ionic oferuje CLI które pomaga w tworzeniu aplikacji a także możliwości kompilacji do gotowych plików wykonywalnych aplikacji .ipa (dla iOS) oraz .apk (dla Androida). Sam Ionic opiera się na Apache Cordova które także pozwala na tworzenie aplikacji hybrydowych, jednak nie oferuje tylu gotowych rozwiązań i komponentów a także nie wspiera z natury wszystkich technologii których użyliśmy do pisania ListsBooka (np, Angulara). Dodatkowym benefitem Ionica jest bardzo dobra dokumentacja z wieloma przykładami i możliwość wsparcia społeczności developerów jak również narzędzia takie jak Ionic viewer które pozwalają testować aplikacje bezpośrednio na naszym urządzeniu.

Naszą aplikację wolimy jednak testować już za pomocą wynikowego pliku .apk dlatego też użyliśmy narzędzi dla developerów od Google (Android Debug Bridge oraz Android SDK) aby za pomocą przeglądarki Google Chrome oraz wbudowanego w nią modułu testowania na urządzeniach mobilnych oraz kabla USB.

Projekt aplikacji hybrydowej na systemy mobilne

Jak wspomniano powyżej projekt ten zakłada stworzenie aplikacji na systemy iOS oraz Android, który będzie powielać i rozszerzać funkcje aplikacji internetowej ListsBook o elementy wbudowane w telefonach komórkowych takie jak np. kamera czy galeria zdjęć.

Aktualnie stworzony został layout (wygląd) do aplikacji hybrydowej który docelowo zostanie połączony z istniejącą już logiką napisaną w Angularze oraz korzystający z tego samego API back endowego co aplikacja internetowa. Po zainstalowaniu aplikacji hybrydowej ukaże nam się ekran logowania ze znanym z aplikacji internetowej obrazkiem w tle oraz odpowiednim formularzem. Rolę slidera z aplikacji internetowej zastąpił przesuwany ekran i przycisk "czytaj więcej".

Poniżej przedstawiono połączone (dla lepszego obrazu) elementy przesuwanego ekranu w wersji na iOS i Android. Zwróćmy uwagę na dopasowanie wyglądu i kontrolek do znanych z używanego systemu operacyjnego.

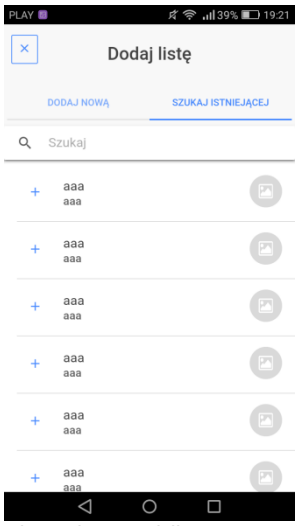
Po zalogowaniu użytkownikowi ukazują się jego listy wraz z możliwością dodania nowej po kliknięciu w przycisk "+". Każdą listę można odpowiednio modyfikować i usuwać tak jak ma to miejsce w aplikacji internetowej.

***Rys. 10.** Widok aplikacji hybrydowej*

Do nawigowania po aplikacji służy wysuwane poprzez przycisk bądź gest przesunięcia w prawo (swipe right) menu. Kliknięcie na odpowiednią pozycję powoduje przejście do wybranego widoku aplikacji. Jeśli chcemy schować menu wystarczy kliknąć przycisk bądź w przesunięcie w lewo (swipe left). Niektóre widoki można także zamykać za pomocą przycisku zamknij po lewej stronie belki nadrzędnej.

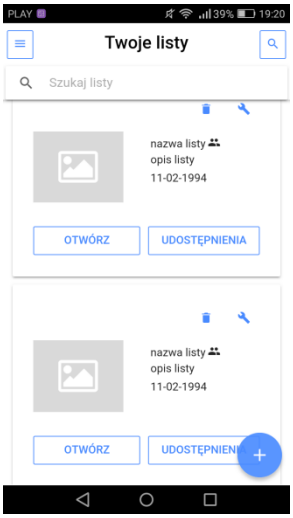
Dodawanie listy odbywa się bardzo podobnie do znanego nam z aplikacji internetowej. Należy uzupełnić formularz, wybrać kolor za

pomocą color pickera oraz wybrać z galerii bądź zrobić telefonem zdjęcie które będzie symbolizować naszą listę. Możemy także wybrać jedną z istniejących już list wybierając szukaj istniejącej.



Rys. 11. *Widok dodawania nowej listy z otwartą wyszukiwarką w aplikacji hybrydowej*

Usuwanie bądź *edycja listy* odbywa się przez kliknięcie na odpowiedni przycisk ikonę jak ma to miejsce w aplikacji internetowej.



Rys. 12. *Widok dostępnych list wraz z otwartą wyszukiwarką w aplikacji hybrydowej*

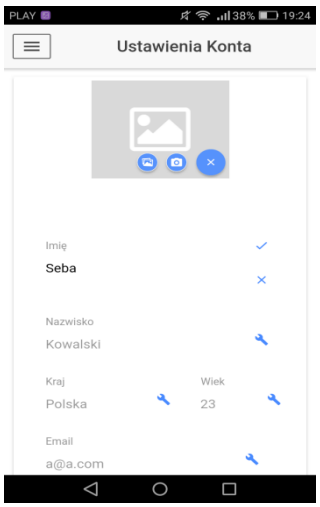
Widok znajomi i grupy pozwala na obsługę grup oraz ich znajomych. Odbywa się to w bardzo podobny sposób jak w aplikacji internetowej.

Rys. 13. *Widok znajomi i grupy*

W *widoku ustawienia*, *konta* można zmienić wprowadzając dane użytkownika. Odbywa się to tak jak w aplikacji internetowej z tą różnicą, że zdjęcie profilowe możemy wybrać z galerii bądź zrobić je aparatem

Aplikacja hybrydowa jak widać jest rozszerzeniem aplikacji internetowej. Wszystko zostało zaprojektowane tak aby jak najbardziej przypominało aplikację internetową, ponieważ zależy nam na swobodzie użytkownika. Aplikacja hybrydowa gwarantuje lepszą dostępność i niezawodność na urządzeniach mobilnych przez co wygoda i wrażenia z użytkowania są dużo wyższe.

.



Rys. 14. *Widok ustawienia konta w aplikacji hybrydowej*

PODSUMOWANIE

Celem niniejszej pracy było zaprojektowanie oraz stworzenie rozbudowanej aplikacji internetowej (ListsBook), która umożliwiałaby użytkownikom w jak najprostszy sposób magazynowanie oraz współdzielenie się danymi. Zostało to zrealizowane za pomocą zgrupowania danych w tzw. „listy" w których każda pozycja reprezentuje dane. Współdzielenie danych opiera się na udostępnianiu list z danymi użytkownikom lub samo organizującym się ich grupom. Aplikacja pozwala na magazynowanie danych w postaci: tekstów, linków, obrazów, filmów, dźwięków, plików.

W pracy zilustrowano metody planowania i współpracy pomiędzy programistami podczas tworzenia rozbudowanych aplikacji internetowych w modelu iteracyjnym. ListsBook jak każda rozbudowana aplikacja internetowa składa się z dwóch części: klienckiej (Front End) oraz serwerowej (Back End). Część kliencka została

oparta na Frameworku Angular [16] bazująca na języku TypeScript oraz standardowych technologiach internetowych: HTML oraz CSS a dokładniej na procesorze SASS [5]. Szata graficzna oraz interface zostały przygotowane i zainspirowane stylem Material Design. Za część serwerową odpowiada serwer napisany za pomocą języka JavaScript a dokładniej technologii NodeJS wraz z jego popularnymi frameworkami i rozszerzeniami takimi jak np. ExpressJS. Bazą danych jest zyskująca popularność nie relacyjna ale obiektowa baza MongoDB [14] za której obsługę i strukturyzację odpowiada biblioteka Mongoose. Dostarczanie danych realizowane jest za pomocą API typu REST. Planowana jest także ewolucja aplikacji poprzez stworzenie jej odpowiednika w postaci aplikacji hybrydowej na najpopularniejsze platformy mobilne (Android i iOS). Aplikacja hybrydowa będzie wykorzystywać istniejące API ListsBook oraz popularny framework Ionic, który oparty jest na wykorzystanych w istniejącej aplikacji internetowej technologiach Front End'owych Testowanie aplikacji internetowej odbywało się poprzez sprawdzenie jej na grupie kilu użytkowników.

Oceniana była użyteczność aplikacji, niezawodność oraz szybkość. Zdaniem testerów aplikacja uzyskała bardzo dobry rezultat w każdym z wyżej wymienionych aspektów. Projekt ten spełnił założone w procesie planowania zadania i może być zaimplementowany np. jako platforma do udostępniania plików lub informacji pomiędzy dużymi grupami ludzi.

BIBLIOGRAFIA

1. Dokumentacja środowiska Webstorm, źródło elektroniczne: https://www.jetbrains.com/webstorm/ dostęp z dnia 2.XII.2017r.
2. Inforrmacje o systemie kontroli wersji GIT, źródło elektroniczne: https://pl.wikipedia.org/wiki/Git_(oprogramowanie) dostęp z dnia 4.XII.2017
3. Przewodnik po scrumie, źródło elektroniczne: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-pl.pdf dostęp z dnia 2.XII.2017r.
4. Dokumentacja języka HTML, źródło elektroniczne: https://www.w3.org/TR/html5/introduction.html#introduction dostęp z dnia 3.XII.2017r.
5. Dokumentacja preprocesora SASS, źródło elektroniczne: http://saslang.com/documentation/file.SASS_REFERENCE.html dostęp z dnia 3.XII.2017r.
6. Statystyki popularności, źródło elektroniczne: http://githut.info/ dostęp z dnia 3.XII.2017r.
7. Crockford D.: JavaScript mocne strony, Helion, Gliwice, 2009
8. Dokumentacja języka TypeSrcipt, źródło elektroniczne: https://github.com/Microsoft/TypeScript dostęp z dnia 3.XII.2017r.
9. Przewodnik po material design, źródło elektroniczne: https://material.io/guidelines/material-design/ dostęp z dnia 3.XII.2017
10. Informacje o SSH, źródło elektroniczne: https://pl.wikipedia.org/wiki/Secure_Shell dostęp z dnia 09.XII.2017
11. Dickey J.: „Nowoczesne aplikacje internetowe. MongoDB, Express, AngularJS, NodeJS", Helion, Gliwice, 2014
12. Haviv A.: "MEAN Web Development", Packt Publishing, 2014
13. Artykuł o logowaniu przez google i facebook, źródło elektroniczne: http://www.benchmark.pl/testy_i_recenzje/logowanie-przez-google-facebook.html dostęp z dnia 18.XII.2017
14. Dokumentacja bazy danych MongoDB, źródło elektorniczne: https://docs.mongodb.com/v3.0/ dostęp z dnia 22.12.2017
15. Dokumentacja frmeworka ExpressJS, źródło elektorniczne: https://expressjs.com/en/4x/api.html dostęp z dnia 22.12.2017
16. Dokumentacja Frameworka Angular, źródło elektorniczne: https://angular.io/docs dostęp z dnia 22.12.2017

Internet application ListsBook for creating, sharing and presentation of personalized lists with data

Preparation of project and its implementation as a rich internet application with own application programming interface for storing, presentation of data stored in personalized groups of data called "lists" and also sharing this "lists" between users of this application or groups of users created by themselves.

Autorzy:

Prof. dr hab. inż. **Tadeusz Niedziela** – Uniwersytet Techniczno – Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki.

Sebastian CZEREWATY - student Uniwersytetu Techniczno – Humanistycznego w Radomiu, Wydział Informatyki i Matematyki.

Kacper DYGAS - student Uniwersytetu Techniczno – Humanistycznego w Radomiu, Wydział Informatyki i Matematyki.

JEL: L96 **DOI**: 10.24136/atest.2018.204
Data zgłoszenia: 2018.05.28 **Data akceptacji**: 2018.06.15