

**Search Engine Optimized Blog**

***SEO BLOG***



# Spis treści

[Spis treści 0](#_Toc499753140)

[Overview 0](#_Toc499753141)

[Features 0](#_Toc499753142)

[Client side 1](#_Toc499753143)

[Search engine 1](#_Toc499753144)

[Admin Panel 6](#_Toc499753145)

# Overview

This application should allow to create high search engine optimized blog.

# Features

--(server) - means that this features is provided by server side only

--#1, #2, #3, ... - means feature priority

## Client side

1. Contact form #1
2. Reviews and/or comments with external identify (google, Facebook) #2
3. Social Media buttons (likes, twits,...) #2
4. Desktop/Tablet/Mobile view #1
5. Content elements:
   1. Logo #1
   2. Navigation Menu #1 (Menus: top, main, left)#3
   3. Slider #1
   4. Headings #1
   5. Paragraphs #1
   6. Images #1
   7. Videos #1
   8. Special blocks:
      1. Code block (c#, js, Html) #3
      2. draw.io viewer #4

## Search engine

1. Accelerated Mobile Pages (AMP) #1

This technology allows google to cache pages in their cache its speed up page loading from google search. If page support this technology than page is higher in search results

**Implementation:** <https://www.ampproject.org/>

1. Structured data: #1
   1. Website
   2. Webpage
   3. Organization
   4. Article
   5. Breadcrumbs
   6. Site navigation elements

**Implementation:** <http://schema.org/> and <https://developers.google.com/search/docs/guides/search-features>

1. Gzip compression (server) #1

**Implementation:** <https://docs.microsoft.com/en-us/iis/configuration/system.webserver/httpcompression/>

1. Keep-alive attribute (server) #1

**Implementation:** <https://docs.microsoft.com/en-us/iis/configuration/system.webserver/httpprotocol/>

1. CDN (server) #4
2. Optimized images (server) #1

**Implementation:** Use tiny png Api for .Net. Each blog instances should have own TinyPNG Api Key, it should be configured by user in admin panel. 500 transformation per month is free. Use images store (for reuse and cross site update) which will be minified while publication or while edition if it will be required (crop/resize). <https://tinypng.com/developers>

1. Google Analytics (server) #1

**Implementation:** Allow to insert GA script to page

1. Google Tag Manager (server) #2

**Implementation:** Allow to insert GTM script to page

1. Google webmaster tools (server) #1

**Implementation:**

* + - 1. Use GA script
      2. Use GTM scrit
      3. Allow to add custom meta tag

1. Robots.txt (server) #1

**Implementation:** Add static Robots.txt file to page

1. H1,H2,H3,H4,strong, #1

**Implementation:** Use h1, h2, … tags instead of custom class styling on span elements for example

1. Friendly URLs #1

**Implementation:** Allow to create simple page address

1. Image Alts #1

**Implementation:** Allow add alts to images (if image store is used allow to override default alt)

1. No inline CSS #1

**Implementation:** Do not use inline style in application

1. Data URIs for top page images (logo especially) (server) #2

**Implementation:** For small images on top of page use dataUrls to insert images without loading it as additional resource (for example logo). Do not use this for large image. As example image data should not change percentage of text size to page size more than 10% of original percentage. Example if textSize/pageSize = 10% max data image impact should be 1%.  
(there no directions for this and above number are just example – this is hidden in Google search engine parameters)

1. Server Cache (server) #1

**Implementation:** Server cache allows to speed up server response. Application should hold on cache all pages state. Data should be reload if user publish new version of specific object.

**-** Use Cache object no cache timeout, with reload cache event and cache dependency

1. Expires headers (browser cache) (server) #1

**Implementations:** Add custom header to page for all page and resources.

**-** Use small cache expiration time for Page

**-** Use long cache expiration time for all Resources with dynamic resources rout changed for each resources publications.

1. JS/CSS minimalization (server) #1

**Implementation:** Styles send to user should be dynamically build and compressed.

1. Css obfuscation (second minimalization level) #1

**Implementation:** Add dynamically obfuscation which will change all style names to as small as possible:  
**Change**:   
- HTML:<div class=”myLongNameCssClass1 myLongNameCssClass2”>…  
- CSS: div.myLongNameCssClass1.myLongNameCssClass2{…}  
**To**:  
- HTML: <div class=”a b”>…  
- CSS: div.a.b{…}

1. URL canonicalization www to non www, http to https (server) #1

**Implementation:** Add to application redirection to single url format: https://domain.name

1. Https (server) #1

**Implementation:** Allow https connection buy certification on hosting or choose hosting with Let’s Encrypt certificate.

1. No server signature (server) #2

**Implementations:** Remove server identification headers from server response – this gives more security than SEO

1. Email protection #2

**Implementation:** Use client side scripts to fill email texts and links instead send them directly in html – this gives more security than SEO but it minimalize spam messages.

1. Top page style including to page code #2

**Implementation:** Add top page required styles directly into html. This allows to show first application screen without loading one more resources and unnecessary styles. Styles have to be divided to first view required styles and full functionality styles.

**For example:** For menu with dropdown we should not load opened dropdown styles, dropdown elements styles or hover styles, if we can identify device resolution we can send only xs, sm or lg styles.

1. Meta tags:
   1. Hreflang (server) #1

**Implement:** Add meta tags list with supported languages

* 1. Description (server) #1

**Implement:** Add configured page meta description to page

* 1. Charset (server) #1
  2. Robots (server) #2

**Implement:** Allow to remove current page from google by adding correct meta tag to browser

* 1. Social media meta tags: #1
     1. Facebook
     2. Twitter
     3. Pinterest

**Implement:** Allow to configure meta tags used by above applications. This should be as easy as possible for user. Example meta page image should be pre inserted by first Image on page or logo and user should be able to change it.

1. Links:
   1. Canonical (server) #2
   2. Pagination (prev, next) (server) #2

**Implement:** If application contains multiple same page version for example: AMP, Print, or blog with parameter changed page layout dark/white. Than canonical url should tell which page address is the original one. We can use base url

<https://moz.com/blog/canonical-url-tag-the-most-important-advancement-in-seo-practices-since-sitemaps>

For example:

**Page Urls:**

- http://mydomain.pl/amp/blog-list?p=3

- http://mydomain.pl/blog-list?p=2&style=dark

- http://mydomain.pl/print/blog-list?p=3

**Canonical URL:**

- http://mydomain.pl/blog-list?p=3

**Prev URL:**

- http://mydomain.pl/blog-list?p=2

**Next URL:**

- http://mydomain.pl/blog-list?p=4

**Base URL:**

- http://mydomain.pl/blog-list

1. Title tag (server) #1
2. Site map (server) #1

**Implementation:** Application should dynamically generate site map with all published pages. Site map can contains list of pages with url, priority, modification date and update frequent. We can add sitemap index too to inform engine about multiple site maps (only if single site map contains more than 50 000 pages or more than 50MB)

<https://www.sitemaps.org/protocol.html>

1. RWD #1
2. Only required styles loading #2

**Implementation:** Dynamically build list of required styles on current page it can be combined/replaced with point 24

1. Only required JS loading #3

## Admin Panel

1. Working copy/copies / Publications #1
2. Versions history? #2
3. Allow users to edit texts just in end view html #1
4. Allow to modify element properties in properties block (right side of window) #1:
   1. All elements
      1. Alignment
      2. Width
   2. Text
      1. Colour
      2. Decoration
   3. Image
      1. Alt
      2. Url (inline/block)
   4. Link
      1. Url
      2. Text (inline)
      3. Target
   5. Gogle Map (iframe)
      1. Url
5. Text properties should use class styling (no inline styles and additional tags) #1
6. Allow users to build his website structure in tree. (notify user if tree have more than 3 levels) #1
7. Allow user to use tags for his articles
8. Allow user to create series of blogs
9. Allow user to add new element to page #1
10. Allow user to move elements #1
11. Desktop View edition with Mobile View Block(Preview/Editon) #3
    1. User can choose device from list
    2. Set custom width, height
    3. Horizontal/Vertical