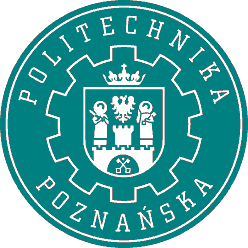
**POLITECHNIKA POZNAŃSKA**

**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](#_Toc438506422)

[1.1 Cel i zakres pracy 4](#_Toc438506423)

[1.2 Struktura pracy 4](#_Toc438506424)

[2. Wprowadzenie 5](#_Toc438506425)

[2.1 Rys historyczny 5](#_Toc438506426)

[2.2 Idea steganografii 5](#_Toc438506427)

[2.3 Nośniki informacji umożliwiające zaimplementowanie steganografii 5](#_Toc438506428)

[2.4 Narzędzia komunikacji w Internecie 5](#_Toc438506429)

[2.4.1 Czat internetowy 5](#_Toc438506430)

[2.4.2 Forum internetowe 5](#_Toc438506431)

[2.5 Steganografia w plikach graficznych 5](#_Toc438506432)

[3. Przedstawienie użytych technologii 6](#_Toc438506433)

[3.1 NodeJS 6](#_Toc438506434)

[3.2 MeteorJS 6](#_Toc438506435)

[3.3 MongoDB 6](#_Toc438506436)

[3.4 AngularJS (wersja 1.x) 6](#_Toc438506437)

[3.5 Preprocessory 6](#_Toc438506438)

[3.5.1 CoffeeScript 6](#_Toc438506439)

[3.5.2 Jade 6](#_Toc438506440)

[3.5.3 Less 6](#_Toc438506441)

[3.6 Narzędzia do zarządzania projektem 6](#_Toc438506442)

[3.6.1 Git 6](#_Toc438506443)

[3.6.2 TravisCI 6](#_Toc438506444)

[3.7 Narzędzia do testowania 6](#_Toc438506445)

[3.7.1 GulpJS 6](#_Toc438506446)

[3.7.2 Jasmine 6](#_Toc438506447)

[3.7.3 Karma 6](#_Toc438506448)

[3.7.4 Protractor 6](#_Toc438506449)

[4. Budowa aplikacji 7](#_Toc438506450)

[4.1 Struktura plików i folderów 7](#_Toc438506451)

[4.2 Modele w bazie danych 7](#_Toc438506452)

[4.3 Jednostki w aplikacji 7](#_Toc438506453)

[5. System uwierzytelniania i autoryzacji 8](#_Toc438506454)

[5.1 Rejestrowanie i logowanie użytkowników 8](#_Toc438506455)

[5.2 Role 8](#_Toc438506456)

[5.3 Struktura ról 8](#_Toc438506457)

[6. Algorytm steganograficzny 9](#_Toc438506458)

[6.1 9](#_Toc438506459)

[7. Zakończenie 10](#_Toc438506460)

[Literatura 11](#_Toc438506461)

# 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.

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.

## AngularJS (wersja 1.x)

### 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