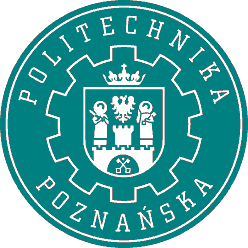
**POLITECHNIKA POZNAŃSKA**

**WYDZIAŁ INFORMATYKI**

**INSTYTUT INFORMATYKI**

Zastosowanie steganografii do wymiany informacji na forum internetowym

DANIEL KOZA  
BARTOSZ KOSTANIAK

Promotor: prof. dr hab. inż. Jerzy Brzeziński   
Instytut Informatyki   
Politechniki Poznańskiej

**KARTA PRACY DYPLOMOWEJ**

(kserokopia z podpisami)

**Spis treści**

[1. Wstęp 3](#_Toc439342065)

[1.1 Cel i zakres pracy 4](#_Toc439342066)

[1.2 Struktura pracy 4](#_Toc439342067)

[2. Wprowadzenie 5](#_Toc439342068)

[2.1 Rys historyczny 5](#_Toc439342069)

[2.2 Idea steganografii 5](#_Toc439342070)

[2.3 Nośniki informacji umożliwiające zaimplementowanie steganografii 5](#_Toc439342071)

[2.4 Narzędzia komunikacji w Internecie 5](#_Toc439342072)

[2.4.1 Czat internetowy 5](#_Toc439342073)

[2.4.2 Forum internetowe 5](#_Toc439342074)

[2.5 Steganografia w plikach graficznych 5](#_Toc439342075)

[3. Przedstawienie użytych technologii 6](#_Toc439342076)

[3.1 Git 6](#_Toc439342077)

[3.2 JavaScript 6](#_Toc439342078)

[3.3 Node.JS 6](#_Toc439342079)

[3.3.1 Platforma Node.JS 6](#_Toc439342080)

[3.3.2 Node Package Manager (NPM) 6](#_Toc439342081)

[3.4 MongoDB 7](#_Toc439342082)

[3.5 MeteorJS 7](#_Toc439342083)

[3.5.1 Ogólny opis 7](#_Toc439342084)

[3.5.2 Struktura plików 7](#_Toc439342085)

[3.5.3 Modele i metody 8](#_Toc439342086)

[3.6 AngularJS (wersja 1.x) 8](#_Toc439342087)

[3.6.1 Ogólny opis 8](#_Toc439342088)

[3.6.2 Jednostki 9](#_Toc439342089)

[3.6.3 Angular-Meteor 9](#_Toc439342090)

[3.7 Preprocessory 9](#_Toc439342091)

[3.7.1 CoffeeScript 9](#_Toc439342092)

[3.7.2 Jade 9](#_Toc439342093)

[3.7.3 Less 9](#_Toc439342094)

[3.8 Narzędzia do zarządzania projektem 9](#_Toc439342095)

[3.8.1 TravisCI 9](#_Toc439342096)

[3.9 Narzędzia do testowania 9](#_Toc439342097)

[3.9.1 GulpJS 9](#_Toc439342098)

[3.9.2 Jasmine 9](#_Toc439342099)

[3.9.3 Karma 9](#_Toc439342100)

[3.9.4 Protractor 9](#_Toc439342101)

[4. Budowa aplikacji 10](#_Toc439342102)

[4.1 Struktura plików i folderów 10](#_Toc439342103)

[4.2 Modele w bazie danych 10](#_Toc439342104)

[4.3 Jednostki w aplikacji 10](#_Toc439342105)

[5. System uwierzytelniania i autoryzacji 11](#_Toc439342106)

[5.1 Rejestrowanie i logowanie użytkowników 11](#_Toc439342107)

[5.2 Role 11](#_Toc439342108)

[5.3 Struktura ról 11](#_Toc439342109)

[5.4 Panel administracyjny 11](#_Toc439342110)

[6. System ukrywania wiadomości 12](#_Toc439342111)

[6.1 12](#_Toc439342112)

[7. Testy 13](#_Toc439342113)

[7.1 Testy jednostkowe 13](#_Toc439342114)

[7.2 Testy integracyjne 13](#_Toc439342115)

[7.3 Testy funkcjonalne 13](#_Toc439342116)

[8. Zakończenie 14](#_Toc439342117)

[Literatura 15](#_Toc439342118)

# Wstęp

Wraz z wykształceniem się u człowieka umiejętności komunikacji powstał problem skutecznego przekazywania tajnych informacji, takich których zawartość mogłaby być znana tylko przez nadawcę i odbiorcę. Jak bezpiecznie przesłać list do adresata, tak aby pośrednik nie był w stanie jej odczytać? Jak wysłać do króla wiadomość, której nikt inny nie byłby w stanie odczytać? Jak przekazać polecenia dla szpiega, który znajduje się za linią frontu? Te i wiele innych pytań towarzyszy ludzkości praktycznie od powstania cywilizacji. Najczęstszym rozwiązaniem tych problemów będzie szyfrowanie wiadomości przez nadawcę i odszyfrowania jej za pomocą tajnego klucza przez odbiorcę. Dziedzinę nauki zajmującą się przekazywaniem informacji w sposób zabezpieczony przed niepowołanym dostępem nazywamy kryptologią. Dzieli się ona na kryprografię, czyli gałąź wiedzy o utajnianiu wiadomości oraz kryptoanalizę, czyli gałąź wiedzy o przełamywaniu zabezpieczeń oraz deszyfrowaniu wiadomości. Przekazywanie informacji z pomocą kryptografii ma jednak jedną, ale zasadniczą wadę: obecność tajnego komunikatu jest jawna, choć treść jest zaszyfrowana. Steganografia rozwiązuje ten problem. Jest to nauka o komunikacji w taki sposób, aby obecność komunikatu nie mogła zostać wykryta. Połączenie tych dwóch dziedzin nauki daje nam najwyższy możliwy poziom bezpieczeństwa w przekazywaniu tajnych wiadomości.

Warto też zauważyć jak bardzo na przestrzeni dziejów zmieniały się narzędzia komunikacji. Zaczynając od prostej mowy, pisma, przechodząc przez radio, telewizję, a kończąc na Internecie człowiek zyskał bardzo wiele możliwości przesyłania dowolnej wiadomości (np. w formie pliku graficznego, filmu) na dowolną odległość. Dzięki temu implementacja steganografii w dwudziestym pierwszym wieku jest teoretycznie ograniczona jedynie przez ludzką wyobraźnię.

Motywacją do podjęcia tematu była chęć stworzenia jak najbezpieczniejszego narzędzia do komunikacji w Internecie. Praktycznie każda większa korporacja, rządy państw nieustannie zbierają informację na temat każdego użytkownika przeglądającego witryny WWW. Dzięki forum steganograficznemu przeciętny człowiek, bez większych umiejętności informatycznych, będzie w stanie wymieniać wiadomości, nie obawiając się o szpiegowanie ze stron trzecich. Istnieje co prawda tzw. Darknet (pl. Ciemny Internet), czyli miejsce gdzie istnieje wolność absolutna, jednakże jest on trudny w obsłudze dla przeciętnego użytkownika.

## Cel i zakres pracy

Celem niniejszej pracy jest stworzenie w pełni funkcjonalnego forum internetowego wraz z zaimplementowanym mechanizmem do ukrywania tajnych wiadomości. Aplikacja powinna umożliwiać rejestrację i logowanie użytkowników, tworzenie, edycję, przeglądanie i usuwanie sekcji, tematów oraz postów. Posty powinny należeć do tematów, a tematy do sekcji. Należy umożliwić wgranie pliku graficznego i przypisanie go do danego postu.

## Struktura pracy

Dalsza część pracy zorganizowana jest w sposób następujący. Rozdział 2 zawiera ...

# Wprowadzenie

## Rys historyczny

## Idea steganografii

## Nośniki informacji umożliwiające zaimplementowanie steganografii

## Narzędzia komunikacji w Internecie

### Czat internetowy

### Forum internetowe

## Steganografia w plikach graficznych

# Przedstawienie użytych technologii

## Git

Git jest rozproszonym systemem kontroli wersji.

## JavaScript

JavaScript jest skryptowym językiem programowania stosowanym najczęściej na stronach internetowych, jednakże dzięki różnym bibliotekom coraz częściej jest używany także w innych środowiskach. Łączy w sobie następujące paradygmaty: obiektowy, funkcyjny oraz imperatywny. Jest on dynamiczne i słabo typowany, czyli typy zmiennych są nadawane dynamicznie w czasie wykonywania skryptu, a konwersje na różne typy danych są wykonywane automatycznie. W języku tym występują obiekty, takie jak: Object, Array oraz typy prymitywne: Boolean, Number, String, Null, Undefined. Standardem JavaScript jest ECMAScript, czyli ustandaryzowany przez organizację ECMA skryptowy, obiektowy język programowania. Specyfikacja ta oznaczone jest jako ECMA-262 i ISO/IEC 16262.

## Node.JS

### Platforma Node.JS

Node.JS jest platformą umożliwiającą uruchamianie skryptów JavaScript (JS) w innym środowisku niż przeglądarka internetowa. Korzysta ona z otwartego silnika V8, który został stworzony i jest rozwijany przez firmę Google. Wykorzystuje go popularna przeglądarka internetowa Google Chrome. Node.JS korzysta także z biblioteki libuv, która zapewnia asynchroniczne powiadamianie o zdarzeniach, jak i również z wielu innych bibliotek, dzięki czemu Node.JS jest w środowiskiem programistycznym wykorzystywanym do tworzenia wysoce skalowalnych aplikacji internetowych, sterowanych zdarzeniami i wykorzystujących asynchroniczny system wejścia-wyjścia.

### Node Package Manager (NPM)

W platformę Node.JS wbudowany jest system modułów nazywany Node Package Manager, w skrócie NPM. Umożliwia on doinstalowywanie zewnętrznych modułów do aplikacji stworzonej w środowisku Node.JS. Za pomocą jednej komendy jesteśmy w stanie ściagnąć konkretną wersję danej biblioteki. Dodatkowo jest możliwość zapisania listy używanych bibliotek do pliku package.json, który znajduje się w głównym katalogu projektu, dzięki czemu inny programista będzie mógł bez problemów ściągnąć wszystkie potrzebne zależności, które nie będą się znajdowały w zdalnym repozytorium.

## MongoDB

MongoDB jest otwartym, nierelacyjnym systemem zarządzania danych. Charakteryzuje się dużą skalowalnością, wydajnością oraz dowolnością w strukturze baz danych. Dane są składowane w postaci dokumentów JSON (JavaScript Object Notation), dzięki czemu umożliwia to aplikacjom bardziej naturalne ich przetwarzanie, zwłaszcza przez biblioteki napisane w języku JavaScript.

## MeteorJS

### Ogólny opis

MeteorJS jest biblioteką wykorzystywaną do tworzenia aplikacji internetowych. Jest ona napisana z użyciem platformy Node.JS. Jest zintegrowana z nierelacyjnymi bazami danych MongoDB. Działa zarówno po stronie klienta (przeglądarka internetowa), jak i serwera. Wykorzystuje wzorzec projektowy publikuj-subskrybuj do automatycznego propagowania modyfikacji danych w stronę klienta bez konieczności pisania kodu synchronizującego przez programistę. Dzięki temu stworzenie w pełni funkcjonalnej aplikacji internetowej wymaga dużo mniej pracy niż w innych bibliotekach, takich jak Ruby on Rails czy Django. Jedną z głównych zalet MeteorJS jest to, że cała komunikacja między przeglądarką internetową a serwerem jest wykonywana za pomocą WebSocket’ów. WebSocket jest to technologia zapewniająca dwukierunkowy kanał komunikacji za pośrednictwem jednego gniazda TCP.

MeteorJS posiada własny menadżer bibliotek i wtyczek, AtmosphereJS, który bardzo przypomina ten z Node.JS. Podobnie jak w NPM, możemy zainstalować zewnętrzną bibliotekę za pomocą jednego polecenia, a informacje o niej są zapisywane do pliku. Jednakże, w odróżnieniu od platformy Node.JS, nie musimy ręcznie dodawać referencji do wtyczek w kodzie, albowiem wszystkie biblioteki są dostępne globalnie.

### Struktura plików

MeteorJS posiada z góry określoną strukturę plików i kolejność ich ładowania, co ma duże znaczenie, zwłaszcza jeżeli korzystamy z dodatkowych bibliotek, takich jak na przykład AngularJS. W uproszczeniu: wszystko co znajduje się w folderze .meteor służy do konfiguracji wewnętrznych i zewnętrznych biblotek i ich wersji, zawartość folderu server jest widoczna i wykonywana tylko po stronie serwera, podobnie jak katalogu private, pliki w folderze client natomiast są wykonywane i widocznie tylko po stronie klienta, czyli przeglądarki internetowej, a skrypty w katalogu tests nie są widoczne ani przez serwer, ani przez klient, służą one tylko i wyłącznie do wykonywania testów. Kolejność ładowania plików jest następująca: wszystkie skrypty znajdujące się w katalogu lib ładowane są w pierwszej kolejności, następnie MeteorJS wczytuje resztę plików według kolejności alfabetycznej, najpierw z podkatalogu, a później z katalogu nadrzędnego, z wyjątkiem plików rozpoczynających się słowem main, które są ładowane na samym końcu.

### Modele i metody

Dane w MeteorJS są przechowywane za pomocą kolekcji. Jako że biblioteka ta korzysta z MongoDB, struktura modeli w bazie danych nie jest określona. Kolekcje tworzy się raz i są dostępne globalnie.

Listing 1. Przykład inicjalizacji kolekcji Messages dostępnej globalnie.

Messages = new Mongo.Collection("messages");

W MeteorJS używany jest po stronie serwera obiekt globalny Meteor, który posiada metodę publish oraz methods. Publish służy do wyświetlania danych bezpośrednio z MongoDB. Każda aktualizacja modelu lub kolekcji w bazie danych jest od razu widoczna w przeglądarce internetowej, bez żadnego przeładowywania strony. Oznacza, to że dane publikowane za pomocą tej metody są bezpośrednio połączone z danymi wyświetlanymi w widoku. Methods natomiast służy do definiowania różnych akcji modyfikujących dane. Zarówno do metody publish, jak i methods, przesyłamy funkcję, która, w pierwszym wypadku musi zwrócić wektor z bazy danych, a w drugim może zwrócić dowolną wartość.

Listing 2. Przykład metody publish, która zwraca wszystkie obiekty z kolekcji Rooms

Meteor.publish("rooms", function () {

return Rooms.find({});

});

## 

Listing 3. Przykład użycia methods. Podany kod tworzy dwie metody: foo oraz bar.

Meteor.methods({

foo: function (arg1, arg2) {

check(arg1, String);

check(arg2, [Number]);

return "some return value";

},

bar: function () {

return "baz";

}

});

## AngularJS (wersja 1.x)

### Ogólny opis

AngularJS w wersji 1.x jest jedną z najbardziej popularnych bibliotek, napisanych w języku JavaScript, do tworzenia SPA (Single Page Application), czyli stron internetowych, które bardzo przypominają aplikacje działające natywnie w środowisku Windows, Linux czy Mac OSX. Wszelkie akcje, jakie użytkownik wykonuje, dzieją się praktycznie tylko na jednej stronie, bez żadnego przeładowywania, a wszystkie dane pobierane i wysyłane do serwera są wykonywane za pomocą technologii AJAX (Asynchronous JavaScript and XML). Jednakże użytkownik ma wrażenie, że użytkuję normalną stronę, miedzy innymi z powodu zmieniających się widoków i adresów URL w przeglądarce internetowej. AngularJS korzysta z popularnego wzorca projektowego MVC (Model View Controller), delikatnie go modyfikując. AngularJS jest stosunkowo młodą biblioteką, albowiem został opublikowany w 2012 roku przez firmę Google. Dzięki wsparciu tej korporacji stał się wiodącą technologią w dziedzinie tworzenia nowoczesnych aplikacji internetowych. W 2016 roku ma zostać wydana wersja 2.0, która w bardzo dużym stopniu różni się od aktualnej edycji. Dlatego tak ważne jest podanie, której wersji programista używa.

### Jednostki

AngularJS opiera się na modułach. Są to podstawowe jednostki w tej bibliotece, dzięki którym organizujemy całą aplikację. Do każdego modułu możemy wstrzyknąć inny moduł, przez co tworzymy drzewo zależności i unikamy globalnego definiowania obiektów. W bardzo dużym stopniu ułatwia to testowanie jednostkowe, ponieważ wstrzykujemy do testu tylko jeden moduł. Wszystkie jednostki opisane poniżej tworzymy właśnie w modułach.

Listing 4. Przykład inicjalizacji modułu myApp wraz ze wstrzykniętym modułem restangular

var myApp = angular.module('myApp',['restangular']);

Rolę modelu we wzorcu MVC pełnią tutaj serwisy, takie jak: service, factory, provider. Mają one dwie podstawowe cechy: AngularJS inicjalizuje je wtedy i tylko wtedy, gdy jakiś komponent od nich zależy, a także korzystają ze wzorca singleton, czyli każdy komponent otrzymuje tę samą, pojedynczą referencję do danego serwisu. Jednostki te odpowiadają głównie za logikę biznesową aplikacji oraz za komunikację z serwerem. Do serwisu możemy wstrzyknąć inny serwis.

Listing 5. Przykład serwisu w AngularJS

angular.

module('myServiceModule', []).

factory('notify', ['$window', function(win) {

var msgs = [];

return function(msg) {

msgs.push(msg);

if (msgs.length == 3) {

win.alert(msgs.join("\n"));

msgs = [];

}

};

}]);

Jedną z podstawowych jednostek w AngularJS są kontrolery, czyli literka C we wzorcu MVC. Odpowiadają one za zarządzanie danymi w widoku oraz za wykorzystywanie serwisów. Są łącznikiem między warstwą logiki a warstwą prezentacji aplikacji. W przeciwieństwie do serwisów, które są inicjalizowane tylko raz w trakcie używania aplikacji, kontrolery są tworzone za każdym razem, gdy dany widok od nich zależy i niszczone, gdy są już nieużywane. Ma to ogromne znaczenie w procesie projektowania aplikacji. Kontrolery powinny zawierać jak najmniej kodu i nie zarządzać logiką biznesową, ze względu na to, że nie przechowują danych (są niszczone) i nie da się ich wstrzyknąć do innego komponentu.

Listing 6. Przykład kontrolera ze wstrzykniętym serwisem $scope w AngularJS

myApp.controller('DoubleController', ['$scope', function($scope) {

$scope.double = function(value) {

return value \* 2;

};

}]);

Esencją AngularJS są dyrektywy. To właśnie dzięki nim ta biblioteka zyskała taką popularność. Jak wiadomo, HTML ma ograniczoną liczbę elementów i atrybutów. Często brakuje jakiegoś znacznika lub chcielibyśmy nie powielać tego samego kodu (ewentualnie w małym stopniu różniącego się) w różnych częściach aplikacji. Z pomocą przychodzą tutaj właśnie dyrektywy.

Listing 7. Przykład dyrektywy w AngularJS

### 

### Two-way binding

### Angular-Meteor

## Preprocessory

### CoffeeScript

### Jade

### Less

## Narzędzia do zarządzania projektem

### TravisCI

## Narzędzia do testowania

### GulpJS

### Jasmine

### Karma

### Protractor

# Budowa aplikacji

## Struktura plików i folderów

## Modele w bazie danych

## Jednostki w aplikacji

# System uwierzytelniania i autoryzacji

## Rejestrowanie i logowanie użytkowników

## Role

## Struktura ról

## Panel administracyjny

# System ukrywania wiadomości

## 

# Testy

## Testy jednostkowe

## Testy integracyjne

## Testy funkcjonalne

# Zakończenie

# Literatura

1. Autorzy: Tytuł, Informacje wydawnicze, data
2. Autorzy: Tytuł, Informacje wydawnicze, data