Installation and configuration guide

Last revision 16.07.2020

Table of contents

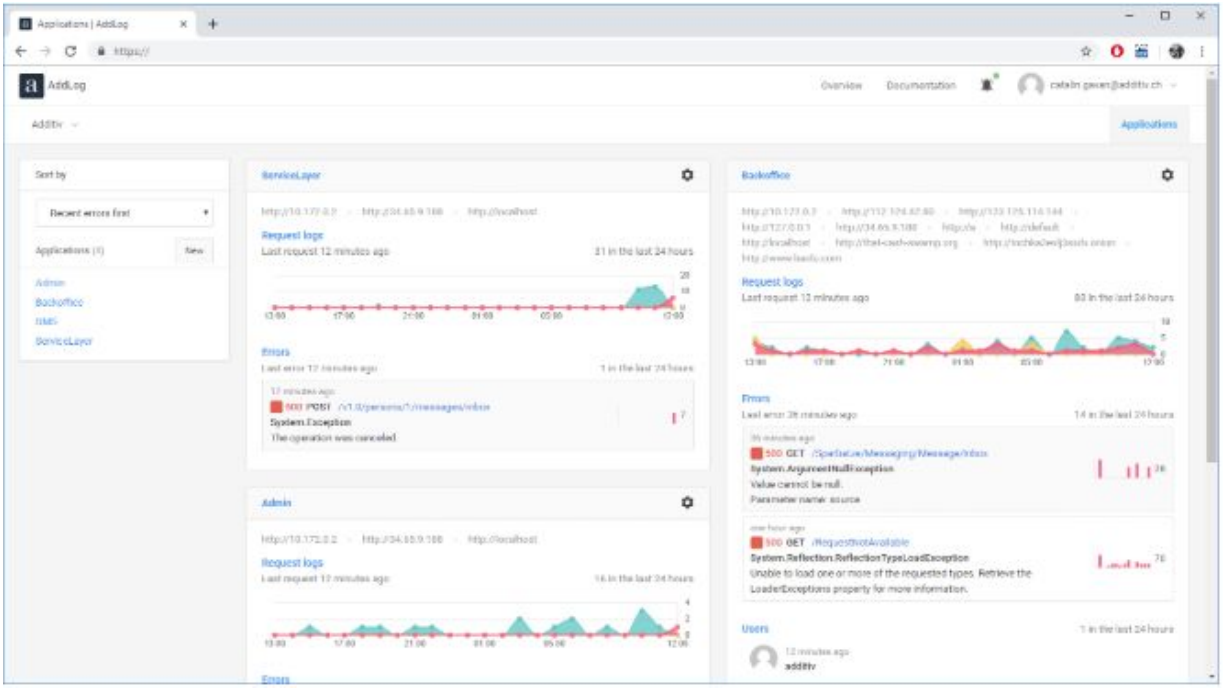
Last revision 16.07.2020

About

This guide will walk through installation and configuration steps for deploying KissLog on-premises.

If you have any issues or questions, please feel free to send me an email to catalingavan@gmail.com and I will try to answer them as soon as possible.

KissLog application screen



Links

KissLog.net
https://kisslog.net/

KissLog SDK
https://github.com/KissLog-net/KissLog.Sdk

Last revision 16.07.2020

Prerequisites

Artifacts

1. KissLog.Backend.AspNetCore.zip
2. KissLog.Frontend.AspNetCore.zip

Services

KissLog on-premises requires the following services:

1. MongoDB Community Server (free)
2. MS-SQL Server or MySQL Community Server
3. IIS web server with .NET Core 3.1 Runtime

Last revision 16.07.2020

Architecture

KissLog on-premises consists of two .NET Core 3.1 web applications:

KissLog.Frontend

User-interface application used by users (developers, IT administrators, application managers and implementation consultants) to visualise the captured errors, logs and other metrics data.

KissLog.Backend

Backend application responsible for managing the logs data. Exposes REST endpoints which can be used to save and to query the logs.

Storage providers

KissLog uses two storage providers:

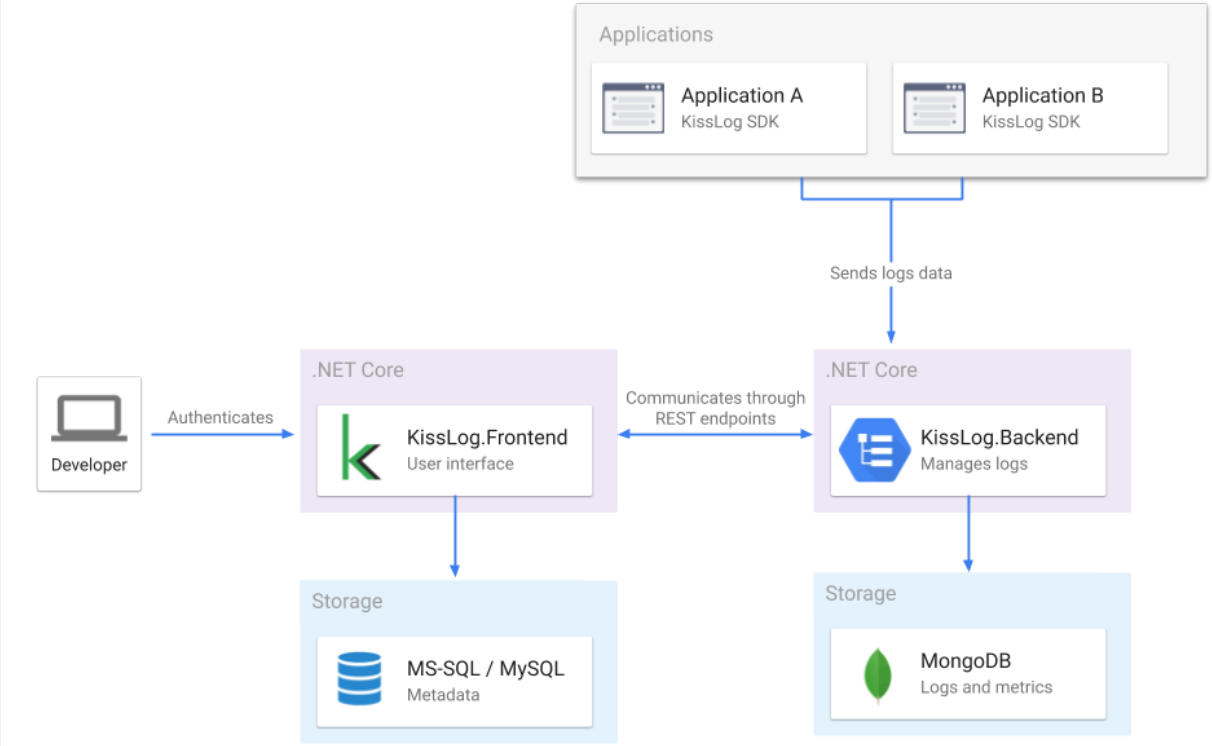
1. MS-SQL / MySQL

   Used for storing metadata information in the database, including information about organizations, applications and other user-interface related properties.

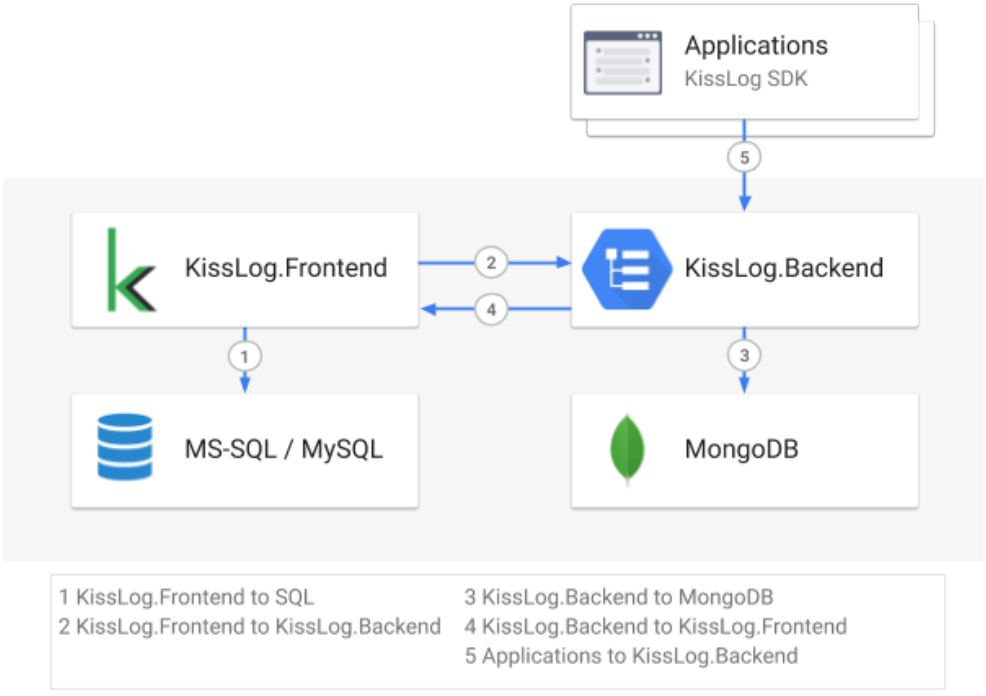
2. MongoDB

   Used for storing logs, captured exceptions and other metrics data.

KissLog architecture



Network access



Last revision 16.07.2020

Installation

MongoDB

For installing MongoDB server, please check the official tutorial.

MS-SQL / MySQL

We will not cover the installation guide for these services. There is a high possibility that the existing system will already have a running instance of MySQL or MS-SQL server.

IIS web applications

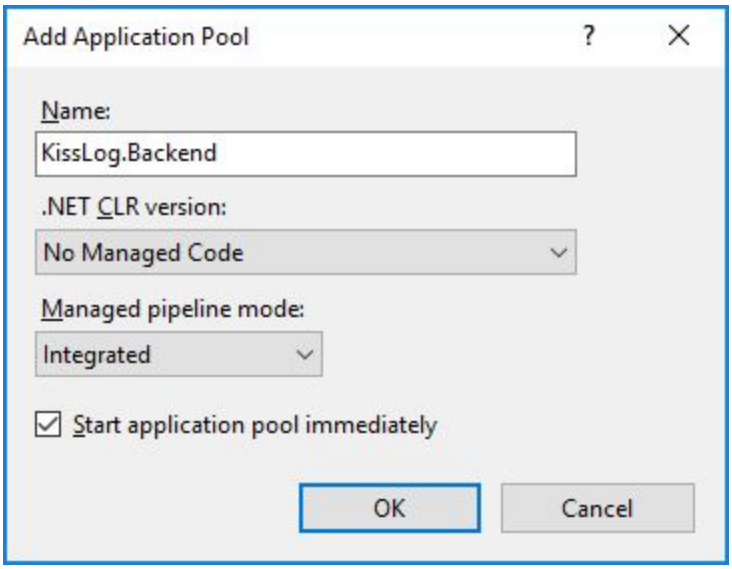
1. On the IIS server, install .NET Core 3.1 Runtime
2. Create two IIS applications, one KissLog.Backend and the second KissLog.Frontend

> **IIS write permissions**
>
> Make sure IIS has write permissions to the application folder.

3. Update the Application Pool settings for both of the applications to the following:

.NET CLR version: No Managed Code
Managed pipeline mode: Integrated

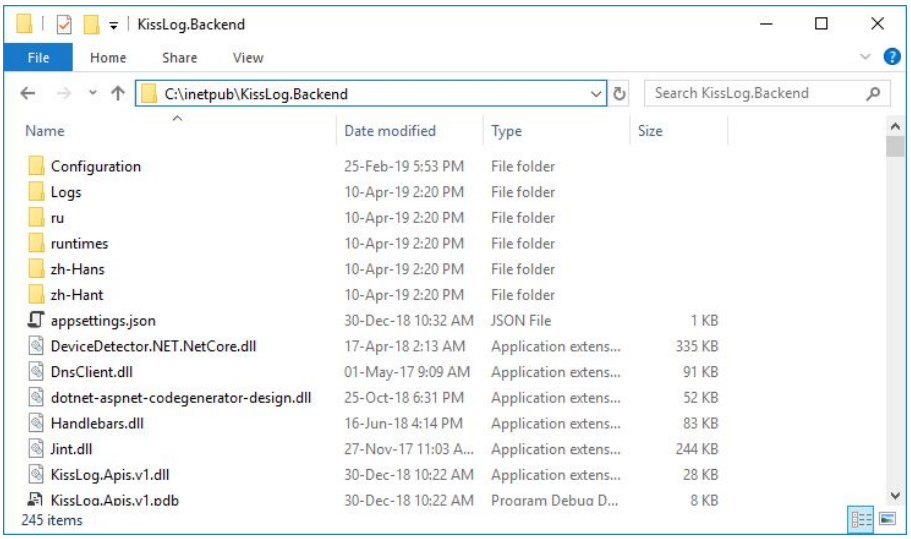
IIS Application Pool settings



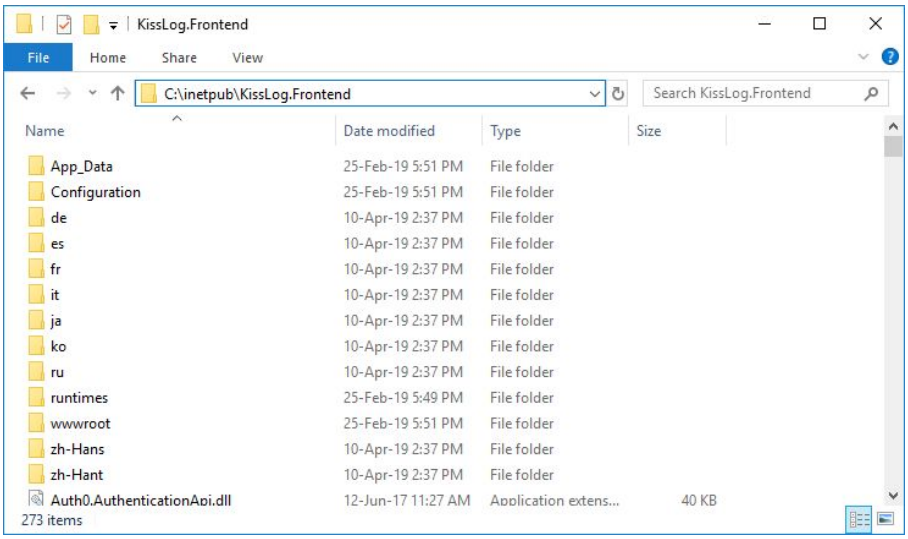
Last revision 16.07.2020

4. Copy into each Site folder the corresponding deploy packages.

KissLog.Backend IIS folder



KissLog.Frontend IIS folder



Last revision 16.07.2020

5. Update the KissLog.json file for both applications:
   KissLog.Backend\Configuration\KissLog.json
   KissLog.Frontend\Configuration\KissLog.json

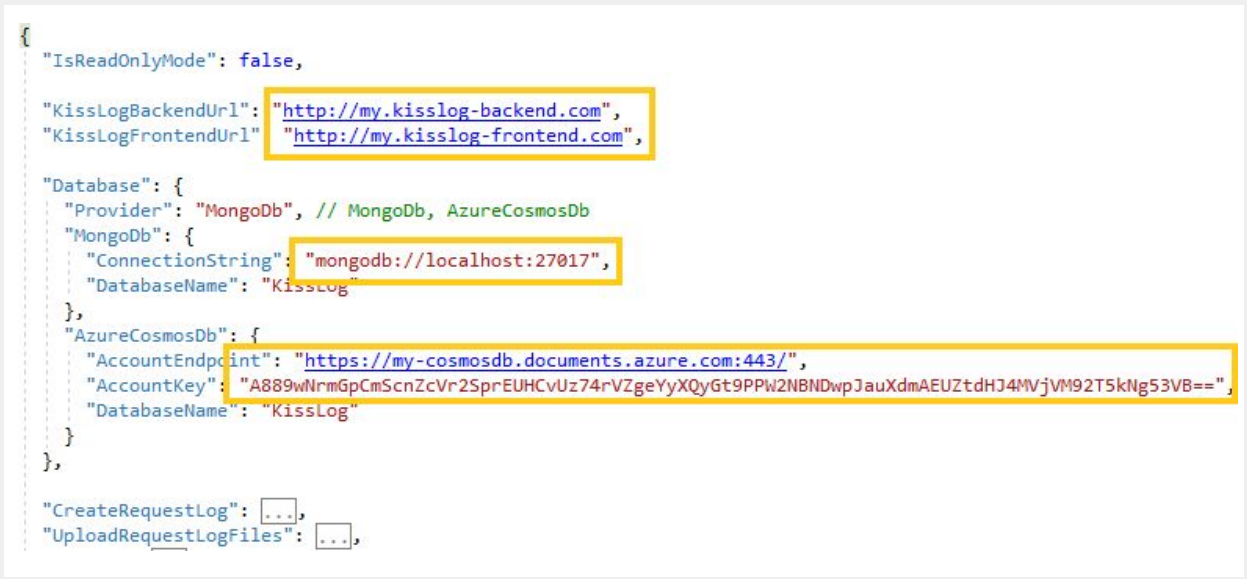
> **Configuration\KissLog.json backup**
>
> If you are updating an existing version of KissLog, make a backup of the KissLog.json file and replace it after the new files have been copied.

KissLog.Backend

Configuration

Before starting the KissLog.Backend application, you need to update the Configuration\KissLog.json file.

KissLog.Backend Configuration\KissLog.json



Replace the highlighted fields with appropriate values, based on your current setup.

- KissLogBackendUrl points to the root url of the KissLog Backend application
- KissLogFrontendUrl points to the root url of the KissLog Frontend application

- Database.Provider can one of MongoDb or AzureCosmosDb

- Database.MongoDb is required if Database.Provider = MongoDb
- Database.AzureCosmosDb is required if Database.Provider = AzureCosmosDb

Last revision 16.07.2020

Startup

After Configuration\KissLog.json file has been updated, KissLog.Backend application should be ready to use.

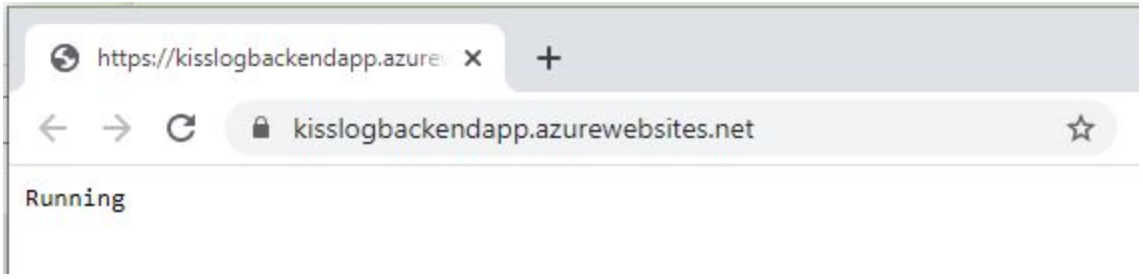
To initialize the startup process, make a single request to the application root URL ("/").

If the startup process went successful, a **200 OK "Running"** response will be returned.

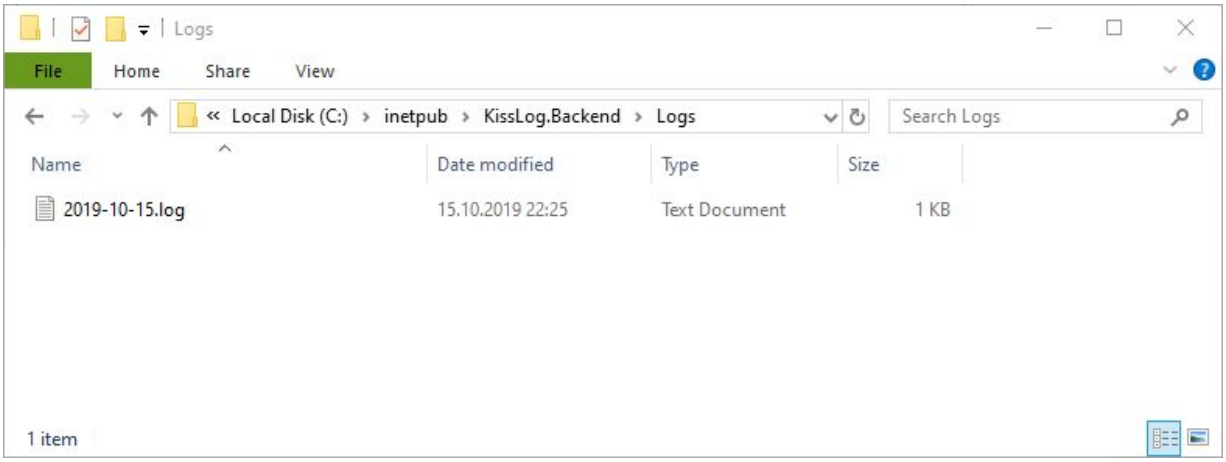
This will bootstrap all the necessary components, including configuration validation and MongoDB / CosmosDB database generation.

A **Logs** folder will be created in the application directory.

KissLog.Backend startup success



KissLog.Backend Logs



Last revision 16.07.2020

IIS write permissions

On the first run, the Logs folder will be automatically created, regardless if the startup process went successful or not.

If this folder has not been created, make sure IIS has write permissions to the application folder.

Updating KissLog.json

After a further update of the KissLog.json file, the IIS application needs to be restarted.

Last revision 16.07.2020

Troubleshooting

Log files

All the application errors are logged under the **Logs** folder.
In case of any problems, this should be the first place to check, as they will give a hint about the current issue.

IIS logs

If the application fails to start, and no logs messages can be found, activate the IIS logs.

To activate the IIS logs, edit the **web.config** and update **stdoutLogEnabled="true"**.

KissLog.Backend web.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <!-- code removed for simplicity -->
      <aspNetCore processPath="dotnet"
arguments=".\KissLog.Backend.AspNetCore.dll" stdoutLogEnabled="true"
stdoutLogFile=".\logs\stdout" />
    </system.webServer>
  </location>
</configuration>
```

After updating the web.config, restart the IIS application, initialize the Startup process and check the **Logs** folder.

Last revision 16.07.2020

KissLog.Frontend

Configuration

Before starting the KissLog.Frontend application, you need to update the Configuration\KissLog.json file.

KissLog.Frontend Configuration\KissLog.json

```
{
  "ApplicationName": "KissLog",

  "KissLogBackendUrl": "http://my.kisslog-backend.com",
  "KissLogFrontendUrl": "http://my.kisslog-frontend.com",

  "KissLogOrganizationId": "200c5a1b-5efc-48a8-8a20-5223e22a4437",
  "KissLogApplicationId": "a2f9f32b-a580-42fb-85c0-eeed4055e6fb",

  "Database": {
    "Provider": "SqlServer",  // SqlServer, MySql
    "KissLogDbContext": "Data Source=192.168.16.11;Initial Catalog=KissLog_Frontend;UID={_username_};PWD={_password_};"
  },
  "Authorization": {
    "DefaultEmailDomain": "kisslog.net",
    "HS256Secret": "_ChangeThis_4H3Q935LLRG5TEPNVUYOQPRS0SXLT3ML_",
    "SessionCookie": {
      "IsPersistentFixedValue": null,
      "ExpireInMinutes": 10080
    }
  },
  "AnonymousAccess": {
    "IsEnabled": true,
    "ValidForSeconds": 7200,
    "HS256Secret": "_ChangeThis_4H3Q935LLRG5TEPNVUYOQPRS0SXLT3ML_"
  },
```

Replace the highlighted fields with appropriate values, based on your current setup.

- KissLogBackendUrl points to the root url of the KissLog Backend application
- KissLogFrontendUrl points to the root url of the KissLog Frontend application

- Database.Provider can one of SqlServer or MySql
- Database.KissLogDbContext is the database connection string

- Authorization.HS256Secret represents the authentication JWT signature key

  In order to authenticate to this KissLog application, the user must create a JWT token which must be signed with the same key (HS256Secret) that you provide here

  The authentication JWT can be created online using https://jwt.io/

Last revision 16.07.2020

- AnonymousAccess.HS256Secret can have the same value as Authorization.HS256Secret

Startup

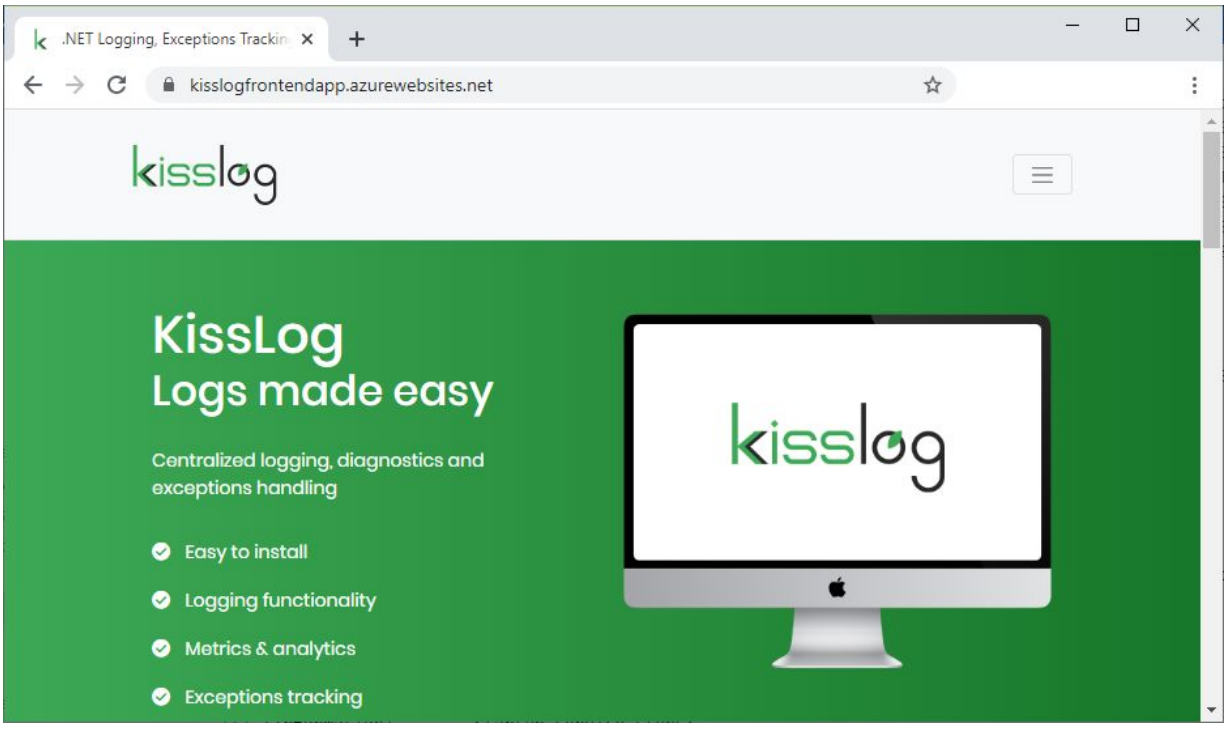
After Configuration\KissLog.json file has been updated, KissLog.Frontend application should be ready to use.

To initialize the startup process, make a single request to the application root URL ("/").

If everything went successfully, you will see the Home page.

A **Logs** folder will be created in the application physical path.

KissLog.Frontend startup success



Last revision 16.07.2020

IIS write permissions

On the first run, the Logs folder will be automatically created, regardless if the startup process went successful or not.

If this folder has not been created, make sure IIS has write permissions to the application folder.

Updating KissLog.json

After a further update of the KissLog.json file, the IIS application needs to be restarted.

Last revision 16.07.2020

Troubleshooting

Log files

All the application errors are logged under the Logs folder.
In case of any problems, this should be the first place to check, as they will give a hint about the current issue.

IIS logs

If the application fails to start, and no logs messages can be found, activate the IIS logs.

To activate the IIS logs, edit the **web.config** and update **stdoutLogEnabled="true"**.

KissLog.Frontend web.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <!-- code removed for simplicity -->
      <aspNetCore processPath="dotnet"
arguments=".\KissLog.Frontend.AspNetCore.dll" stdoutLogEnabled="true"
stdoutLogFile=".\logs\stdout" />
    </system.webServer>
  </location>
</configuration>
```

After updating the web.config, restart the IIS application, initialize the Startup process and check the **Logs** folder.

Last revision 16.07.2020