Seyyid Gökcinar
Chrischonastrasse 1
4132 Muttenz
(+41) 78 606 0469

Musikmixer, Technische Dokumentation

21. September 2020

ÜBERSICHT

Der Musikmixer wird eine Webapplikation programmiert und erstellt mi ASP.NET MVC. Das Programm soll in der Lage sein mehrere Musikstücke zusammenzusetzen und einen Musikmix zu kreieren. Der Musikmix soll die Musikstücke aneinander reihen ~~und vom aktuellen zum nächsten Stück geschmeidig übergehen (Crossfade)~~(funktion wurde wegen komplikationen gestrichen). Dafür wird hauptsächlich eine open source audio library geschrieben in C# verwendet.

Funktionen

Der User soll in der Lage sein mehrere Musikstücke auf einmal hochzuladen. Diese werden dann automatisiert vom Server verarbeitet und zurück gesendet und zum abspielen oder direkt zum herunterladen zur verfügung gestellt. Auf einer weiteren Seite kann man bereits erstellte Mixes Suchen, Abspielen oder Herunterladen.

Umsetzung der technischen Kompetenzen

Erstellt Multimedia-Inhalte dynamisch mit hilfe geeigneter Geräten, Bibliotheken, APIs selbst oder ladet Multimedia-Inhalte über ein Formular in die Applikation. Beim Upload beachtet er / sie die Filegrösse und schränkt mögliche Dateiformate ein.

Es werden Audiodateien vom User selbst hochgeladen. Die Formate werden eingeschränkt auf .mp3, .wma und .wav da diese erfolgreich getestet wurden.

Ein einfaches Formular für das Hochladen von Audiofiles:

```
<form asp-controller="Home" asp-action="MultipleFiles"
      enctype="multipart/form-data" method="post">
    <span>Titel für den Mix: </span>
    <input type="text" name="titel" required />
    <br/>
    <input type="file" multiple name="files" id="fileUpload" accept=".mp3, .wma, .wav" />
    <button type="submit">Upload</button>
</form>
```

Die Formate werden mit dem “accept” Attribut eingeschränkt auf .mp3, .wma und .wav.

Kann Multimedia-Inhalte dynamisch für unterschiedliche Geräte, Betriebssysteme und Browser optimieren. Dabei unterstützt er / sie die wichtigsten Betriebssystem und Browser.

Die Mixes sind abhörbar durch einen Audioplayer im Browser. Die Audiodateien werden als MP3 gespeichert da es von praktisch allen Browsern unterstützt wird.

Audiotag im HTML.

```
<audio controls>
   <source src="@Url.Content(@mix.MixPath)" type="audio/mpeg">
</audio>
```

Verarbeitet die generierten oder hochgeladenen Multimedia-Inhalte weiter in dem er / sie die Inhalte dynamisch verändert, kombiniert oder analysiert.

Die Mixes werden zu einem einheitlichen Format konvertiert und aneinander gehängt.

```
public void ConvertFiles(string[] files)
      {
         //Converted alle files zu .wav
         int i = 0;
         foreach (var file in files)
         {
            var infile = file;
            var outfile = _dirConverted + $"/fileconverted{i++}";
            FileInfo fileInfo = new FileInfo(infile);
            //Liest die Audiodatei mit dem Reader der alles Encoded;
            using (var reader = new MediaFoundationReader(infile))
            {
               //Schreibt das gelesene auf eine neue .wav Datei
               WaveFileWriter.CreateWaveFile(outfile, reader);
               fileInfo.Delete();
            }
         }
         string[] _files = Directory.EnumerateFiles(_dirConverted).ToArray();
         DetectBPM(_files);
      }

      public void StitchFiles(string title)
      {
         //Eröffnet einen neuen FileStream zum schreiben einer neuen Datei
         using (var output = new FileStream(Path.Combine(_dirMixes, title + ".mp3"),
FileMode.Create, FileAccess.Write))
            foreach (string file in Directory.EnumerateFiles(_dirConverted))
            {
               {
                  //Liest die Mp3 Dateien
                  Mp3FileReader reader = new Mp3FileReader(file);
                  //Setzt Id3 Tags für die Datei beim ersten durchlauf
                  if ((output.Position == 0) && (reader.Id3v2Tag != null))
                  {
                     output.Write(reader.Id3v2Tag.RawData, 0, reader.Id3v2Tag.RawData.Length);
                  }
                  Mp3Frame frame;
                  while ((frame = reader.ReadNextFrame()) != null)
                  {
                     output.Write(frame.RawData, 0, frame.RawData.Length);
                  }
               }
```

Dazwischen werden auch die BPM der einzelnen Dateien gemessen und in den Tag geschrieben.

```
public void DetectBPM(string[] files)
{
    //Detected und setzt im ID3-Tag die BPM der einzelne files
    string filename;
    double bpm;

    foreach (var file in Directory.EnumerateFiles(_dirConverted))
    {
        var fileinfo = new FileInfo(file);
        filename = fileinfo.Name;
        var outfile = _dirConverted + filename + ".mp3";
        //BPMDetector zusätzliches plugin zur Naudio Library zum messen der BPM
        BPMDetector bPMDetector = new BPMDetector(file);
        bpm = bPMDetector.getBPM();
        //Konvertiert die Dateien zu MP3 um den Tag setzten zu können
        using (var reader = new MediaFoundationReader(file))
        {
            MediaFoundationEncoder.EncodeToMp3(reader, outfile);
            fileinfo.Delete();
        }
        //Setzt die BPM in den Id3-Tag
        var tfile = TagLib.File.Create(outfile);
        tfile.Tag.BeatsPerMinute = Convert.ToUInt32(bpm);
        tfile.Save();

        Debug.WriteLine("Filename: " + outfile + ", File BPM: " + bpm);
    }
}
```

Zeigt die Multimedia-Inhalte auf unterschiedlichen Bildschirmgrössen korrekt an. (Responsive Web Design).

Das Webdesign wird mit hilfe von Bootstrap Responsive gemacht.

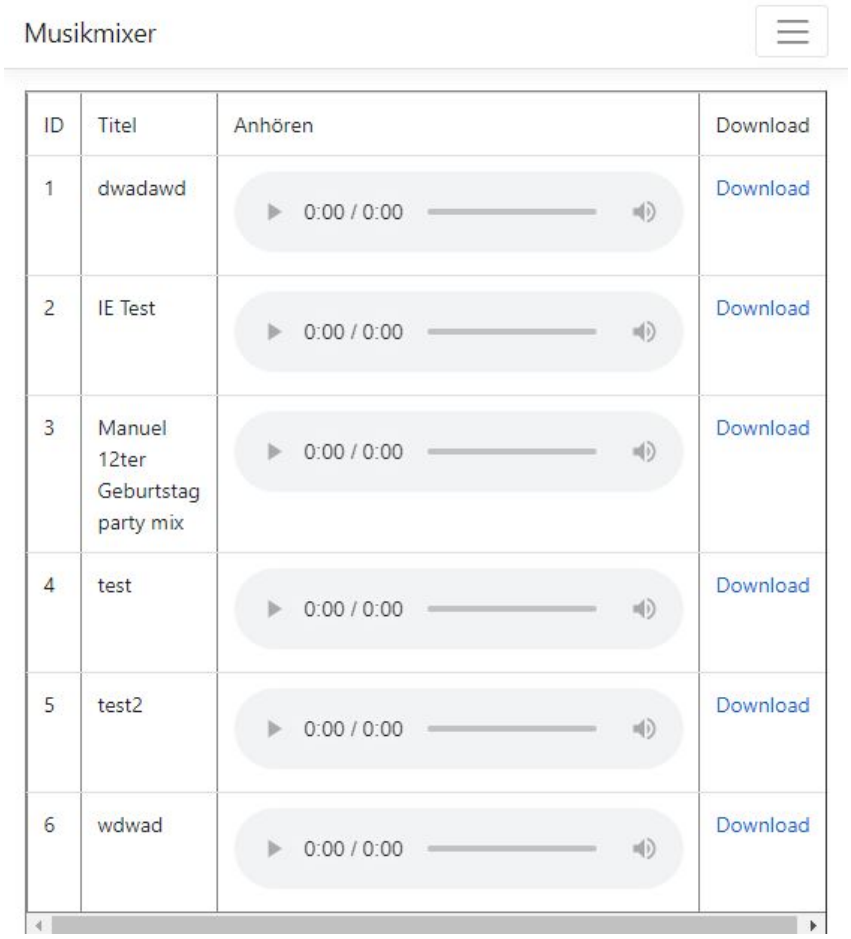
Index.html

```
<div class="container">
    <h1>Musikmixer Songs hochladen:</h1>
    <form asp-controller="Home" asp-action="MultipleFiles"
          enctype="multipart/form-data" method="post" class="flex-fill">
        <div class="form-group">
            <span>Titel für den Mix: </span>
            <input type="text" name="titel" required class="form-control" />
        </div>
        <br />
        <div class="form-group">
            <label for="fileUpload">Songs hochladen: </label>
            <input type="file" multiple name="files" required id="fileUpload"
class="form-control-file" accept=".mp3, .wma, .wav" />
        </div>
        <div class="form-group">
            <button type="submit" class="form-control">Mix erstellen</button>
        </div>
    </form>
    @Html.Label((string)ViewBag.Error)
    <audio controls class="embed-responsive-item">
        <source src="@ViewBag.Url" type="audio/mpeg">
    </audio>
</div>
```

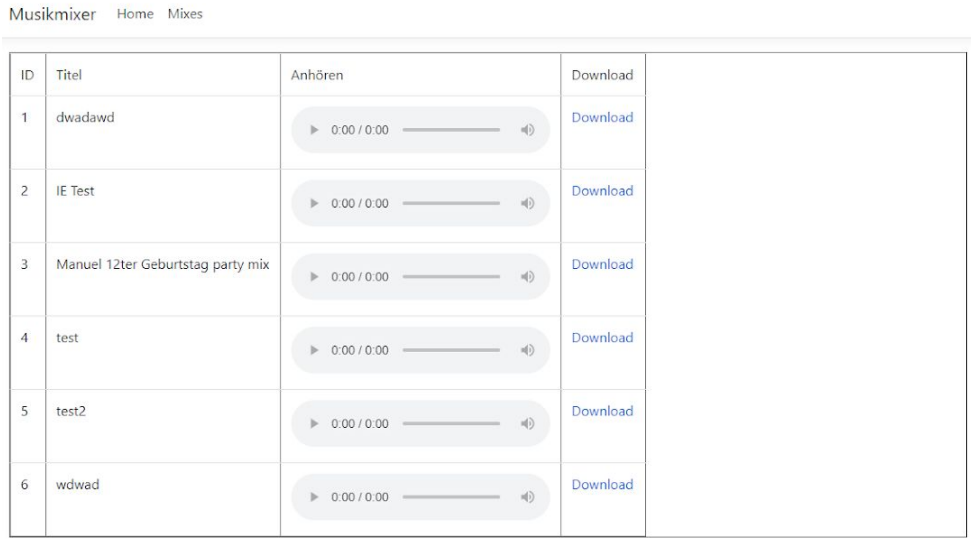
Mixes.html

```
<div class="container">
    <input type="text" id="search" placeholder="Suche" class="input-group-text" />
    <table class="table table-responsive table-hover">
        <tr>
            <td>ID</td>
            <td>Titel</td>
            <td>Anhören</td>
            <td>Download</td>
        </tr>
        @foreach (var mix in Model)
        {
            <tr id="trsearch">
                <td>@mix.MixID</td>
                <td id="tdsearch">@mix.MixTitle</td>
                <td>
                    <audio controls>
                        <source src="@Url.Content(@mix.MixPath)" type="audio/mpeg">
                    </audio>
                </td>
                <td>@Html.ActionLink("Download", "DownloadMix", new { Name = mix.MixTitle
})</td>
                @Html.Label((string)ViewBag.Error)
            </tr>
        }
    </table>
</div>
```

Mobile

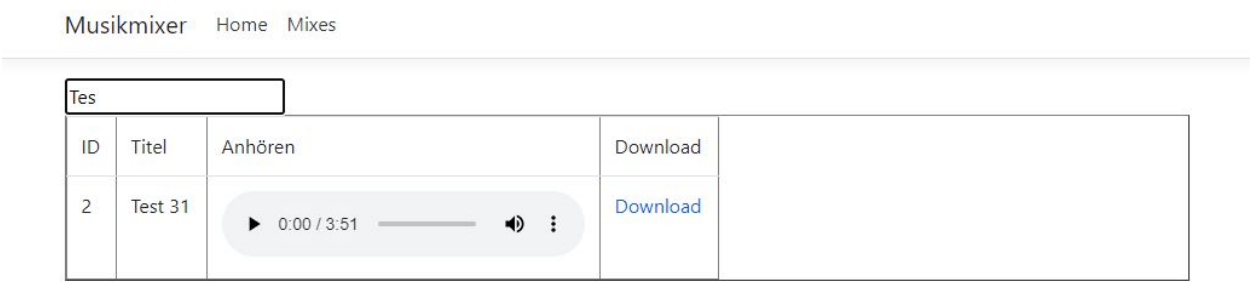


Desktop:



Achtet auf Usability der Webseite. Dazu bietet er / sie benutzerdefinierte Such- / Filterfunktionen wie Filter, Pagination oder Suche an. Leitet den Benutzer mit sinnvollen Meldungen durch alle Prozesse. Validiert HTML und CSS.

Die erstellten Mixes können auf der “Mixes” Seite gesucht werden.



Dafür wird ein Jquery Plugin verwendet das in Spezifischen HTML Elementen nach dem gleichen Text den man in der Textbox eingibt sucht und alle anderen nicht entsprechenden Elementen ausblendet.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script type="text/javascript" src="~/js/jquery.quicksearch.js"></script>
<script type="text/javascript">
    /*$(input_selector).quicksearch(elements_to_search, options;
     Die Option 'selector' spezifiziert in dem gesuchten Element*/
    $('input#search').quicksearch('table tbody tr#trsearch', {
        selector: 'td#tdsearch'
    });
</script>
```