

Politechnika Łódzka

Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

INTERNETOWA APLIKACJA BEZPIECZNEGO PRZECHOWYWANIA DANYCH

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Promotor: dr inż. Michał Karbowańczyk

Dyplomant: Adam Zacharzewski

Nr albumu: 173245

Kierunek: Informatyka

Specjalność: Sieci Komputerowe i Systemy Teleinformatyczne

Łódź (16.03.2015)



Instytut Informatyki

90-924 Łódź, ul. Wólczajska 215, **budynek B9**

tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl

Spis treści

| | | |
|----------|--|-----------|
| 1 | Wstęp | 1 |
| 1.1 | Wprowadzenie | 1 |
| 1.1.1 | Prezentacja zagadnienia | 1 |
| 1.1.2 | Identyfikacja zagrożeń | 2 |
| 1.2 | Cel i założenia pracy | 3 |
| 1.3 | Przegląd istniejących rozwiązań | 4 |
| 1.3.1 | Kryteria oceny | 4 |
| 1.3.2 | Dropbox | 5 |
| 1.3.3 | Google Drive | 5 |
| 1.3.4 | Amazon Simple Storage | 6 |
| 1.3.5 | Microsoft OneDrive | 6 |
| 1.3.6 | Tresorit | 6 |
| 1.3.7 | SpiderOak | 7 |
| 1.3.8 | Wuala | 7 |
| 1.3.9 | Podsumowanie | 8 |
| 2 | Teoretyczne podstawy bezpieczeństwa | 9 |
| 2.1 | Wprowadzenie do kryptografii | 9 |
| 2.2 | Kryptografia asymetryczna | 10 |
| 2.3 | Podpis cyfrowy | 11 |
| 2.4 | Certyfikaty klucza publicznego | 11 |
| 2.5 | Struktura certyfikatu X.509 | 12 |
| 2.6 | Bezpieczeństwo w Internecie | 12 |
| 3 | Projekt aplikacji | 14 |
| 3.1 | Analiza wymagań | 14 |
| 3.1.1 | Wymagania funkcjonalne | 14 |
| 3.1.2 | Wymagania jakościowe | 15 |
| 3.2 | Aktorzy | 15 |
| 3.3 | Moduły funkcjonalne | 16 |
| 3.4 | Diagram przypadków użycia | 16 |
| 3.5 | Widoki dla przypadków użycia | 16 |

| | | |
|----------|---|-----------|
| 3.6 | Słownik obiektów encji | 17 |
| 3.7 | Diagram klas encji | 18 |
| 3.8 | Cykl życia pliku | 19 |
| 3.8.1 | Przesyłanie pliku | 19 |
| 3.8.2 | Pobieranie pliku | 21 |
| 3.9 | Identyfikacja słabych stron | 22 |
| 4 | Opis implementacji | 23 |
| 4.1 | Wybór podstawowych technologii i narzędzi | 23 |
| 4.2 | Konfiguracja bezpiecznego połączenia | 24 |
| 4.3 | Warstwa danych aplikacji | 25 |
| 4.3.1 | Konfiguracja połączenia z bazą danych | 25 |
| 4.3.2 | Mapowanie obiektowo-relacyjne | 25 |
| 4.4 | Warstwa logiki biznesowej | 26 |
| 4.4.1 | Tworzenie konta użytkownika | 26 |
| 4.4.2 | Uwierzytelnianie | 28 |
| 4.4.3 | Autoryzacja | 29 |
| 4.4.4 | Realizacja szyfrowania | 30 |
| 4.5 | Zachowanie poufności haseł | 30 |
| 4.6 | Warstwa prezentacji | 32 |
| 4.6.1 | Implementacja widoków | 32 |
| 4.6.2 | Silnik szablonów | 32 |
| 5 | Podsumowanie | 34 |
| 5.1 | Osiągnięty efekt | 34 |
| 5.2 | Plany rozwojowe | 35 |
| 5.3 | Wnioski z przeprowadzonych działań | 35 |
| 5.4 | Realizacja efektów kształcenia | 36 |

Rozdział 1

Wstęp

1.1 Wprowadzenie

1.1.1 Prezentacja zagadnienia

Przechowywanie danych stanowi obecnie jedno z częściej spotykanych zastosowań Internetu. Dowodem na to jest ankieta przeprowadzona w 2012 roku przez serwis imagazine.pl [1], w której 87% ankietowanych zadeklarowało korzystanie z internetowych aplikacji w celu składowania danych. Trudno się temu dziwić, gdyż liczba zalet takiego rozwiązania jest znacząca.

- Dostęp do danych może zostać zrealizowany z dowolnego miejsca na świecie, pod warunkiem posiadania urządzenia z dostępem do Internetu. Dzięki temu nie trzeba pamiętać o noszeniu przy sobie fizycznego nośnika danych, z których skorzystania może zająć konieczność w pracy, na uczelni lub w podróży.
- Nie występuje konieczność synchronizacji danych pomiędzy każdym z posiadanych urządzeń, co przekłada się na znaczną oszczędność czasu. Pliki są zawsze dostępne w najnowszej wersji - rozpoczętą pracę można kontynuować na dowolnej maszynie.
- Jest to jednocześnie sposób na wykonanie kopii zapasowej ważnych plików. W przypadku awarii twardego dysku komputera lub zgubienia pamięci przenośnej dane przechowywane w Internecie będą nadal dostępne.

Początki tego zjawiska można datować na rok 2006, w którym uruchomiona została usługa Amazon S3 (Simple Storage Service) [2]. Niedługo później powstały aplikacje Dropbox [3] oraz Google Drive [4], co spowodowało dalszy wzrost popularności internetowych dysków. Ten utrzymujący się trend można zaobserwować także

dziś. Usługodawcy oferujący internautom przestrzeń na wirtualnych dyskach prześcigają się, proponując coraz więcej gigabajtów pamięci w niższych cenach, a także udostępniając dodatkowe usługi, takie jak na przykład możliwość dzielenia się danymi z innymi osobami [5]. Zwykle użytkownik otrzymuje także niewielką ilość darmowej przestrzeni, która często okazuje się w zupełności wystarczająca [6].

1.1.2 Identyfikacja zagrożeń

Niestety, przechowywanie danych w Internecie wiąże się często z ryzykiem ich odczytania przez osoby niepowołane. Może się tak stać z wielu powodów, przy czym często wina leży po stronie samego internauty. Niedostateczna dbałość o ochronę dostępu do konta jest powszechnie spotykana, choć oczywiście istnieją też zagrożenia zupełnie niezależne od użytkownika.

- Jednym z poważniejszych naruszeń bezpieczeństwa jest wielokrotne wykorzystywanie tego samego hasła w różnych aplikacjach. Wystarczy wówczas, że odkryte zostanie hasło użytkownika do jednej z nich, a zagrożone są także wszystkie pozostałe. W taki sposób wykradziono dane użytkowników usługi Dropbox [7].
- Inną przyczyną uzyskania dostępu do konta użytkownika może być stosowanie przez niego zbyt prostych i krótkich haseł. Jak pokazuje badanie przeprowadzone przez firmę Trustwave [8], średnio w przypadku jednego włamania na trzy przyczyną okazuje się zbyt łatwe do złamania hasło.
- Utrata danych może być także skutkiem włamania przez cyberprzestępcę do komputera, na którym składowane są dane. Dlatego tak ważne jest, aby prywatne dokumenty były przechowywane w postaci zaszyfrowanej. Wówczas ich odczytanie jest niemożliwe bez posiadania dostępu do odpowiedniego klucza.
- Dane mogą także zostać ujawnione przez dostawcę usługi na wniosek organizacji rządowej [9], co nierzadko jest zaznaczone w regulaminie korzystania z aplikacji [10]. Z tego powodu system powinien uniemożliwiać uzyskanie dostępu do plików nawet jego administratorom. To wymaganie jest jednak niemożliwe do zweryfikowania, gdy szyfrowanie i deszyfrowanie danych odbywa się po stronie serwera. Nawet jeśli kod źródłowy aplikacji zostanie upubliczniony, nie sposób uzyskać pewności, że jego wersja jest identyczna z wdrożoną na serwerze. Problem ten dotyczy zdecydowanej większości istniejącego oprogramowania.

O tym, że takie naruszenie bezpieczeństwa może mieć bardzo przykre konsekwencje, nikogo nie trzeba przekonywać. Jest to szczególnie dotkliwe, gdy w grę wchodzi materiały o charakterze ściśle prywatnym [11]. Choć na lekkomyślność użytkownika niewiele można poradzić, to jednak na etapie projektowania aplikacji możliwe jest częściowe ograniczenie opisanych zagrożeń.

1.2 Cel i założenia pracy

Celem pracy jest zaprojektowanie aplikacji internetowej umożliwiającej przechowywanie danych w sposób bezpieczny. Z uwagi na to, że bezpieczeństwo jest pojęciem bardzo ogólnym, dokonano jego uszczegółowienia poprzez określenie aspektów szczególnie istotnych z punktu widzenia pracy.

- Należy zapobiec możliwości dostępu do konta użytkownika przez osobę niepowołaną nawet w przypadku złamania lub ujawnienia hasła użytkownika. W tym celu zostanie wprowadzony mechanizm uwierzytelniania dwustopniowego. Aby uzyskać dostęp do konta, użytkownik będzie musiał nie tylko wykazać się znajomością loginu oraz hasła, ale także być w posiadaniu identyfikującego go pliku, uzyskiwanego jedynie podczas rejestracji.
- Odczytanie danych powinno być niemożliwe przez osobę niebędącą ich właścicielem, nawet w przypadku uzyskania przez nią nieuprawnionego dostępu do miejsca ich przechowywania. Dane muszą więc być składowane w postaci niejawnej, a klucz pozwalający na ich odszyfrowanie powinien posiadać tylko i wyłącznie ich właściciel. Należy także rozważyć sytuację, w której plik jest współdzielony przez więcej niż jednego użytkownika.
- Transmisja danych pomiędzy komputerem użytkownika a miejscem ich przechowywania także powinna być chroniona przed ujawnieniem. Z tego względu wymagane jest zastosowanie bezpiecznego protokołu HTTPS [12], który zapewnia poufność i integralność transmisji.

Wymagania dotyczące bezpieczeństwa mają w pracy kluczowe znaczenie. Obok nich istnieją jednak inne wymogi, które także odgrywają ważną rolę. Przede wszystkim należy zapewnić możliwość komfortowej pracy z aplikacją jak najszerszemu gronu odbiorców.

- Z uwagi na to, że docelowo aplikacja ma być dostępna w Internecie, oczywistym wymaganiem jest zapewnienie możliwości korzystania z niej równocześnie

przez wielu użytkowników. Ponadto, korzystanie z całej funkcjonalności powinno być możliwe przy użyciu jedynie przeglądarki internetowej, bez potrzeby instalowania dodatkowego oprogramowania. Pozwoli to uruchomić aplikację na maksymalnej liczbie urządzeń, a przy tym nie wystąpi konieczność dostarczania osobnej wersji programu dla każdego z popularnych systemów operacyjnych.

- Projektując graficzny interfejs użytkownika należy położyć duży nacisk na jego responsywność [13], czyli dostosowywanie się do rozdzielczości ekranu na którym jest wyświetlany. Jest to ważne przede wszystkim z punktu widzenia użytkowników posiadających urządzenia mobilne, takie jak tablety i smartfony. Rozdzielczości ich ekranów są najczęściej dużo niższe niż w przypadku monitora komputerowego, co mogłoby utrudniać pracę z aplikacją, a w konsekwencji powodować irytację użytkownika. Należy mieć to na uwadze i zaprojektować aplikację w taki sposób, aby korzystanie z niej na urządzeniach przenośnych odbywało się bez problemów.

1.3 Przegląd istniejących rozwiązań

1.3.1 Kryteria oceny

W niniejszym przeglądzie przedstawione zostały najpopularniejsze na rynku rozwiązania z zakresu internetowego przechowywania danych. Aby porównanie było obiektywne, przyjęto jawne kryteria oceny, które zostały zastosowane dla każdej z aplikacji.

- Pierwsze kryterium stanowi bezpieczeństwo - ocenie poddano sposoby wykorzystane do ochrony prywatności danych użytkownika. W szczególności zwrócono uwagę na to, czy przed zapisem danych następuje ich szyfrowanie. Jeśli tak, zbadano rodzaj użytego algorytmu kryptograficznego. Uwzględniono także ochronę transmisji przed podsłuchaniem oraz możliwości odczytu plików przez administratorów aplikacji.
- Jako kolejne kryterium przyjęto wieloplatformowość aplikacji. Przede wszystkim zbadano możliwość jej uruchomienia z poziomu jedynie przeglądarki internetowej, bez potrzeby instalacji dodatkowego oprogramowania. Jeśli było to możliwe, przeprowadzono także analizę działania aplikacji na urządzeniu mobilnym.

1.3.2 Dropbox

Jedną z najpopularniejszych dostępnych na rynku aplikacji umożliwiających internetowe przechowywanie danych jest Dropbox [3]. O jej wysokiej pozycji na rynku najlepiej świadczy liczba zarejestrowanych użytkowników - jest ich ponad 300 milionów [14]. Aplikacja pozwala nie tylko przechowywać pliki, ale też współdzielić je z innymi użytkownikami. Jest także przejrzysta i intuicyjna.

Niestety, kwestie bezpieczeństwa w tej aplikacji pozostawiają wiele do życzenia. Pośród zalet można wyróżnić transmisję przy pomocy szyfrowanego protokołu HTTPS oraz możliwość uruchomienia funkcji uwierzytelniania dwustopniowego. Wówczas do skorzystania z aplikacji niezbędna jest znajomość hasła oraz kodu SMS. Stanowi to jednak niewielkie pocieszenie, gdyż dane przechowywane są na serwerze w postaci niezaszyfrowanej. Nie można więc mówić o żadnej gwarancji zachowania prywatności danych. Co więcej, firma Dropbox, Inc. jawnie oświadczyła, że nie zawaha się przed ujawnieniem danych należących do użytkowników na życzenie organizacji rządowych [9]. Świadczy to jednocześnie o tym, że właściciele firmy są w stanie odczytać pliki użytkowników.

Na korzyść aplikacji przemawia natomiast fakt, że korzystanie z niej jest możliwe przy użyciu dowolnego urządzenia posiadającego przeglądarkę internetową. Jest także bardzo dobrze przystosowana do uruchamiania na urządzeniach mobilnych, łącznie z tymi o niewielkich ekranach.

1.3.3 Google Drive

Google Drive [4] jest aplikacją pod wieloma względami podobną do opisanej powyżej. Jest to także jedna z bardziej znanych aplikacji tego typu. Na jej popularność wpływa fakt, że doskonale integruje się z pozostałymi usługami tej samej firmy, takimi jak na przykład poczta elektroniczna. Jest to znacząca zaleta, gdyż dzięki temu nie jest konieczne tworzenie kolejnego konta w innym serwisie.

Niestety, także ta aplikacja nie oferuje szyfrowania danych przechowywanych po stronie serwera. Umożliwia to dostęp do prywatnych informacji użytkownika nie tylko potencjalnym przestępcom, ale także właścicielom serwisu. Możliwe jest natomiast uwierzytelnianie dwuetapowe z wykorzystaniem wybranego przez użytkownika kanału. Zalecaną opcję stanowi dedykowana aplikacja Google Authenticator [15] służąca do generowania jednorazowych kodów, zaś alternatywą jest przesyłanie ich w wiadomości SMS. Transmisja danych następuje przy pomocy protokołu HTTPS.

Plusem aplikacji Google Drive jest natomiast niewątpliwie wieloplatformowość.

Dostęp można uzyskać z poziomu przeglądarki internetowej na dowolnym urządzeniu, przy czym także na niewielkich urządzeniach przenośnych nie przysparza to większych trudności.

1.3.4 Amazon Simple Storage

Jest to kolejna popularna aplikacja służąca do przechowywania danych [2]. Jak sama nazwa wskazuje, nacisk położony jest w niej na prostotę obsługi oraz brak zbędnych i ponadprogramowych funkcji. Aplikacja ma za zadanie realizować przede wszystkim swój główny cel, czyli przechowywać dane.

W kwestii bezpieczeństwa usługa firmy Amazon niewiele różni się od poprzednich. Tutaj także nie występuje szyfrowanie danych, co stanowi poważną wadę. Podobnie jak w opisanych powyżej, w tej aplikacji także istnieje możliwość wyboru uwierzytelniania dwuetapowego, a dostęp do danych jest możliwy przez przeglądarkę internetową.

1.3.5 Microsoft OneDrive

OneDrive [16] to produkt korporacji znanej przede wszystkim ze względu na ogromny udział w rynku systemów operacyjnych. Z aplikacji można korzystać przez przeglądarkę internetową lub pobrać wersję do zainstalowania na twardym dysku. Aplikacja OneDrive jest obecnie dostarczana wraz z systemem Windows 8.1, lecz możliwe jest także pobranie wersji dla systemów Windows 7, 8, Mac OS X oraz dla platform mobilnych, takich jak Android, Windows Phone oraz iOS. Niestety twórcy aplikacji nie przewidują możliwości jej instalacji przez użytkowników systemów operacyjnych z rodziny GNU/Linux - jedynym wyjściem jest wówczas dostęp przy użyciu przeglądarki.

Także i w tej aplikacji bezpieczeństwo nie stoi na wysokim poziomie. Dostęp do aplikacji odbywa się z wykorzystaniem protokołu HTTPS, co jest standardem w tego typu aplikacjach, jednak twórcy nie wspominają o jakimkolwiek szyfrowaniu danych. Można więc wnioskować, że dane nie są w żaden sposób chronione.

1.3.6 Tresorit

Inną aplikacją umożliwiającą przechowywanie danych jest Tresorit [17]. Jeśli chodzi o bezpieczeństwo, tej aplikacji nie można niczego zarzucić. Dane użytkownika są szyfrowane po stronie klienta algorytmem symetrycznym AES z kluczem o długości 256 bitów. Do dyspozycji jest także uwierzytelnianie dwuetapowe. Twórcy

zapewniają [18], że plików nie uda się odczytać nawet administratorom serwisu, co jest niewątpliwą zaletą.

Aplikacja nie jest jednak pozbawiona wad. Podstawowym problemem jest fakt, że do korzystania z niej nie wystarcza posiadanie przeglądarki internetowej - użytkownik musi zainstalować na swoim komputerze dodatkowe oprogramowanie. Nie stanowiłoby to samo w sobie przeszkody, gdyby twórcy zapewnili możliwość jego instalacji dla każdego systemu operacyjnego. Tak niestety nie jest - użytkownicy systemów z rodziny GNU/Linux nie będą w stanie skorzystać z aplikacji. Jest to poważny minus.

1.3.7 SpiderOak

Aplikacja SpiderOak [19] powstała jako bezpieczna alternatywa dla usługi Dropbox. Oferuje intuicyjność i przejrzystość podobną do swojego pierwowzoru, zapewniając jednak dużo wyższy poziom bezpieczeństwa. Twórcy aplikacji nie mają dostępu do przechowywanych danych, ponieważ klucz szyfrujący nie opuszcza komputera użytkownika. Dane są szyfrowane po stronie klienta algorytmem AES z kluczem o długości 256 bitów. Klucz ten tworzony jest na podstawie hasła użytkownika przy pomocy funkcji PBKDF2. Co za tym idzie, nie jest możliwe odzyskanie hasła - jego zapomnienie skutkuje nieodwracalną utratą dostępu do danych. Jest to cena, jaką trzeba zapłacić za bezpieczeństwo.

Z uwagi na to, że szyfrowanie danych odbywa się po stronie klienta, niezbędne jest pobranie i instalacja przez użytkownika specjalnego oprogramowania. Jest ono dostępne dla wielu popularnych systemów operacyjnych. Wspierane są Windows, Mac OS oraz także niektóre systemy GNU/Linux, co zdecydowanie wyróżnia aplikację spośród pozostałych. Obsługiwane są także dwa systemy przeznaczone dla urządzeń mobilnych: Android oraz iOS. Brakuje natomiast wsparcia dla systemu Windows Phone.

1.3.8 Wuala

Wuala [20] to aplikacja pod wieloma względami bardzo podobna do SpiderOak. Szyfrowanie następuje po stronie klienta, z wykorzystaniem algorytmu AES-256. Tutaj także hasło pozostaje na komputerze użytkownika, co powoduje, że w przypadku jego zapomnienia odzyskanie dostępu do danych jest niemożliwe. Kopie zapasowe danych przechowywane są na kilku serwerach znajdujących się w różnych krajach, co ma za zadanie zapobiec ich utracie w przypadku awarii jednej z maszyn.

Dostęp do aplikacji możliwy jest wyłącznie poprzez oprogramowanie instalowane na komputerze użytkownika. Podobnie jak w przypadku SpiderOak, program dostępny jest dla większości najpopularniejszych systemów operacyjnych, łącznie z niektórymi systemami GNU/Linux, co stanowi ogromną zaletę. Wspierane są także systemy mobilne: Android oraz iOS. Podobnie jak w przypadku aplikacji opisanej powyżej, twórcy nie przewidują udostępnienia wersji dla systemu Windows Phone.

1.3.9 Podsumowanie

Opisane w tym rozdziale aplikacje nie są oczywiście jedynymi dostępnymi na rynku - jest ich o wiele więcej. Jednak znalezienie pośród nich takiej, która nie posiada wad, wydaje się zadaniem niemal niemożliwym. Wiele z nich nie zapewnia wystarczającego poziomu bezpieczeństwa, lub nie oferuje go wcale. Ta kwestia dotyczy zwłaszcza najpopularniejszych rozwiązań przygotowanych przez znane korporacje takie jak Google, Amazon czy Microsoft. Inne wymagają instalowania dodatkowego oprogramowania, które często współpracuje jedynie z systemami operacyjnymi firmy Microsoft lub Apple. Dodatkowo, zdecydowana większość z nich to aplikacje komercyjne, nastawione na generowanie zysków, co pociąga za sobą zamknięty kod źródłowy. Powoduje to, że użytkownik nie jest w stanie samodzielnie przeprowadzić analizy bezpieczeństwa aplikacji i jest zmuszony do zaufania w tej kwestii producentowi.

Rozdział 2

Teoretyczne podstawy bezpieczeństwa

2.1 Wprowadzenie do kryptografii

Większość stosowanych obecnie form komunikacji wiąże się z możliwością przechwycenia treści komunikatu przez osoby niepowołane. Z tego powodu niezbędne stało się wynalezienie sposobu na przekazywanie informacji w sposób zrozumiały jedynie dla nadawcy oraz odbiorcy. Potrzeba ta była szczególnie paląca w zastosowaniach militarnych - przechwycenie strategicznej wiadomości przez wroga mogłoby skutkować porażką całej operacji wojskowej. W efekcie to właśnie wojskowość stała się motorem napędzającym rozwój kryptografii - dziedziny wiedzy, zajmującej się zabezpieczaniem informacji przed niepowołanym dostępem. Obecnie kryptografia jest wszechobecna, przy czym najłatwiej można zaobserwować jej istnienie w Internecie. Wykorzystywana jest w bankowości internetowej, handlu elektronicznym oraz w każdym serwisie, w którym użytkownik uwierzytelnia się przy pomocy hasła [21].

Algorytmy kryptograficzne opierają się na przekształcaniu komunikatu w formie jawnej na wiadomość zaszyfrowaną, czyli szyfrogram. Do przekształcenia niezbędny jest także ciąg znaków zwany kluczem, który najczęściej jest znacznie krótszy od szyfrowanej wiadomości. Ten sam klucz może być później wykorzystany do odszyfrowania komunikatu - mowa wówczas o algorytmie symetrycznym [22]. Jednym z popularnych algorytmów tego typu jest AES [23]. Jest on wykorzystywany między innymi przez rząd Stanów Zjednoczonych Ameryki do ochrony ściśle tajnych informacji [24]. Stosowanie klucza sprawia, że algorytm może być całkowicie jawny bez szkody dla poufności transmisji. Można więc wykorzystywać wielokrotnie ten

sam algorytm, zmieniając jedynie klucz. Dodatkowo, jawność algorytmu szyfrowania ma bardzo ważną zaletę. Dzięki temu wiele osób może dokonywać jego analizy w poszukiwaniu potencjalnie słabych punktów. Jeśli nikomu nie uda się go złamać, oznacza to, że jest naprawdę bezpieczny [25].

Algorytmy symetryczne mają jednak jedną podstawową wadę. Związana jest ona z koniecznością przekazania klucza drugiej stronie jeszcze przed rozpoczęciem komunikacji. Przekazanie klucza musi zaś nastąpić przy użyciu kanału, którego bezpieczeństwo jest potwierdzone. W przeciwnym wypadku klucz mógłby zostać przechwycony, a co za tym idzie komunikacja prowadzona z jego użyciem nie byłaby bezpieczna.

2.2 Kryptografia asymetryczna

W odpowiedzi na problem z dystrybucją kluczy zaczęła rozwijać się nowa grupa algorytmów kryptograficznych - algorytmy z kluczem publicznym [26]. Opierają się one na wykorzystaniu nie jednego, ale dwóch kluczy, przy czym znajomość żadnego z nich nie pozwala w prosty sposób wyznaczyć drugiego. Jeden z kluczy jest jawny i udostępniany wszystkim zainteresowanym - dlatego nazywany jest kluczem publicznym. Przy jego pomocy można dokonać zaszyfrowania komunikatu. Odszyfrowanie jest natomiast możliwe tylko przy użyciu drugiego z kluczy - nazywanego prywatnym - który jest znany jedynie odbiorcy wiadomości. Wiadomość może więc być wysłana przez dowolną osobę, natomiast odebrana jedynie przez tą, do której jest skierowana. Wymaga to posiadania przez każdą ze stron komunikacji własnej pary kluczy - publicznego i prywatnego.

Algorytmy asymetryczne posiadają bardzo wiele zastosowań, które opisane zostały w dalszej części pracy. Niestety, mają także jedną podstawową wadę - ich prędkość działania pozostawia wiele do życzenia. Z tego względu często w połączeniu z asymetrycznym wykorzystuje się algorytm symetryczny. Zaszyfrowanie wiadomości następuje przy użyciu algorytmu symetrycznego z dowolnie wybranym kluczem, który jest następnie szyfrowany asymetrycznie z użyciem klucza publicznego odbiorcy. Tak utworzona wiadomość jest przekazywana razem z utajnionym kluczem. Odbiorca odczytuje najpierw klucz symetryczny korzystając ze swojego klucza prywatnego, dzięki czemu jest w stanie poznać treść komunikatu. Takie wykorzystanie kryptografii nazywane jest algorytmem hybrydowym [27].

2.3 Podpis cyfrowy

Zastosowania algorytmów asymetrycznych nie kończą się jednak na szyfrowaniu transmisji. Umożliwiają one także zagwarantowanie autentyczności dokumentu lub komunikatu oraz zabezpieczenie go przed zmianą dokonaną przez osobę nieuprawnioną. Taka procedura nosi nazwę złożenia podpisu cyfrowego. Polega ona na wyznaczeniu sumy kontrolnej dla danej wiadomości przy pomocy wybranej funkcji skrótu (może to być na przykład MD5 [28] lub SHA-1 [29]), a następnie zaszyfrowanie wyliczonej wartości przy użyciu klucza prywatnego osoby składającej podpis. Tak zaszyfrowana suma jest dołączona do oryginalnego dokumentu. Gdy zajdzie potrzeba weryfikacji jego autentyczności, przy pomocy klucza publicznego suma kontrolna jest odszyfrowywana, a następnie porównywana z wartością wyznaczoną ponownie dla tego dokumentu, z użyciem tej samej funkcji skrótu. Jeśli wartości się zgadzają, oznacza to, że od momentu podpisania dokument nie został zmodyfikowany.

Dodatkowym atutem podpisu cyfrowego jest niezaprzeczalność. Nadawca nie jest w stanie wyprzeć się wysłania wiadomości, która została podpisana przy użyciu jego klucza prywatnego. Jego tożsamość może być bardzo łatwo potwierdzona - wystarczy skorzystać z dostępnego publicznie klucza i zweryfikować podpis. Problem może tutaj natomiast stanowić sytuacja, w której klucz zostaje zmieniony lub skradziony. Wówczas rozwiązanie sytuacji zależy od obowiązującego prawa [30].

2.4 Certyfikaty klucza publicznego

Z kryptografią asymetryczną związany jest kolejny problem - każda z komunikujących się stron musi najpierw poznać klucz publiczny partnera. Klucze te z założenia są powszechnie dostępne, mogą być na przykład umieszczone w Internecie. Poznanie ich nie powinno więc stanowić problemu. Jednak w przypadku jego podmiany przez osobę o złych zamiarach, cała podsłuchana komunikacja stałaby się prosta do odszyfrowania. Umożliwiłoby to nie tylko poznanie treści przesyłanych wiadomości, ale także zastępowanie ich własnymi bez wzbudzania jakichkolwiek podejrzeń. Zatem w jaki sposób można stwierdzić, czy klucz publiczny rzeczywiście należy do danej osoby?

Rozwiązaniem jest certyfikat klucza publicznego, który wiąże klucz z tożsamością jego posiadacza. Certyfikat może na przykład potwierdzać wiarygodność domeny internetowej. Autentyczność certyfikatu jest zaś gwarantowana przez pod-

pis cyfrowy zaufanej instytucji. Jeden z najpopularniejszych systemów certyfikatów, o nazwie X.509 [31], oparty jest o strukturę hierarchiczną. Na szczycie stoi główny urząd certyfikacji, który wystawia certyfikaty dla urzędów podrzędnych. One z kolei także mogą certyfikować własne urzędy podrzędne. Tak powstaje „łańcuch certyfikacji”, na którego końcu może znajdować się na przykład certyfikat domeny banku lub sklepu internetowego [32].

2.5 Struktura certyfikatu X.509

Certyfikat X.509 zawiera następujące informacje:

- wersja X.509 - obecnie najczęściej stosowana jest wersja trzecia,
- numer seryjny, wygenerowany dla danego certyfikatu,
- rodzaj algorytmu wykorzystanego w procesie podpisywania certyfikatu,
- nazwa wystawcy certyfikatu,
- określenie terminu ważności certyfikatu,
- nazwa podmiotu, dla którego certyfikat jest wystawiany,
- klucz publiczny podmiotu wraz z identyfikatorem algorytmu,
- podpis cyfrowy wystawcy wraz z identyfikatorem algorytmu.

2.6 Bezpieczeństwo w Internecie

Wprowadzenie certyfikatów klucza publicznego umożliwiło przejście sieci WWW na zupełnie inny poziom, jeśli chodzi o kwestie prywatności transmisji. Standardowy protokół HTTP, wykorzystywany powszechnie w komunikacji między przeglądarką internetową użytkownika a serwerem, na którym znajduje się strona internetowa, pozbawiony jest jakichkolwiek zabezpieczeń. Cała transmisja odbywa się w sposób jawny, a zatem możliwe jest jej przechwycenie oraz modyfikowanie przez osoby niepowołane. Wykonywanie elektronicznych transakcji bankowych przy użyciu tego protokołu stanowiłoby ogromne ryzyko.

Aby zaradzić temu problemowi, wprowadzony został protokół SSL (Secure Socket Layer), a następnie jego rozwinięcie TLS (Transport Layer Security). Stanowi on dodatkową warstwę bezpieczeństwa na której oparty jest HTTP, określany

wówczas mianem HTTPS. Najczęściej komunikacja odbywa się wtedy poprzez port 443, dla odróżnienia od zwykłego HTTP dostępnego przez port 80. Zadaniem warstwy SSL jest przede wszystkim zapewnienie nieustannego szyfrowania transmisji. Szyfrowanie odbywa się przy pomocy algorytmu symetrycznego, z wykorzystaniem ustalonego klucza sesji. Jednak aby w bezpieczny sposób ustalić ten klucz, najpierw wykorzystywany jest algorytm asymetryczny. W procesie zestawiania połączenia serwer przesyła do klienta swój klucz publiczny oraz ewentualnie certyfikat potwierdzający jego autentyczność. Klient generuje wstępny klucz sesji, szyfruje go otrzymanym kluczem publicznym i przesyła do serwera. Następnie na podstawie klucza wstępnego oraz przesłanych wcześniej liczb losowych ustalany jest właściwy klucz sesji, którym od tej pory szyfrowana jest transmisja [33].

Rozdział 3

Projekt aplikacji

3.1 Analiza wymagań

3.1.1 Wymagania funkcjonalne

Projekt rozpoczęty został od określenia wymagań, jakie ma spełniać aplikacja. W pierwszej kolejności wyodrębnione zostały wymagania funkcjonalne. Opisują one czynności, które użytkownik może wykonać podczas pracy z aplikacją.

- Użytkownik który nie posiada konta w aplikacji musi mieć możliwość jego utworzenia. W przeciwnym razie nie będzie miał dostępu do pozostałej funkcjonalności oferowanej przez aplikację.
- Użytkownik posiadający konto może uwierzytelnić się w aplikacji, co pozwoli mu uzyskać dostęp do całej dostępnej funkcjonalności.
- Użytkownik ma możliwość utworzenia jednego lub większej liczby katalogów nadrzędnych, w których przechowywane będą inne katalogi oraz pliki. Bez utworzenia co najmniej jednego katalogu nadrzędnego nie będzie możliwe przesyłanie plików.
- Użytkownik ma możliwość przesłania wybranego pliku na serwer. Plik zostanie zaszyfrowany, a następnie umieszczony w wybranym wcześniej katalogu.
- Użytkownik może pobrać wybrany plik z serwera. Plik zostanie zapisany na dysku twardym użytkownika w postaci jawnej.
- Użytkownik może nieodwracalnie usunąć wybrany plik lub katalog z całą zawartością.

- Użytkownik może w prosty sposób przeglądać posiadane przez niego katalogi i pliki, przy czym w każdej chwili zna swoją aktualną pozycję w hierarchii katalogów.

3.1.2 Wymagania jakościowe

Dodatkowo wyróżnione zostały wymagania jakościowe, mające na celu poprawę łatwości użytkowania aplikacji. Wymagania te skierowane są przede wszystkim na zwiększenie satysfakcji użytkownika.

- Aplikacja powinna być dostępna w języku angielskim, a dodanie kolejnych języków powinno być możliwe w jak najłatwiejszy sposób.
- Aplikacja powinna działać poprawnie z wykorzystaniem najpopularniejszych przeglądarek internetowych w najnowszych dostępnych wersjach - wspierane powinny być co najmniej Internet Explorer, Mozilla Firefox, Opera, Google Chrome, Safari.
- Aplikacja powinna wyświetlać się poprawnie niezależnie od rozdzielczości ekranu urządzenia.

3.2 Aktorzy

Po przeanalizowaniu wymagań, dokonano wyodrębnienia aktorów mogących wchodzić w interakcję z aplikacją.

- Użytkownik anonimowy - jest to podmiot który nie posiada konta lub jeszcze nie dokonał uwierzytelnienia w aplikacji. Jedyne dostępne dla niego akcje to utworzenie konta oraz uwierzytelnienie. Zostanie wprowadzony mechanizm zabezpieczający przed dostępem anonimowego użytkownika do pozostałych funkcji systemu.
- Użytkownik uwierzytelniony, który ma możliwość korzystania ze wszystkich funkcji aplikacji związanych z zarządzaniem katalogami i plikami. Dostępne dla niego akcje opisane są jako wymagania funkcjonalne.

W miarę rozwoju aplikacji możliwe jest dodanie kolejnych aktorów. W takim przypadku nastąpi podział roli użytkownika uwierzytelnionego na kilka podgrup, w zależności od poziomu uprawnień w systemie.

3.3 Moduły funkcjonalne

Następnie dokonano rozdzielenia funkcjonalności oferowanej w aplikacji na dwa niezależne od siebie moduły:

- Moduł USR - obejmuje funkcje związane bezpośrednio z kontem użytkownika w aplikacji, takie jak rejestracja i uwierzytelnienie.
- Moduł RES - dotyczy zarządzania zasobami posiadanymi przez użytkownika, czyli plikami i katalogami. Moduł będzie obejmował między innymi ich przeglądanie, dodawanie oraz usuwanie.

W miarę rozwoju aplikacji możliwe jest dodanie kolejnych modułów funkcjonalnych, co w żaden sposób nie wpłynie na pracę dotychczas istniejących modułów.

3.4 Diagram przypadków użycia

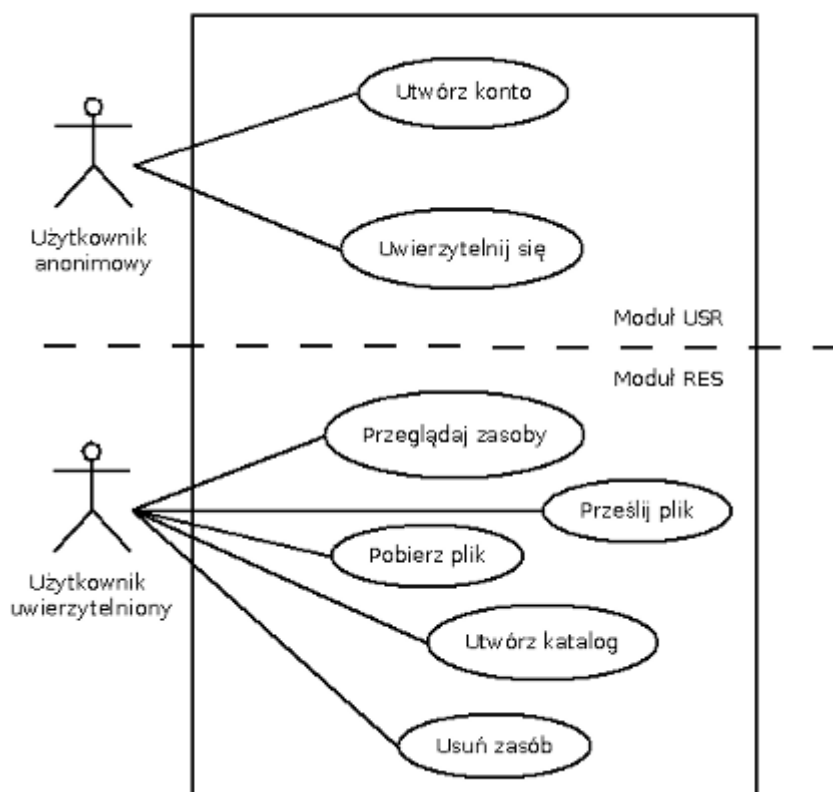
Na podstawie przeprowadzonej do tej pory analizy sporządzono diagram przypadków użycia, który widoczny jest na rys. 3.1. Przedstawia on wszystkie czynności, które użytkownik może wykonać w aplikacji w zależności od jego roli. Jest to jednocześnie punkt wyjścia do dalszych etapów projektowania.

Dodatkowo diagram został podzielony na dwie części, odpowiadające modułom funkcjonalnym aplikacji. Pozwala to lepiej zobrazować funkcjonalność dostępną w ramach każdego z modułów.

3.5 Widoki dla przypadków użycia

Na podstawie diagramu przypadków użycia dokonano wyodrębnienia ekranów aplikacji, które będą służyły do realizacji poszczególnych przypadków użycia. W skład warstwy widoku wchodzić będą następujące podstrony:

- strona główna aplikacji, na której wyświetlone będą odnośniki do logowania i rejestracji,
- widok rejestracji z formularzem do wprowadzenia danych,
- widok uwierzytelnienia z formularzem zawierającym pola do wprowadzenia nazwy użytkownika i hasła,
- widok zasobów, zawierający listę katalogów i plików użytkownika,



Rysunek 3.1: Diagram przypadków użycia

- widok tworzenia nowego katalogu,
- widok przesyłania pliku,
- widok z ostrzeżeniem przed usunięciem pliku lub katalogu.

Dodatkowo, na każdej podstronie widoczny będzie wspólny element w postaci paska nawigacyjnego, znajdującego się w górnej części ekranu. Będą się na nim znajdować między innymi odnośnik do strony głównej, tożsamość aktualnie zalogowanego użytkownika oraz przycisk wylogowania z aplikacji.

Widoki dostępne tylko dla uwierzytelnionego użytkownika zostaną zgrupowane w jednym katalogu w celu umożliwienia łatwiejszej kontroli dostępu.

3.6 Słownik obiektów encji

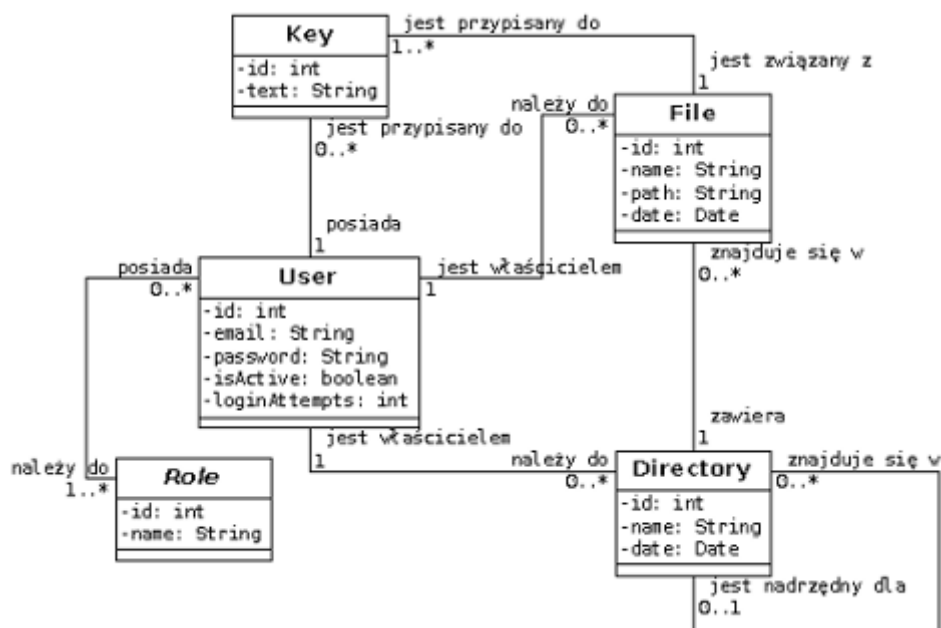
W kolejnym etapie dokonano określenia encji występujących w aplikacji. Wyodrębniono następujące obiekty:

- User - jest to obiekt reprezentujący konto użytkownika w aplikacji. Jego atrybuty to między innymi adres e-mail (tożsamy z loginem), skrócona postać hasła, wartość logiczna służąca do potencjalnego zablokowania dostępu do aplikacji oraz wartość określająca dotychczasową liczbę nieudanych prób uwierzytelnienia.
- Role - obiekt reprezentujący rolę użytkownika w aplikacji. Służy do weryfikowania uprawnień użytkownika do korzystania z poszczególnych funkcji aplikacji. Na chwilę obecną przewidziana jest tylko jedna rola - użytkownik uwierzytelniony. Wraz z rozwojem aplikacji możliwe jest jednak dodanie kolejnych ról.
- Directory - obiekt reprezentujący katalog utworzony przez użytkownika. Atrybuty katalogu to nazwa oraz data utworzenia. Katalog może zawierać w sobie inne katalogi oraz pliki. Katalog posiada też właściciela - jest to użytkownik, który go utworzył. Dodatkowo katalog posiada też referencję do swojego katalogu nadrzędnego. Jeśli takiego nie ma, oznacza to, że katalog znajduje się na szczycie hierarchii. Referencja do katalogu nadrzędnego jest wówczas pusta.
- File - obiekt reprezentujący plik umieszczony przez użytkownika na serwerze. Podobnie jak w przypadku katalogu, plik posiada nazwę, datę utworzenia oraz właściciela. Dodatkowo, dla pliku obowiązkowe jest posiadanie katalogu nadrzędnego.
- Key - obiekt reprezentujący klucz, który umożliwia szyfrowanie i odszyfrowanie pliku algorytmem symetrycznym. Każdy klucz przypisany jest jednocześnie do użytkownika oraz do pliku. Jest on przechowywany w postaci zaszyfrowanej kluczem publicznym użytkownika. W przypadku, gdy wielu użytkowników współdzieli dany plik, każdy z nich posiada własny egzemplarz klucza, zaszyfrowanego własnym kluczem publicznym.

3.7 Diagram klas encji

Relacje między encjami przedstawione zostały na diagramie klas encji. Jest on widoczny na rys. 3.2. Diagram ten przekłada się bezpośrednio na implementację warstwy modelu danych aplikacji.

Przedstawiony powyżej schemat wraz ze słownikiem obiektów encji reprezentuje jednocześnie jeden z możliwych sposobów rozwiązania postawionego problemu. Został on wybrany z następujących powodów:



Rysunek 3.2: Diagram klas encji.

- umożliwia współdzielenie plików między wieloma użytkownikami bez konieczności przechowywania wielu kopii tego samego pliku - wystarczy utworzyć dla danego pliku kolejny obiekt klucza,
- szyfrowanie oraz deszyfrowanie pliku odbywa się przy pomocy algorytmu symetrycznego, co zapewnia odpowiednią wydajność.

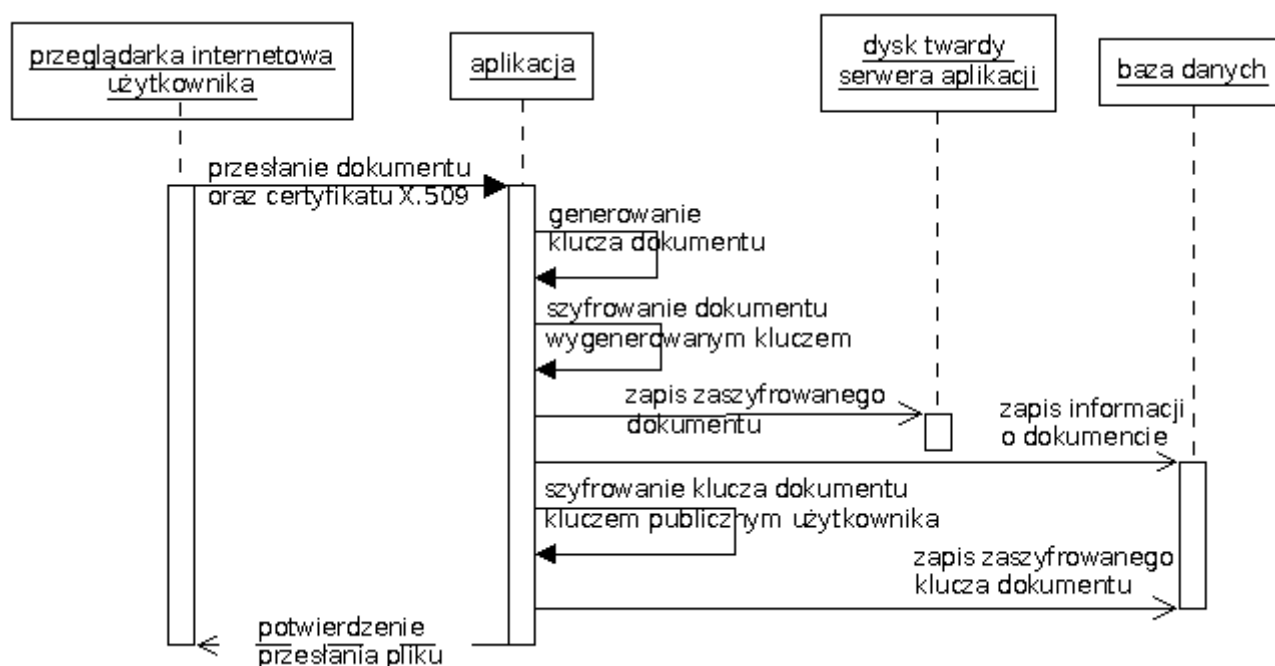
3.8 Cykl życia pliku

Najważniejsze funkcje aplikacji to przesyłanie oraz pobieranie plików wraz z ich szyfrowaniem oraz deszyfrowaniem. Z tego względu istotne staje się szczegółowe zaprojektowanie przebiegu tych procesów.

3.8.1 Przesyłanie pliku

Proces przesyłania pliku przedstawiony został na diagramie sekwencji, widocznym na rys. 3.3.

- Użytkownik znajduje się na ekranie realizującym przypadek użycia przesyłania pliku.
- Użytkownik wskazuje lokalizację wybranego pliku na dysku twardym swojego komputera.



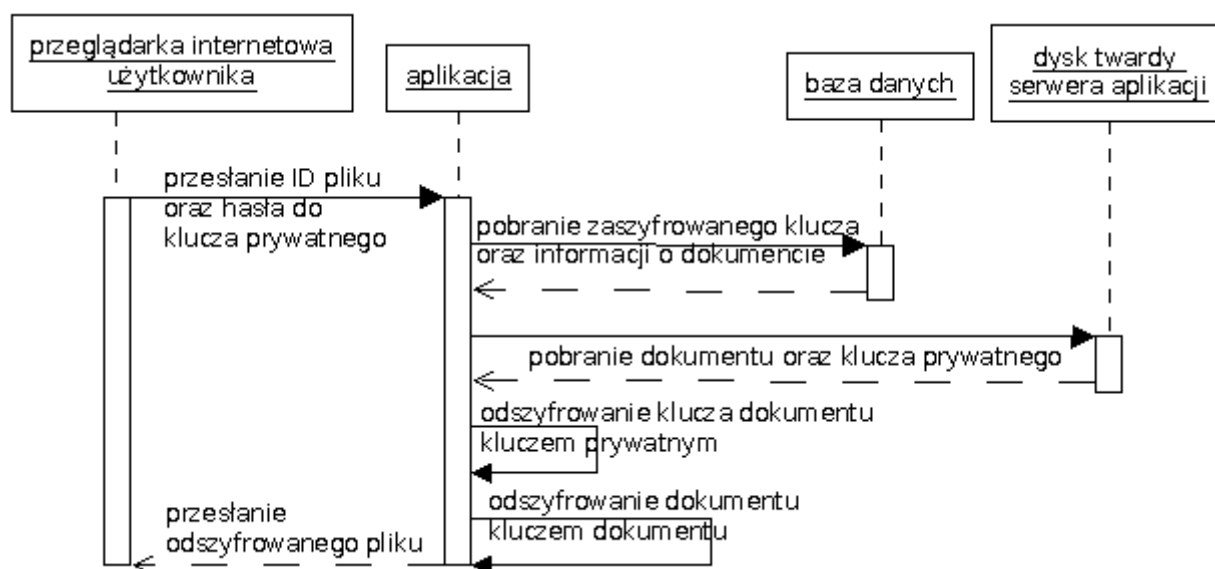
Rysunek 3.3: Diagram sekwencji przesyłania pliku

- Po zatwierdzeniu wyboru przez użytkownika następuje przesłanie pliku do serwera z wykorzystaniem metody POST. Dane transmitowane są bezpiecznym protokołem HTTPS.
- Po stronie serwera generowany jest losowo klucz o zadanej długości.
- Przy użyciu wygenerowanego klucza plik zostaje zaszyfrowany wybranym algorytmem symetrycznym.
- Klucz symetryczny zostaje zaszyfrowany algorytmem asymetrycznym przy użyciu klucza publicznego użytkownika. Klucz publiczny pochodzi z certyfikatu X.509, przy pomocy którego użytkownik uwierzytelnia się w aplikacji.
- Tak zaszyfrowany klucz pliku zostaje umieszczony w bazie danych wraz z informacją o pliku i użytkowniku których dotyczy.
- Plik zostaje zapisany na dysku twardym serwera, a informacja o ścieżce, pod którą się znajduje, trafia do bazy danych.

Podczas całej procedury, plik ani klucz w postaci jawnej nie są przechowywane na dysku twardym serwera, a jedynie w pamięci operacyjnej. Zapobiega to powstawaniu śladów, które mogłyby zostać odnalezione przez potencjalnego agresora.

3.8.2 Pobieranie pliku

Proces pobierania pliku przedstawiony został na diagramie sekwencji, widocznym na rys. 3.4.



Rysunek 3.4: Diagram sekwencji pobierania pliku

- Użytkownik wykonuje w aplikacji akcję pobrania dla wybranego pliku.
- Użytkownik zostaje poproszony o wprowadzenie hasła do kontenera, w którym znajduje się klucz prywatny.
- Hasło przesyłane jest do serwera przy pomocy protokołu HTTPS, co pozwala uzyskać dostęp do klucza prywatnego użytkownika.
- W bazie danych wyszukiwany jest zaszyfrowany klucz symetryczny pliku dla danego użytkownika.
- Klucz symetryczny pobrany z bazy danych zostaje odszyfrowany z użyciem klucza prywatnego użytkownika.
- Przy użyciu klucza symetrycznego następuje odszyfrowanie pliku.
- Odszyfrowany plik zostaje pobrany na dysk twardy użytkownika przy użyciu bezpiecznego protokołu HTTPS.

3.9 Identyfikacja słabych stron

Zaprojektowane rozwiązanie nie jest idealne z punktu widzenia bezpieczeństwa. Istnieje w nim kilka zagrożeń, które należy wziąć pod uwagę.

- Podstawowym problemem jest przechowywanie klucza prywatnego użytkownika po stronie serwera. W idealnej sytuacji klucz nie powinien opuszczać komputera użytkownika. Jednak z uwagi na to, że projektowana aplikacja działa w przeglądarce internetowej, takie rozwiązanie jest niemożliwe. Wymagałoby ono zaprojektowania odrębnej aplikacji realizującej deszyfrowanie pliku po stronie klienta, co stoi w sprzeczności z założeniami pracy. W tym przypadku niezbędny jest więc pewien kompromis.
- Innym zagrożeniem jest możliwość przeniesienia wrażliwych danych (plik lub klucz do niego w postaci jawnej) z pamięci operacyjnej serwera do przestrzeni wymiany. Dane znajdujące się na dysku mogą zaś stanowić lukę w bezpieczeństwie w przypadku włamania. Jednym z możliwych rozwiązań jest regularne czyszczenie przestrzeni wymiany i nadpisywanie jej losowymi danymi, co utrudni odczytanie uprzednio znajdujących się tam informacji. Innym, bardziej radykalnym wyjściem jest zapewnienie odpowiedniej ilości pamięci RAM oraz odmontowanie partycji wymiany. Należy jednak wcześniej zweryfikować, czy nie wpłynie to na stabilność serwera.
- Wrażliwe dane mogą też zostać odczytane bezpośrednio z pamięci operacyjnej serwera. Aby zapobiec takiemu atakowi, należy nieustannie dbać o jego bezpieczeństwo. Niezbędne jest wprowadzenie daleko idących restrykcji fizycznego oraz zdalnego dostępu do serwera przez osoby do tego nieuprawnione, a także wdrożenie metod ochrony przed szkodliwym oprogramowaniem.

Rozdział 4

Opis implementacji

4.1 Wybór podstawowych technologii i narzędzi

Pierwszym etapem implementacji aplikacji był wybór technologii, narzędzi oraz języka programowania. Do realizacji projektu wybrany został język Java, a jako szkielet aplikacji posłużył Spring Framework [34]. Jest to jeden z bardziej znanych i często wykorzystywanych szkieletów aplikacji internetowych. W jego skład wchodzi wiele modułów, które ułatwiają tworzenie aplikacji zmniejszając wymagany nakład pracy programisty.

- Kontener IoC [35] pozwala na tworzenie zależności między komponentami programu, w czasie jego działania, przez zewnętrzny proces. Mechanizm ten nazywany jest „wstrzykiwaniem zależności”.
- Szablon Model-Widok-Kontroler [36] pozwala na podział aplikacji z graficznym interfejsem użytkownika na trzy części, które współdziałają ze sobą.
- Szablon zarządzania transakcjami pozwala na łatwe definiowanie granic transakcji bazodanowych oraz ich wycofywanie w przypadku wystąpienia błędu.
- Szablon dostępu do danych ułatwia integrację aplikacji z frameworkami zapewniającymi mapowanie obiektowo-relacyjne, jak na przykład Hibernate [37].
- Szablon uwierzytelniania i autoryzacji dostarcza sposobów na potwierdzenie tożsamości użytkownika oraz kontroli jego dostępu do poszczególnych zasobów.

W kwestii środowiska programistycznego wybór padł na Eclipse - jest to wieloplatformowe środowisko udostępniane jako oprogramowanie open source. Oprócz standardowych funkcji takich jak kolorowanie składni czy automatyczne uzupełnianie kodu, Eclipse zapewnia wiele mechanizmów wspierających tworzenie aplikacji

```
1 <Connector port="8080" redirectPort="8443" />
2 <Connector
3     scheme="https" secure="true" SSLEnabled="true"
4     sslProtocol="TLS" SSLEngine="on" port="8443"
5     SSLVerifyClient="require" clientAuth="want"
6     keystoreFile="/opt/tomcat/ca/serverKeyStore.jks"
7     keystoreType="JKS" keystorePass="lGz2n9J1Uz9pe50f"
8     truststoreFile="/opt/tomcat/ca/serverKeyStore.jks"
9     truststoreType="JKS" truststorePass="lGz2n9J1Uz9pe50f"
10 />
```

Listing 4.1: Konfiguracja SSL na serwerze Apache Tomcat

internetowych. Dzięki temu możliwe jest na przykład zintegrowanie środowiska z serwerem aplikacyjnym, co pozwala śledzić aktualnie wykonywany kod w poszukiwaniu błędów. Inną z zalet jest system wtyczek, który pozwala w prosty sposób rozszerzać funkcjonalność środowiska. Przykładowo, jedna z wykorzystanych w projekcie wtyczek - o nazwie Checkstyle - zapewnia nadzór nad estetyką kodu źródłowego, co ułatwia jego późniejsze utrzymanie.

4.2 Konfiguracja bezpiecznego połączenia

Aby zabezpieczyć komunikację między klientem a serwerem, dokonano konfiguracji połączenia z wykorzystaniem szyfrowanego protokołu HTTPS. Konfiguracja odbywa się po stronie serwera Apache Tomcat w pliku `server.xml`, którego fragment przedstawiono na listingu 4.1.

Dostęp do aplikacji przy pomocy protokołu szyfrowanego odbywa się poprzez port 8443. Dodatkowo wprowadzono przekierowanie z portu 8080 na port 8443, gdzie szyfrowanie jest zagwarantowane. W konfiguracji uwzględniono także dane dostępowe do kontenera w którym znajdują się klucz publiczny oraz prywatny serwera. Jest to niezbędne w procesie nawiązywania bezpiecznego połączenia przy pomocy protokołu SSL.

```
1 DriverManagerDataSource dataSource
2     = new DriverManagerDataSource();
3 dataSource.setDriverClassName(driver);
4 dataSource.setUrl(url);
5 dataSource.setUsername(username);
6 dataSource.setPassword(password);
```

Listing 4.2: Konfiguracja połączenia z bazą danych w klasie DatabaseConfig

```
1 dataSource.driverClassName=org.h2.Driver
2 dataSource.url=jdbc:h2:tcp://localhost/~strongbox
3 dataSource.username=user
4 dataSource.password=agloawb80b
```

Listing 4.3: Fragment pliku konfiguracyjnego settings.properties

4.3 Warstwa danych aplikacji

4.3.1 Konfiguracja połączenia z bazą danych

Za konfigurację połączenia z bazą danych odpowiada klasa DatabaseConfig. Następuje w niej między innymi utworzenie obiektu klasy DriverManagerDataSource, który bezpośrednio odpowiada za przechowywanie danych dostępnych do bazy. Proces ten został przedstawiony na listingu 4.2.

Dane wykorzystane do inicjalizacji źródła danych zostały przeniesione do pliku settings.properties. Z tego pliku następuje ich wstrzyknięcie do klasy DatabaseConfig. Fragment pliku settings.properties przedstawiono na listingu 4.3.

Takie rozwiązanie pozwala uniknąć konieczności rekompilacji kodu źródłowego aplikacji w przypadku zmiany danych konfiguracyjnych. Jego dodatkową zaletą jest to, że konfiguracja aplikacji zostaje zgromadzona w jednym pliku, co pozwala ją łatwiej zarządzać.

4.3.2 Mapowanie obiektowo-relacyjne

Odwzorowywanie informacji zapisanych w relacyjnej bazie danych na struktury obiektowe następuje z użyciem adnotacji zdefiniowanych w standardzie Java Persistence Api [38]. Szczególnie ważne są tutaj adnotacje @OneToMany oraz @ManyToMany które pozwalają na odzwierciedlenie wiązania encji relacjami z wykorzystaniem klucza obcego.

Interakcja z bazą danych odbywa się przy pomocy metod udostępnianych przez

```
1 String sql = "SELECT a FROM Directory a
2   WHERE a.user.id = :id AND a.directory = null";
3 Query query = entityManager.createQuery(sql, Directory.class);
4 query.setParameter("id", id);
5 return (List<Directory>) query.getResultList();
```

Listing 4.4: Przykład użycia metody `createQuery` klasy `EntityManager`

klasę `EntityManager`, takich jak `find`, `persist` czy `remove`. Są one najzupełniej wystarczające w typowych przypadkach odczytu i zapisu danych. Zdarzają się jednak sytuacje, w których zachodzi konieczność posłużenia się bardziej skomplikowanym zapytaniem SQL. Wówczas z pomocą przychodzi metoda `createQuery`, której przykładowe zastosowanie przedstawione jest na listingu 4.4.

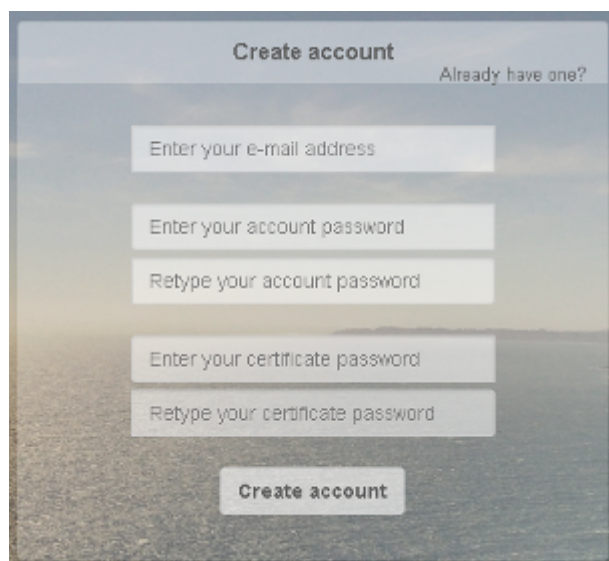
Powyższy listing prezentuje jednocześnie wykorzystanie języka Java Persistence Query Language [39] w celu utworzenia dynamicznego zapytania SQL. Zapytanie to przyjmuje jako parametr liczbę oznaczającą numer identyfikacyjny użytkownika. Takie parametryzowanie kwerend SQL stanowi zabezpieczenie przed atakiem znanym jako „SQL Injection”.

4.4 Warstwa logiki biznesowej

4.4.1 Tworzenie konta użytkownika

Aby uzyskać dostęp do funkcjonalności oferowanej przez aplikację, użytkownik musi utworzyć konto wybierając swój login oraz dwa hasła, co zostało zaprezentowane na rys. 4.1. Jedno z nich wykorzystywane jest podczas uwierzytelnienia użytkownika w aplikacji - jest to hasło do konta. Drugie służy do zabezpieczenia klucza prywatnego, który zostaje utworzony podczas rejestracji i jest przechowywany na serwerze. Za każdym razem, gdy zachodzi potrzeba skorzystania z tego klucza podczas odszyfrowywania dokumentu, wymagane jest wprowadzenie hasła. Pozwala to zapobiec wykorzystaniu klucza przez osobę, która bezprawnie weszła w jego posiadanie.

Pomyślna rejestracja powoduje dodanie nowego rekordu do tabeli `User` w bazie danych. Ze względów bezpieczeństwa hasło nie jest przechowywane w postaci jawnej, ale skróconej przy pomocy funkcji SHA-256. Oprócz dodania wpisu do bazy danych, podczas tworzenia konta generowany jest dla użytkownika certyfikat X.509 potwierdzający jego tożsamość. Musi on zostać przez użytkownika pobrany z serwera



Rysunek 4.1: Formularz rejestracji użytkownika

```
1 private byte[] generateSignature(DERObjectIdentifier sigOID,
2     TBSCertificateStructure tbsCert) throws Exception {
3     Signature sig = Signature.getInstance(sigOID.getId());
4     sig.initSign(serverPrivateKey, new SecureRandom());
5     sig.update(getAsByteArray(tbsCert));
6     return sig.sign();
7 }
```

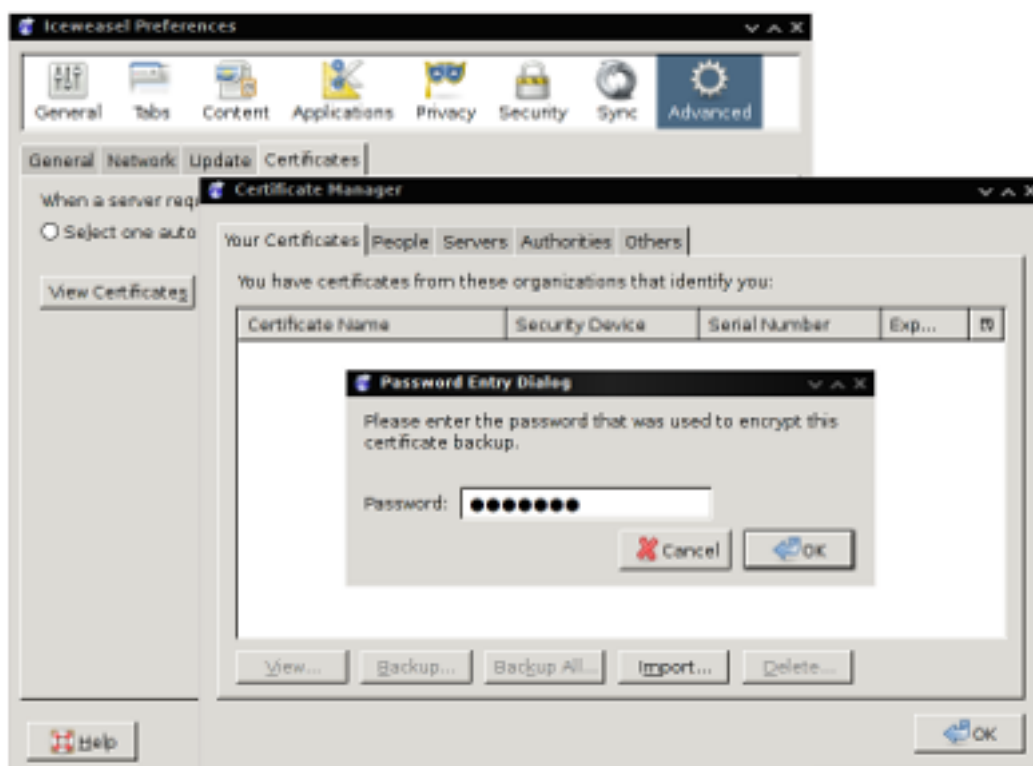
Listing 4.5: Metoda odpowiedzialna za tworzenie podpisu cyfrowego

oraz zaimportowany do przeglądarki internetowej, aby było możliwe uwierzytelnienie się w aplikacji. Jest to dodatkowe zabezpieczenie przed dostępem niepowołanej osoby do konta użytkownika. Proces importu certyfikatu do przeglądarki Iceweasel przedstawiony został na rys. 4.2.

Po zaimportowaniu certyfikat jest widoczny w przeglądarce, co przedstawiono na rys. 4.3. Pozwala to na zapoznanie się z jego właściwościami.

Certyfikat jest podpisany kluczem prywatnym serwera, co umożliwia weryfikację jego autentyczności podczas uwierzytelniania. Na listingu 4.5 przedstawiono metodę odpowiedzialną za generowanie podpisu cyfrowego.

Wszystkie procedury związane z certyfikatami X.509, takie jak ich generowanie, podpisywanie czy weryfikacja, odbywają się z wykorzystaniem pakietu `java.security` oraz biblioteki kryptograficznej BouncyCastle.



Rysunek 4.2: Import certyfikatu w przeglądarce internetowej



Rysunek 4.3: Prawidłowo zaimportowany certyfikat

4.4.2 Uwierzytelnianie

W procesie uwierzytelniania wykorzystano moduł platformy Spring Framework o nazwie Spring Security. Moduł ten dostarcza wielu przydatnych narzędzi służących do uwierzytelniania oraz autoryzacji. Za implementację bazowej funkcjonalności uwierzytelnienia przy pomocy loginu oraz hasła odpowiada klasa `AuthenticationProvider` [40]. W aplikacji została ona rozszerzona, a metoda `authenticate()` nadpisana, w celu dodania mechanizmu weryfikacji certyfikatu X.509. Dodatkowo w metodzie następuje zliczanie błędnych prób uwierzytelnienia. Liczba nieudanych prób zostaje zapisana w bazie danych oraz wyzerowana przy poprawnym uwierzy-

```
1  if (name.equals(user.getEmail())
2      && password.equals(user.getPassword()) && user.isActive()
3      && isCertificateCorrect(user.getEmail())) {
4      List<GrantedAuthority> grantedAuths
5      = new ArrayList<GrantedAuthority>();
6      for (Role r : user.getRoles()) {
7          grantedAuths.add(new SimpleGrantedAuthority(r.getName()));
8      }
9      user.setLoginAttempts(0);
10     userService.update(user);
11     return new UsernamePasswordAuthenticationToken(user,
12         password, grantedAuths);
```

Listing 4.6: Fragment metody odpowiedzialnej za uwierzytelnienie

telnieniu. Mechanizm ten może być wykorzystany do zabezpieczenia kont użytkowników przed włamaniem. Fragment tej metody przedstawiono na listingu 4.6.

Aby mogła nastąpić weryfikacja certyfikatu podczas uwierzytelniania, musi on najpierw zostać wybrany przez użytkownika w oknie pojawiającym się po wpisaniu adresu aplikacji. Jego wygląd prezentuje rys. 4.4. Jeśli użytkownik zamknie okno nie wybierając certyfikatu lub wybierze inny certyfikat niż przypisany do jego adresu e-mail, wówczas uwierzytelnienie w aplikacji nie będzie możliwe.

Aby możliwe było przedstawienie się certyfikatem, musi on wcześniej zostać zaimportowany do przeglądarki internetowej. Czynność ta została przedstawiona na rys. 4.2 oraz 4.3.

4.4.3 Autoryzacja

Aby zapobiec dostępowi niezalogowanego użytkownika do nieprzeznaczonych dla niego fragmentów aplikacji ponownie wykorzystano moduł Spring Security. Jego konfiguracja polega na edycji pliku XML, w którym zdefiniowane są uprawnienia wymagane do wyświetlenia poszczególnych podstron aplikacji. Fragment tego pliku przedstawiono na listingu 4.7.

```
1  <intercept-url pattern="authenticated/**"
2      access="isAuthenticated()" />
```

Listing 4.7: Konfiguracja dostępu do podstron aplikacji

Powyższa konfiguracja gwarantuje, że do widoków znajdujących się w katalogu o nazwie `authenticated` użytkownicy będą mieli dostęp dopiero po uwierzytelnieniu



Rysunek 4.4: Okno wyboru certyfikatu

w aplikacji. Próba dostępu do nich użytkownika niezalogowanego spowoduje zaś przekierowanie do ekranu logowania.

4.4.4 Realizacja szyfrowania

Jednym z kluczowych zadań realizowanych przez aplikację jest szyfrowanie danych. Do realizacji szyfrowania wykorzystany został pakiet `javax.crypto`, a w szczególności klasy `Cipher`, `CipherInputStream` oraz `CipherOutputStream`. Przykładową metodę odpowiedzialną za szyfrowanie danych algorytmem asymetrycznym RSA przedstawiono na listingu 4.8.

Na powyższym listingu można zaobserwować przebieg szyfrowania asymetrycznego z kluczem publicznym. Klucz ten jest pobierany z dostarczonego certyfikatu X.509.

4.5 Zachowanie poufności haseł

Jednym z zagrożeń opisanych w rozdziale „Identyfikacja słabych stron” jest przeniesienie wrażliwych danych znajdujących się w pamięci operacyjnej do przestrzeni wymiany. Takie dane to na przykład hasło do klucza prywatnego użytkow-

```
1 public static byte[] asymmetricEncrypt(byte[] dataToEncrypt,
2     X509Certificate cert) throws Exception {
3     Cipher pkCipher = Cipher.getInstance("RSA");
4     pkCipher.init(Cipher.ENCRYPT_MODE, cert.getPublicKey());
5     ByteArrayOutputStream bos = new ByteArrayOutputStream();
6     CipherOutputStream os = new CipherOutputStream(bos, pkCipher);
7     os.write(dataToEncrypt);
8     os.close();
9     return bos.toByteArray();
10 }
```

Listing 4.8: Metoda odpowiedzialna za szyfrowanie asymetryczne

```
1 Field stringValue = String.class.getDeclaredField("value");
2 stringValue.setAccessible(true);
3 String randomString = RandomGenerator
4     .generateRandomString(string.length());
5 stringValue.set(string, randomString.toCharArray());
```

Listing 4.9: Nadpisywanie hasła w pamięci operacyjnej

nika, które przez pewien czas przechowywane jest w postaci jawnej. Aby walczyć z tym zagrożeniem, wprowadzono zabezpieczenie polegające na nadpisywaniu hasła ciągiem losowych znaków w momencie, gdy przestaje być potrzebne.

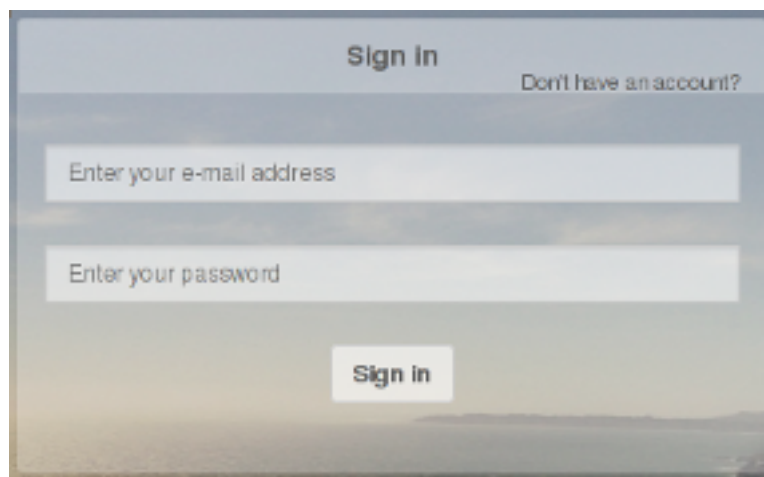
Ponieważ hasło przechowywane jest w aplikacji jako obiekt klasy `String`, nie jest możliwe jego wymazanie w standardowy sposób. Klasa `String` cechuje się bowiem tym, że raz utworzony obiekt nie może być modyfikowany. Łańcuch znaków, który stanowi reprezentację napisu, jest polem prywatnym i nie posiada publicznego akcesora. Dostęp do niego jest jednak możliwy w inny sposób - poprzez refleksję. Jest to mechanizm pozwalający na modyfikację programu w trakcie jego działania, poprzez traktowanie kodu w sposób analogiczny do danych. Dzięki temu możliwa jest na przykład zmiana specyfikatora dostępu pola z prywatnego na publiczny, a następnie jego modyfikacja, co zaprezentowano na listingu 4.9.

Nadpisywanie hasła losowymi danymi minimalizuje jego czas przebywania w pamięci operacyjnej. Dzięki temu prawdopodobieństwo jego przeniesienia do przestrzeni wymiany, a w konsekwencji pozostawienia śladu na twardym dysku, staje się pomijalnie małe.

4.6 Warstwa prezentacji

4.6.1 Implementacja widoków

Do implementacji warstwy widoku posłużyły technologie HTML oraz CSS. Na rys. 4.5 jako przykład przedstawiono formularz logowania. Jest to jeden z wielu formularzy występujących w aplikacji.



Rysunek 4.5: Formularz logowania

Do stylizacji elementów graficznego interfejsu użytkownika, takich jak formularze i przyciski, wykorzystano platformę Twitter Bootstrap. Pozwala ona w prosty sposób uzyskać przyjemny dla oka efekt wizualny, który można niewielkim nakładem pracy dostosować do własnych potrzeb. Ogromną zaletą jest też skalowalność tak otrzymanego interfejsu [41]. Została ona osiągnięta dzięki zastosowaniu reguł @media wprowadzonych w wersji trzeciej CSS. Pozwalają one na stosowanie wybranych właściwości CSS tylko dla konkretnych rozdzielczości ekranu urządzenia, dzięki czemu korzystanie z aplikacji na niewielkich ekranach nie wiąże się ze zbytnimi niedogodnościami [42].

4.6.2 Silnik szablonów

Do generowania dokumentów HTML5 wykorzystano silnik szablonów o nazwie Thymeleaf [43], który bardzo dobrze integruje się ze szkieletem aplikacji Spring MVC. Silnik ten umożliwia wyznaczanie i wyświetlanie wartości zmiennych, wywoływanie metod, a także ułatwia internacjonalizację aplikacji. W szablonie wystarczy posłużyć się kluczem wiadomości, a zostanie on automatycznie zamieniony

```
1 <div class="intro-message">
2   <h1 th:text="#{universal.applicationName}"></h1>
3   <h3 th:text="#{index.slogan}"></h3>
4   <hr class="intro-divider" />
5   <ul class="list-inline">
6     <li> <a th:text="#{login.title}" href="login"
7         class="btn btn-default btn-lg"></a> </li>
8   </ul>
9 </div>
```

Listing 4.10: Wykorzystanie silnika szablonów do internacjonalizacji

```
1 index.slogan=Protect your data
2 login.title=Sign in
3 login.registerPrompt=Don't have an account?
4 login.emailPrompt=Enter your e-mail address
5 login.passwordPrompt=Enter your password
6 register.title=Create account
```

Listing 4.11: Fragment pliku messages.properties

na wartość zdefiniowaną w pliku z wiadomościami w wybranym języku. Przykład wykorzystania tego mechanizmu przedstawiono na listingu 4.10.

Przy pomocy atrybutu `th:text` do danego elementu HTML wstawiana jest wartość tekstowa. Treści komunikatów zdefiniowane są w pliku `messages.properties`, którego fragment przedstawiono na listingu 4.11.

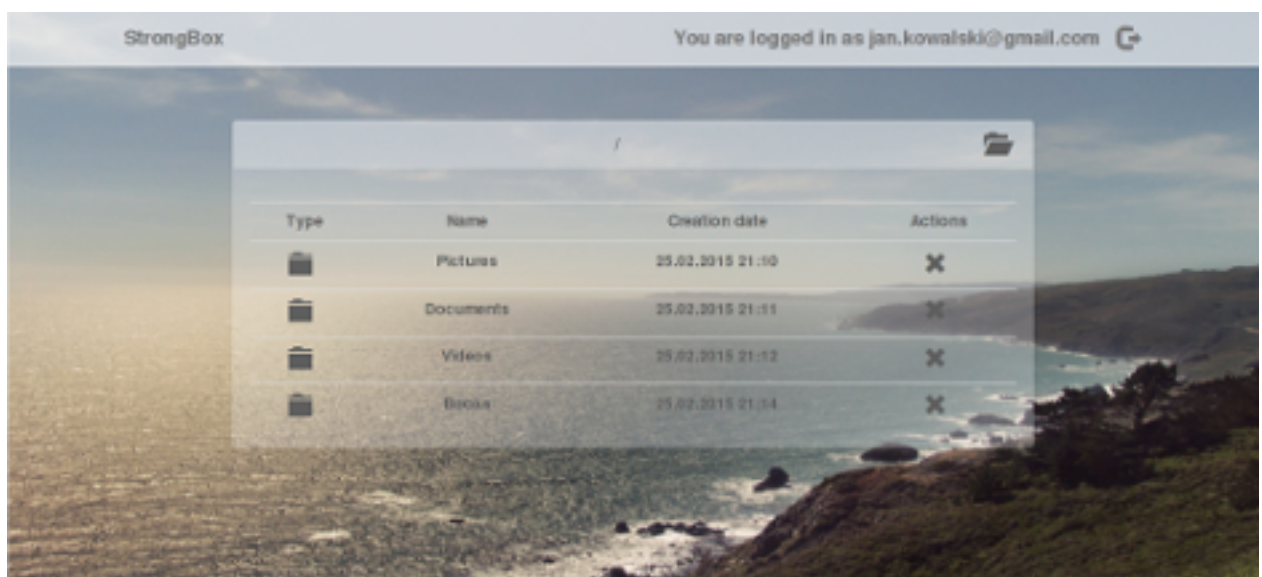
Dzięki takiemu rozwiązaniu wzbogacanie aplikacji o kolejne języki nie stanowi żadnego problemu.

Rozdział 5

Podsumowanie

5.1 Osiągnięty efekt

Efektem opisanej pracy jest aplikacja spełniająca założenia opisane w podrozdziale „Cel i założenia pracy”. Przykładowy zrzut ekranu prezentujący działającą aplikację zaprezentowany został na rys. 5.1. Przedstawiono na nim widok plików i katalogów użytkownika, widoczny po jego uwierzytelnieniu.



Rysunek 5.1: Widok prezentujący działającą aplikację

- Dzięki przywiązaniu szczególnej uwagi do kwestii bezpieczeństwa, aplikacja umożliwia przechowywanie poufnych danych bez ryzyka dostępu do nich przez osoby niepowołane. Transmisja odbywa się przy pomocy protokołu szyfrowanego, a pliki składowane są w postaci niejawnej.

- Aplikacja jest dostępna z poziomu przeglądarki internetowej, co zapewnia jej wieloplatformowość - jej uruchomienie jest możliwe na każdym z dostępnych systemów operacyjnych, łącznie z systemami przeznaczonymi dla urządzeń mobilnych.
- Korzystanie z aplikacji przy użyciu urządzeń mobilnych o niewielkich rozmiarach ekranu nie wiąże się z żadnymi niedogodnościami. Graficzny interfejs użytkownika jest w pełni responsywny, dzięki czemu prezentuje się prawidłowo niezależnie od rozdzielczości ekranu urządzenia.

5.2 Plany rozwojowe

Aplikacja spełnia swoje podstawowe zadanie, to znaczy pozwala na bezpieczne przechowywanie danych. Nie wyklucza to jednak rozwinięcia jej o dodatkowe funkcje, które mogłyby poprawić komfort korzystania z niej przez użytkowników.

- Jedną z takich funkcji byłoby umożliwienie współdzielenia plików między użytkownikami poprzez nadawanie praw odczytu i zapisu do poszczególnych plików lub katalogów.
- Dodatkowo możliwe byłoby rozbudowanie aplikacji o moduł społecznościowy, który pozwalałby użytkownikom na tworzenie listy znajomych w celu łatwiejszego udostępniania danych.
- Należy także rozważyć możliwość komunikacji poprzez wysyłanie krótkich komunikatów tekstowych.

Jak widać, możliwości rozwoju jest wiele, jednak przystępując do ulepszania aplikacji należałoby w pierwszej kolejności przeprowadzić ankietę wśród użytkowników w celu lepszego poznania ich potrzeb. Nierzadko zdarza się bowiem, że zmiana, która postrzegana jest przez twórcę aplikacji jako pozytywna, okazuje się nieakceptowalna z punktu widzenia użytkownika. Zadowolenie użytkownika jest zaś kwestią priorytetową.

5.3 Wnioski z przeprowadzonych działań

Podczas realizacji pracy zanotowano kilka spostrzeżeń dotyczących nie tylko zdefiniowanego problemu informatycznego, ale także ogólnie pojętego bezpieczeństwa.

- Nie zawsze problem jest możliwy do rozwiązania w sposób idealny. W projekcie wystąpił konflikt wymagań. Z jednej strony aplikacja miała za zadanie zachować maksymalny poziom bezpieczeństwa, a z drugiej - być przyjazna dla użytkownika i wieloplatformowa. Wymagania te okazały się nawzajem wykluczające. W całkowicie bezpiecznej aplikacji deszyfrowanie danych powinno odbywać się po stronie klienta, aby klucz prywatny użytkownika ani na chwilę nie opuścił jego komputera. Taki model został przyjęty między innymi przez aplikację Tresorit. W opisanej pracy zdecydowano się jednak na inne rozwiązanie - algorytmy kryptograficzne wykonują się po stronie serwera. Stanowi ono kompromis - dzięki temu osiągnięto wieloplatformowość aplikacji kosztem obniżenia bezpieczeństwa.
- Zapewnienie bezpieczeństwa aplikacji internetowej jest zadaniem bardzo skomplikowanym. Istnieje wiele potencjalnych słabych punktów, na które należy zwrócić uwagę w fazie projektowania systemu. W szczególności należy zabezpieczyć dostęp do serwera aplikacji oraz serwera bazy danych przed nieautoryzowanym dostępem. Pozostawienie jednej luki wystarczy, aby umożliwić włamanie, a nie sposób uzyskać pewności, że załatane zostały dokładnie wszystkie.
- Niezwykle ważne jest dokładne wykonanie i przeanalizowanie projektu przed rozpoczęciem implementacji. Wówczas możliwe jest zapobiegnięcie wielu problemom jeszcze przed ich wystąpieniem. Dokonywanie zmian projektowych w momencie gdy faza implementacji znajduje się w zaawansowanym stadium jest niezwykle kosztowne. Najczęściej efektem jest duża ilość czasu zmarnowanego na wprowadzenie poprawek, a następnie gruntowne testowanie wszystkich modułów systemu.

5.4 Realizacja efektów kształcenia

W ramach niniejszej pracy zrealizowane zostały wymagane efekty kształcenia określone w programie nauczania.

Po realizacji pracy dyplomowej inżynierskiej student potrafi:

1. *Przedstawiać w jasny sposób, zagadnienia teoretyczne niezbędne do zdefiniowania i rozwiązania wybranego problemu informatycznego. Zagadnienia teoretyczne zostały opisane w rozdziale drugim, zatytułowanym „Teoretyczne podstawy bezpieczeństwa”.*

-
2. *Definiować problem informatyczny i jego składowe, dostrzegając ich wzajemne powiązania.* Problem do rozwiązania został zdefiniowany w rozdziale pierwszym, w podrozdziale zatytułowanym „Cel i założenia pracy”.
 3. *Projektować i przeprowadzać eksperymenty informatyczne obejmujące zagadnienia niezbędne do rozwiązania nieskomplikowanego problemu informatycznego.* Wybrane eksperymenty informatyczne wykonane w celu rozwiązania problemu przedstawione zostały w rozdziale czwartym pod tytułem „Opis implementacji”.
 4. *Proponować poszczególne etapy samodzielnego rozwiązania typowego, nieskomplikowanego problemu informatycznego.* Propozycja rozwiązania wybranego problemu została przedstawiona w rozdziale trzecim, zatytułowanym „Projekt aplikacji”.
 5. *Wykorzystywać do formułowania i rozwiązywania zadania inżynierskiego wiedzę i umiejętności nabyte w trakcie studiów.* W szczególności wykorzystane zostały wiedza i umiejętności zdobyte w ramach kursów:
 - Sieciowe systemy baz danych,
 - Projektowanie systemów informatycznych,
 - Inżynieria oprogramowania.

Przykłady ich zastosowania znajdują się w rozdziałach trzecim „Projekt aplikacji” oraz czwartym „Opis implementacji”.

6. *Formułować prawidłowe wnioski i sądy dotyczące rozwiązywanego problemu informatycznego.* Wnioski dotyczące rozwiązywanego problemu opisane zostały w rozdziale piątym, zatytułowanym „Podsumowanie”.

Spis rysunków

| | | |
|-----|--|----|
| 3.1 | Diagram przypadków użycia | 17 |
| 3.2 | Diagram klas encji. | 19 |
| 3.3 | Diagram sekwencji przesyłania pliku | 20 |
| 3.4 | Diagram sekwencji pobierania pliku | 21 |
| 4.1 | Formularz rejestracji użytkownika | 27 |
| 4.2 | Import certyfikatu w przeglądarce internetowej | 28 |
| 4.3 | Prawidłowo zaimportowany certyfikat | 28 |
| 4.4 | Okno wyboru certyfikatu | 30 |
| 4.5 | Formularz logowania | 32 |
| 5.1 | Widok prezentujący działającą aplikację | 34 |

Spis listingów

| | | |
|------|---|----|
| 4.1 | Konfiguracja SSL na serwerze Apache Tomcat | 24 |
| 4.2 | Konfiguracja połączenia z bazą danych w klasie DatabaseConfig . . . | 25 |
| 4.3 | Fragment pliku konfiguracyjnego settings.properties | 25 |
| 4.4 | Przykład użycia metody createQuery klasy EntityManager | 26 |
| 4.5 | Metoda odpowiedzialna za tworzenie podpisu cyfrowego | 27 |
| 4.6 | Fragment metody odpowiedzialnej za uwierzytelnienie | 29 |
| 4.7 | Konfiguracja dostępu do podstron aplikacji | 29 |
| 4.8 | Metoda odpowiedzialna za szyfrowanie asymetryczne | 31 |
| 4.9 | Nadpisywanie hasła w pamięci operacyjnej | 31 |
| 4.10 | Wykorzystanie silnika szablonów do internacjonalizacji | 33 |
| 4.11 | Fragment pliku messages.properties | 33 |

Bibliografia

- [1] Łada Dominik, Chmury - która najlepsza? Dostępny w Internecie: <http://imagazine.pl/2012/04/25/chmury-ktora-najlepsza/>
- [2] Strona internetowa aplikacji Amazon Simple Storage: <http://aws.amazon.com/s3/>
- [3] Strona internetowa aplikacji Dropbox: <https://www.dropbox.com/home>
- [4] Strona internetowa aplikacji Google Drive: https://www.google.com/intl/pl_pl/drive/
- [5] Serafinowicz Agnieszka, Dropbox, Google Drive, OneDrive i inni - usługi chmurowe 2014 i sposób na własną chmurę. Dostępny w Internecie: <http://softonet.pl/publikacje/zestawienia/Dropbox.Google.Drive.OneDrive.i.inni-uslugi.chmurowe.2014.i.sposob.na.wlasna.chmure,168>
- [6] Gajkowski Paweł, Jaka chmura danych daje nam najwięcej miejsca za darmo? Dostępny w Internecie: <http://www.gsmmaniak.pl/152979/jaka-chmura-danych-za-darmo-dropbox-box-googledrive/>
- [7] Seltzer Larry, Don't blame Dropbox: It's all your fault. Dostępny w Internecie: <http://www.zdnet.com/article/dont-blame-dropbox-its-all-your-fault/>
- [8] Barker Ian, Weak passwords are still a major problem for business security. Dostępny w Internecie: <http://betanews.com/2014/09/30/weak-passwords-are-still-a-major-problem-for-business-security/>
- [9] Jacobsson Purewal Sarah, Update: Dropbox Will Hand Over Your Files to the Feds If Asked. Dostępny w Internecie: http://www.pcworld.com/article/225549/Dropbox_Government_Files_Turn_Over.html

-
- [10] Polityka prywatności Dropbox: <https://www.dropbox.com/pl/privacy#privacy>
 - [11] BBC News, FBI investigates 'Cloud' celebrity picture leaks. Dostępny w Internecie: <http://www.bbc.com/news/technology-29011850>
 - [12] Shema Mike, Web Security: Why You Should Always Use HTTPS. Dostępny w Internecie: <http://mashable.com/2011/05/31/https-web-security/>
 - [13] Google Developers, Podstawy elastycznego projektowania witryn. Dostępny w Internecie: <https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/>
 - [14] Hong Kaylene, Dropbox reaches 300m users, adding on 100m users in just six months. Dostępny w Internecie: <http://thenextweb.com/insider/2014/05/29/dropbox-reaches-300m-users-adding-100m-users-just-six-months/>
 - [15] Hoffman Chris, How to Secure Your Google Account with Google Authenticator. Dostępny w Internecie: <http://www.howtogeek.com/105041/how-to-secure-your-google-account-with-google-authenticator/>
 - [16] Strona internetowa aplikacji OneDrive <https://onedrive.live.com/about/pl-PL/>
 - [17] Strona internetowa aplikacji Tresorit: <https://tresorit.com/>
 - [18] Strona internetowa aplikacji Tresorit, opis wykorzystanych zabezpieczeń: <https://tresorit.com/security>
 - [19] Strona internetowa aplikacji SpiderOak: <https://spideroak.com/>
 - [20] Strona internetowa aplikacji Wuala: <https://www.wuala.com/>
 - [21] Tanenbaum Andrew, „Computer Networks”, Prentice Hall, 2010, s. 766
 - [22] Tamże, s. 778-780
 - [23] Daemen Joan, Rijmen Vincent, The Rijndael Block Cipher. AES Proposal. Dostępny w Internecie: <http://www.nist.gov/aes>
 - [24] Federal Information Processing Standards Publications, FIPS PUB 197. Announcing the Advanced Encryption Standard (AES). Dostępny w Internecie: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

-
- [25] Tanenbaum Andrew, op. cit., s. 767-769
 - [26] Tamże, s. 793-794
 - [27] Tamże, s. 796-797
 - [28] Rivest R., The MD5 Message-Digest Algorithm, RFC 1321. Dostępny w Internecie: <http://www.rfc-editor.org/rfc/rfc1321.txt>
 - [29] Eastlake 3rd D., Jones P., US Secure Hash Algorithm 1 (SHA1), RFC 3174. Dostępny w Internecie: <http://www.rfc-editor.org/rfc/rfc3174.txt>
 - [30] Tanenbaum Andrew, op. cit., s. 797-800
 - [31] Tamże, s. 810
 - [32] Tamże, s. 806-809
 - [33] Tamże, s. 853-857
 - [34] Ho Clarence, Harrop Rob, „Pro Spring 3”, Apress, 2012, s. 1-9
 - [35] Tamże, s. 54-55
 - [36] Goncalves Antonio, „Beginning Java EE 7”, Apress, 2013, s. 349-350
 - [37] Strona internetowa projektu Hibernate: <http://hibernate.org/>
 - [38] Goncalves Antonio, op. cit., s. 103-105
 - [39] Tamże, s. 192-197
 - [40] Wheeler Willie, White Joshua, „Spring in Practice”, Manning, 2013, s. 173-177
 - [41] Spurlock Jake, „Bootstrap”, O'Reilly, 2013 s. 6-8
 - [42] Hogan Brian, „HTML5 & CSS3. Develop with Tomorrow's Standards Today”, The Pragmatic Programmers, 2010, s. 94-96
 - [43] Strona internetowa projektu Thymeleaf: <http://www.thymeleaf.org/>

Łódź, dnia roku

.....
(imię i nazwisko Studenta)

.....
(adres)

.....
(numer albumu)

.....
(wydział)

.....
(kierunek studiów)

.....
(rodzaj i forma studiów)

Oświadczenie

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedstawiona praca licencjacka / inżynierska / magisterska ¹ na temat

.....
.....
.....

została napisana przeze mnie samodzielnie.

Jednocześnie oświadczam, że ww. praca

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i praw pokrewnych (j.t. Dz. U. z 2006 r. Nr 90, poz. 631, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

.....
(Czytelny podpis Studenta)

¹ Niepotrzebne skreślić

Łódź, dnia roku

.....
(imię i nazwisko Studenta)

.....
(adres)

.....
(numer albumu)

.....
(wydział)

.....
(kierunek studiów)

.....
(rodzaj i forma studiów)

Oświadczenie

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedstawiona na nośniku CD/DVD praca licencjacka / inżynierska / magisterska ¹ na temat

.....
.....
.....

zawiera te same treści, co oceniany przez Opiekuna pracy i Recenzenta wydruk komputerowy.

.....
(Czytelny podpis Studenta)

¹ Niepotrzebne skreślić