

**Philosophische** Fakultät III

Sprach- , Literatur- und Kulturwissenschaften

Institut für Information und Medien, Sprache und Kultur (I:IMSK)  
Lehrstuhl für Medieninformatik

Digital Humanities

Modul: MEI-M 30 (M.Sc.)

WS 2014/2025

Leitung: Manuel Burghardt

Music XML Analyzer

**Projektdokumentation**

Lukas Lamm, David Lechler, Matthias Schneider, Tobias Semmelmann

1511167, 1510484, 1439829, 1509311

2. und 3. Semester M.Sc. Medieninformatik

Inhalt

[1 Zusammenfassung 4](#_Toc416170656)

[2 Setup-Anweisungen 4](#_Toc416170657)

[3 Problemstellung 5](#_Toc416170658)

[4 Projektbeschreibung 5](#_Toc416170659)

[4.1 Upload 6](#_Toc416170660)

[4.2 Statistische Analyse 7](#_Toc416170661)

[4.3 Dashboard 8](#_Toc416170662)

[4.4 Suche nach Patterns 9](#_Toc416170663)

[4.5 Suchergebnisse 10](#_Toc416170664)

[5 Architektur und Implementierung 12](#_Toc416170665)

[5.1 Projektablauf 12](#_Toc416170666)

[5.2 Frameworks und Bibliotheken 13](#_Toc416170667)

[5.3 Systemarchitektur 13](#_Toc416170668)

[6 Ausblick 15](#_Toc416170669)

[Anhang A User Stories 17](#_Toc416170670)

[Anhang B Digitaler Anhang 19](#_Toc416170671)

Abbildungen

[Abbildung 1: Startseite (Screenshot) 5](#_Toc416170672)

[Abbildung 2: Upload (Screenshot) 6](#_Toc416170673)

[Abbildung 3: Code-Beispiel (Screenshot) 7](#_Toc416170674)

[Abbildung 4: Dashboard (Screenshot) 8](#_Toc416170675)

[Abbildung 5: Search (Screenshot) 9](#_Toc416170676)

[Abbildung 6: Searchresult (Screenshot) 10](#_Toc416170677)

[Abbildung 7: Searchresult Details (Screenshot) 11](#_Toc416170678)

[Abbildung 8: Generierung von Ergebnissauschnitten(eigene Grafik) 11](#_Toc416170679)

[Abbildung 9: Systemarchitektur (eigene Grafik) 14](#_Toc416170680)

[Abbildung 10: Datenbankschema (eigene Grafik) 15](#_Toc416170681)

# Zusammenfassung

Bei einer komplexen Untersuchung von Musikstücken ist es oft unumgänglich, auch die Partitur auszuwerten. Die manuelle Analyse ist, vor allem wenn mehrere Partituren gleichzeitig untersucht werden sollen, oftmals aufwendig und mühselig. Neben kostenpflichtiger Software wie Finale[[1]](#footnote-1) und kostenloser Software wie Musescore[[2]](#footnote-2), die zum Erstellen und Bearbeiten von Partituren geeignet sind, entstehen auch einige Projekte, die das Festhalten und Bearbeiten von Noten im Browser erlauben. Beispiele hierfür sind Vexflow[[3]](#footnote-3) oder Noteflight[[4]](#footnote-4). Diesen Programmen fehlt jedoch eine umfangreiche Komponente zur Analyse und statistischen Auswertung von Musikstücken. Der Music-XML-Analyzer setzt genau hier an und bietet die Möglichkeit, Partituren im Music-XML-Format[[5]](#footnote-5) statistisch auszuwerten und in Musikstücken nach verschiedenen Pattern zu suchen.

# Setup-Anweisungen

Da es sich um eine Webanwendung handelt, sind zum Betrieb keine Installationsschritte notwendig. Die Projektseite ist unter der URL <http://music-xml-analyzer.herokuapp.com> zu erreichen. Da es sich bei der Serverumgebung um eine kostenlose *Platform as a Service* handelt gehen damit auch kleinere Einschränkungen einher. Bei längerer Inaktivität der Anwendung begibt sich die Laufzeitumgebung in einen Ruhemodus, welcher das erneute „Aufwecken“ der Anwendung erfordert. Das führt dazu, dass der Initialzugriff auf die Anwendung unter Umständen etwas länger dauern kann.

Aufgrund der Beschaffenheit des Projekts wurde die Seite lediglich für Bildschirme mit einer Mindestbreite von 1200 Pixeln optimiert. Um eine spätere Anpassung für weitere Bildschirmgrößen zu ermöglichen wurde das Twitter Bootstrap Framework eingesetzt.

# Problemstellung

Das System soll Musikinteressierte oder Forscher dabei unterstützen, Musik einerseits statistisch auszuwerten und andererseits konkrete Suchmuster in mehreren Musikstücken aufzufinden. Aktuelle Softwarelösungen wie Musescore bieten meist keine Komponenten, um diese Aufgaben zu erfüllen.

# Projektbeschreibung

Das System lässt sich grob in drei Teilbereiche unterteilen, die in den folgenden Abschnitten näher erläutert werden. Zur besseren Übersichtlichkeit werden die Teilbereiche entsprechend dem Workflow des Nutzers aufgeteilt. Im ersten Schritt erfolgt der Upload von Dateien und daraufhin die automatische, statistische Auswertung. Den letzten Schritt stellt die Suche nach Mustern in den hochgeladenen Dateien dar. Um die einzelnen Teilbereiche auch technisch voneinander zu trennen sind diese in den folgenden Kapiteln in fünf Komponenten entsprechend der technischen Zuständigkeiten aufgeteilt.

Auf der Startseite der Anwendung (Abbildung 1) bekommt der Nutzer einen kleinen Überblick über die Funktionen und kann per Klick auf den Button *START UPLOADING NOW!* mit der Arbeit beginnen.

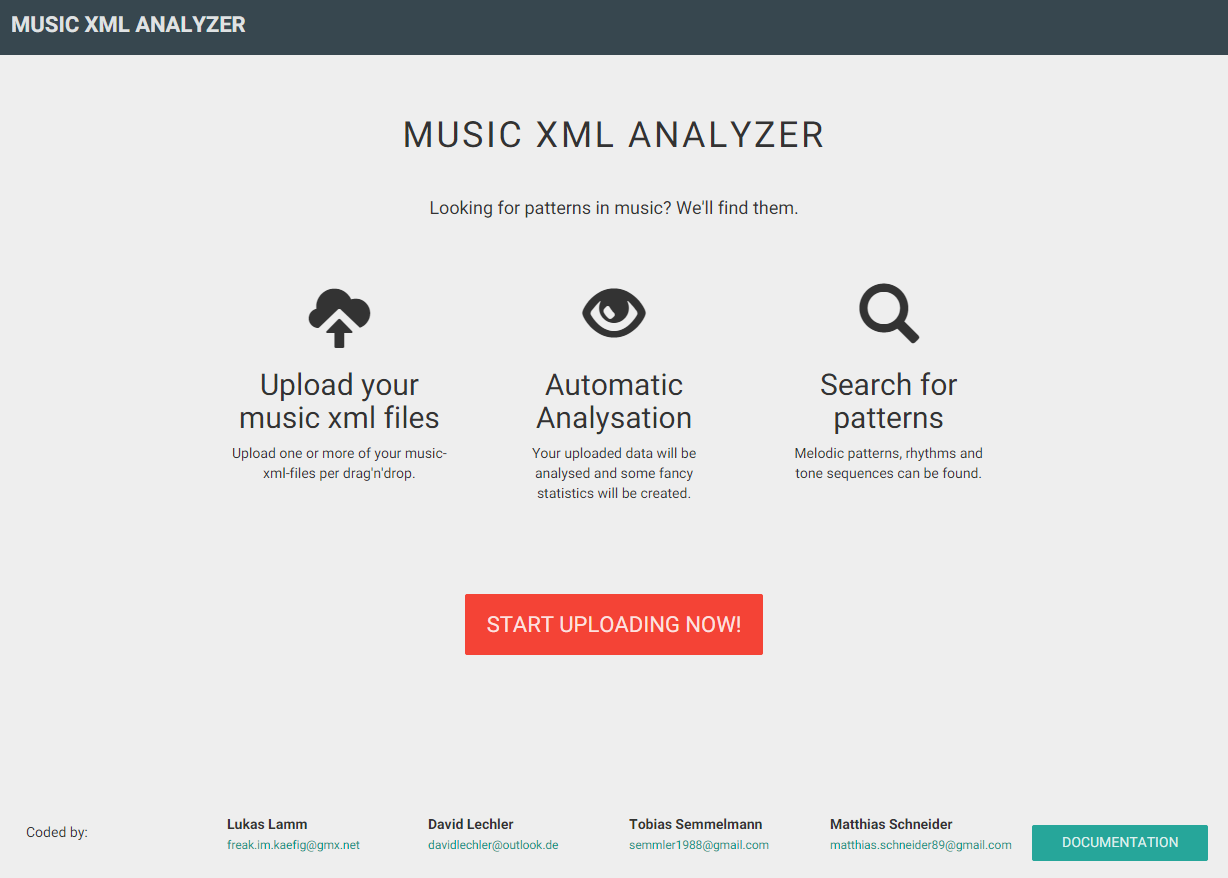


Abbildung : Startseite (Screenshot)

## Upload

Um dem Anwender die Möglichkeit der statistischen Analyse von Musikstücken zu bieten, müssen zunächst Dateien im MusicXML-Format hochgeladen werden. Das im XML-Standard realisierte, offene Austauschformat MusicXML wird von vielen Notensatzprogrammen unterstützt und löst das vorangehende Notation Interchange File Format (NIFF)[[6]](#footnote-6)ab. Zudem bieten einige Portale, wie zum Beispiel Musescore den kostenlosen Download von Partituren im MusicXML-Format an, wodurch eine Vielzahl von Musikstücken bereits verfügbar ist. Auf der Homepage von Musescore sind die Partituren in verschiedenen Formaten verfügbar und können nach kostenloser Registrierung heruntergeladen werden. Mit Hilfe der ebenfalls kostenlosen Software können die Partituren in andere Formate[[7]](#footnote-7) umgewandelt werden. Abbildung 2 zeigt die Uploadfunktion des Music-XML-Analyzer, nachdem verschiedene Dateien hochgeladen wurden. Mit einem Klick auf *ANALYZE* werden die Dateien ausgewertet.

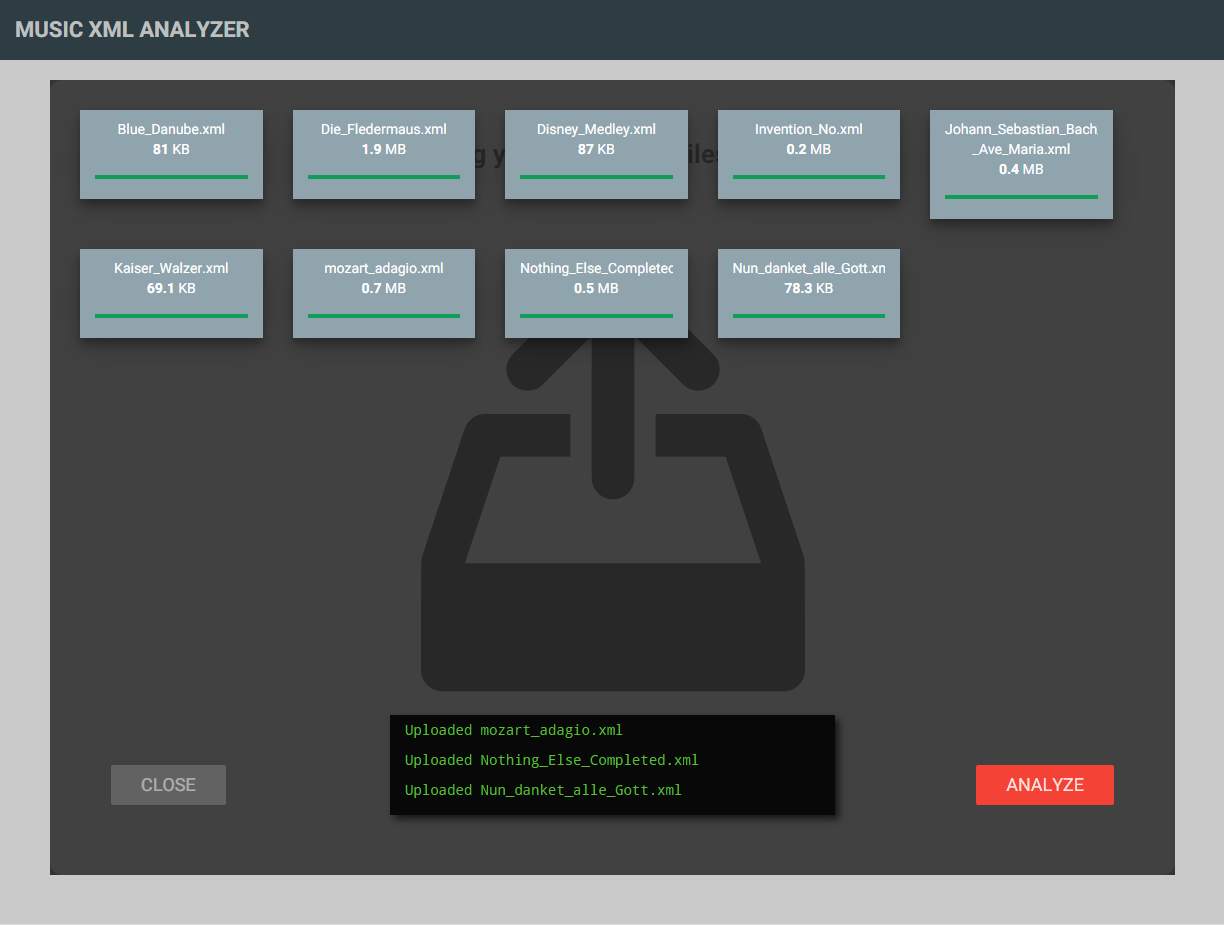


Abbildung : Upload (Screenshot)

## Statistische Analyse

Die hochgeladenen Dateien werden nach erfolgreichem Upload automatisch analysiert. Dabei werden neben den Noten und Pausen unter anderem auch die enthaltenen Intervalle, Taktarten und Instrumente gezählt.

Die serverseitige Analyse von XML Dateien beginnt stets mit einem XPath-Ausdruck[[8]](#footnote-8) gefolgt vom Parsen der resultierenden Daten mittels PHP[[9]](#footnote-9). Anhand des nachfolgenden Beispiels (Abbildung 3), soll der Ablauf und die Funktionsweise kurz erläutert werden.

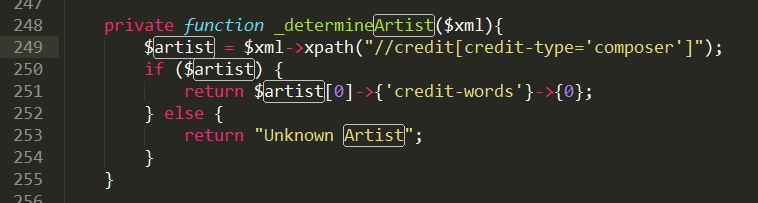


Abbildung : Code-Beispiel (Screenshot)

Für die Verarbeitung von XML bietet PHP die Schnittstelle SimpleXML[[10]](#footnote-10). Das Objekt *„$xml“* der Klasse SimpleXMLElement wurde vorab durch die Klassen-Funktion „*simplexml\_load\_file()“* mit einer XML-Datei erzeugt. Dieses Objekt ermöglicht die Verwendung der eigentlichen Xpath-Ausdrücke.

Der Name des Künstlers bzw. Komponisten eines musikalischen Werkes steckt innerhalb eines *„<credit>“ –* Elements, welches wiederum ein *„<credit-word>“* – Element mit dem Wert „composer“ enthält. Der Xpath-Ausdruck in Zeile 249 durchsucht das SimpleXMLElement (*„$xml“*) nach allen Elementen, die diese Bedingungen erfüllen. Da die Angaben zu Künstler bzw. Komponist nach MusicXML-Konvention zu Beginn einer XML-Datei gemacht werden, wird in Zeile 251 auf das erste Element zugegriffen, dass die oben genannten Bedingungen erfüllt. Der Name des Künstlers versteckt sich letztlich in einem weiteren XML-Element *„<credit-words>“*.

Obiges Beispiel veranschaulicht die Umwandlung einer XML-Datei zu einem PHP-Simple-XML-Element-Objekt, anhand dessen XPath-Ausdrücke angewandt werden können.

## Dashboard

Im Dashboard (Abbildung 4)erhält der Nutzer einen Überblick über die wichtigsten Fakten und Statistiken zu den einzelnen Musikstücken oder über den gesamten hochgeladenen Korpus. Dabei werden die Noten- und Intervallverteilung in einem Balkendiagramm, die Distribution der Tonarten, Notenlängen und Taktarten in einem Kreisdiagramm dargestellt. Daneben werden noch einige Fakten zur Anzahl und zum Auftreten der Takte, Noten, Pausen und Instrumente angegeben. Darüber hinaus bietet der Button *EXPORT AS CSV* die Möglichkeit die Daten für die einzelnen Musikstücke als CSV zu exportieren.

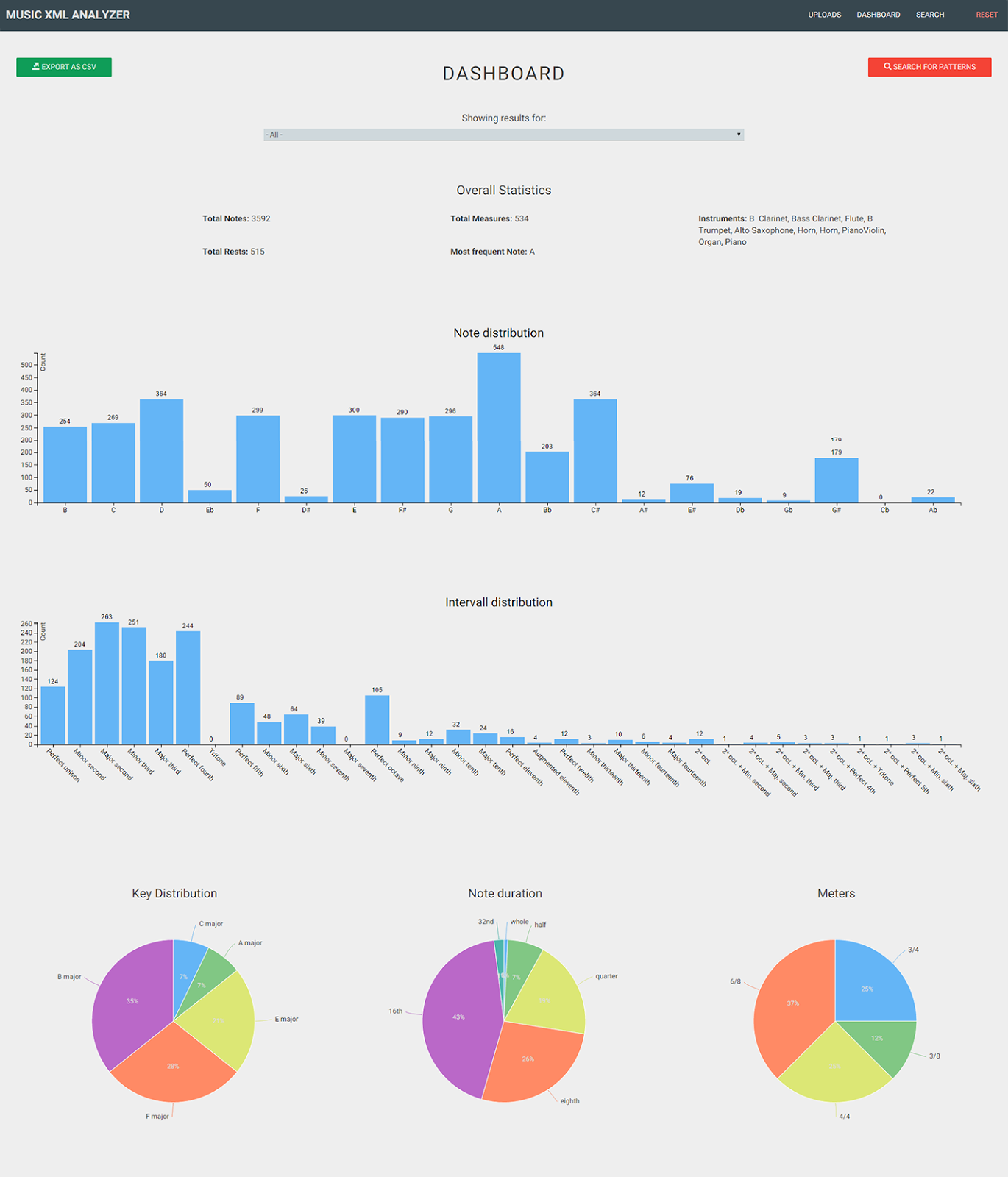


Abbildung : Dashboard (Screenshot)

## Suche nach Patterns

Vom Dashboard aus gelangt der User durch einen Klick auf den Button *SEARCH FOR PATTERNS* oder im Header via *SEARCH* zur Patternsuche. Die verschiedenen Möglichkeiten, die der Benutzer hat, um seine Musikstücke zu durchsuchen, werden im Folgenden kurz erläutert und sind in Abbildung 5 dargestellt. Als ersten Schritt muss zwischen *MELODY*, *SOUND* *SEQUENCE* oder *RHYTHM* unterschieden werden, wobei *MELODY* hier voreingestellt ist. Daraufhin kann der Benutzer entweder per Maus direkt auf den Notenzeilen, oder mit Hilfe der Buttons die gewünschten Noten eingeben. Werden die Buttons zur Noteneingabe verwendet, muss jede Note mit einem Klick auf *ADD* hinzugefügt werden. Durch Klick auf *DELETE* wird die zuletzt eingegebene Note wieder gelöscht. Alle Noten werden in der angezeigten Notenzeile dargestellt und können jederzeit abgespielt werden. Wird die Maus zur Noteneingabe verwendet, so wird standardmäßig eine Viertelnote ohne Vorzeichen an der entsprechenden Stelle eingegeben. Triolen, punktierte Noten, Vorzeichen oder eine andere Notenlänge müssen mit Hilfe der Buttons eingestellt werden. Alle Noten der dritten bis sechsten Oktave mit einer Dauer von einer Ganzen Note bis zur 64tel Note können ausgewählt werden. Auch Noten mit Kreuz- oder B-Vorzeichen können hinzugefügt werden.

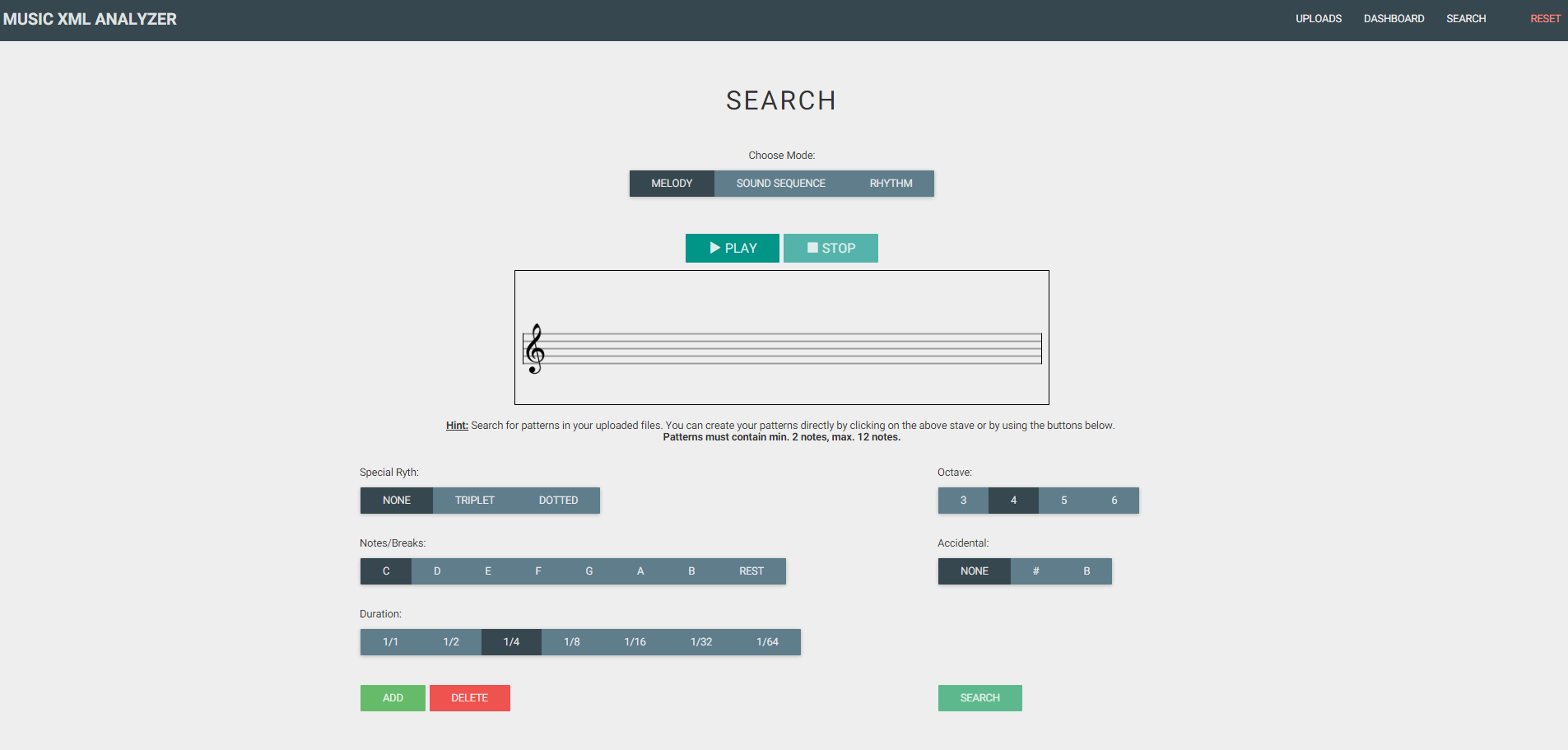


Abbildung : Search (Screenshot)

## Suchergebnisse

Ist der Benutzer mit seiner Noteneingabe zufrieden, werden durch einen Klick auf *SEARCH* alle hochgeladenen Musikstücke nach dem erstellten Pattern durchsucht. Abbildung 6 zeigt die Ergebnisseite der Suche. Hier wird im oberen Bereich das gesuchte Pattern noch einmal dargestellt und darunter die Musikstücke, welche das Pattern enthalten, samt jeweiliger Häufigkeit der Treffer.

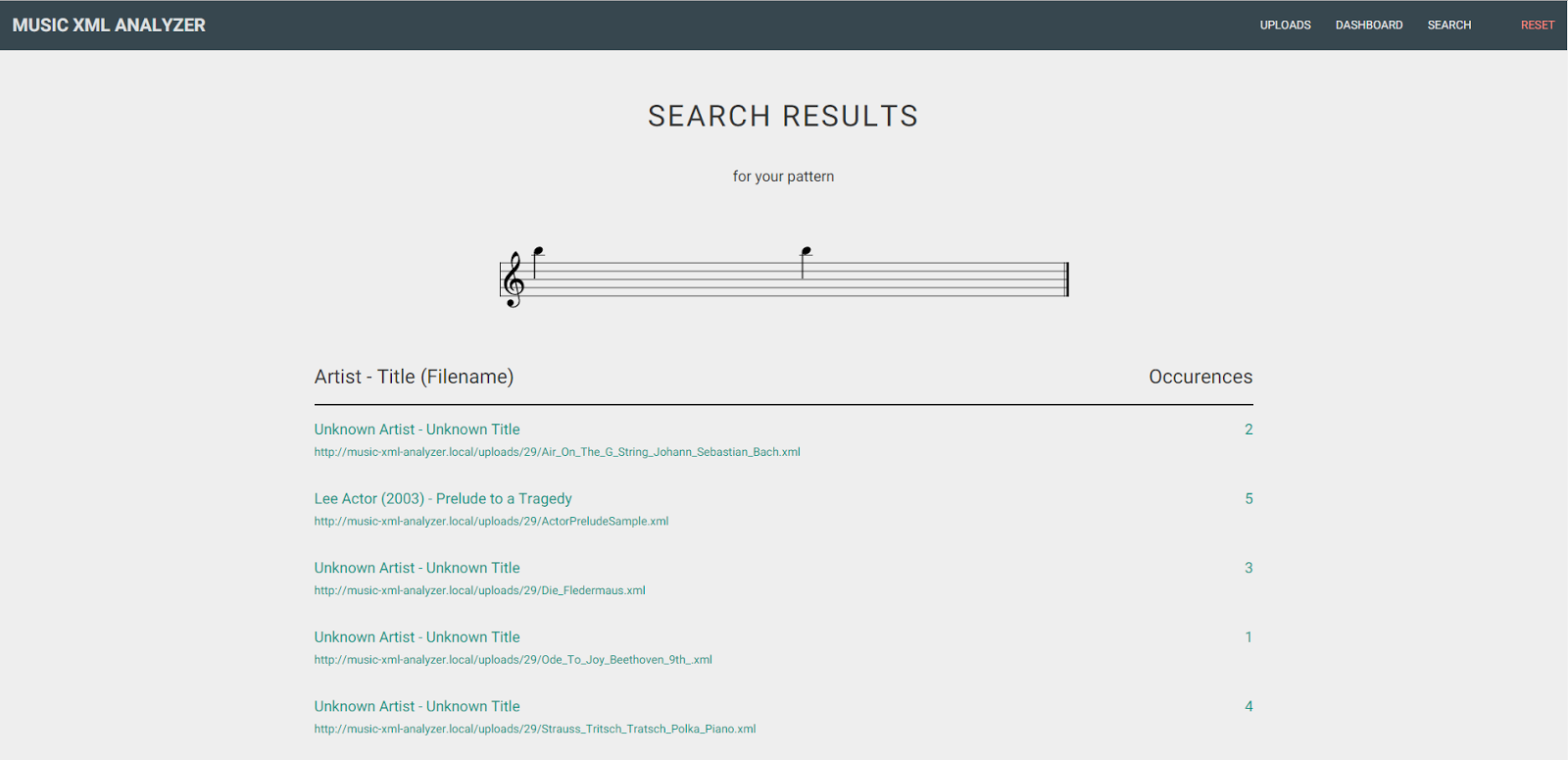


Abbildung : Searchresult (Screenshot)

Klickt der User auf einen Treffer, so wird eine Detailansicht aufgerufen, wie in Abbildung 7 erkennen ist. Hier werden neben dem Titel des Stücks die jeweiligen Trefferstellen in einem Carousel dargestellt und farblich hervorgehoben. Im unteren Bereich der Seite wird aktuell betrachtete Seite der Treffer angezeigt.

Mit Hilfe der Pfeile kann zwischen den Treffern hin und her gewechselt werden. Informationen, in welcher Stimme und in welchem Takt das Pattern gefunden wurde, werden ebenfalls dargestellt. Ein Klick auf *PLAY* spielt einen Auszug, der das Pattern enthält, ab. Der Nutzer hat außerdem die Möglichkeit, alle visualisierten Treffer als PDF-Dokument zu exportieren, was mit einem Klick auf den Button *EXPORT* *AS* *PDF* erreicht werden kann.

Abbildung 8 zeigt die dazu benötigte Komponente zur Generierung von Partituren aus MusicXML-Daten.

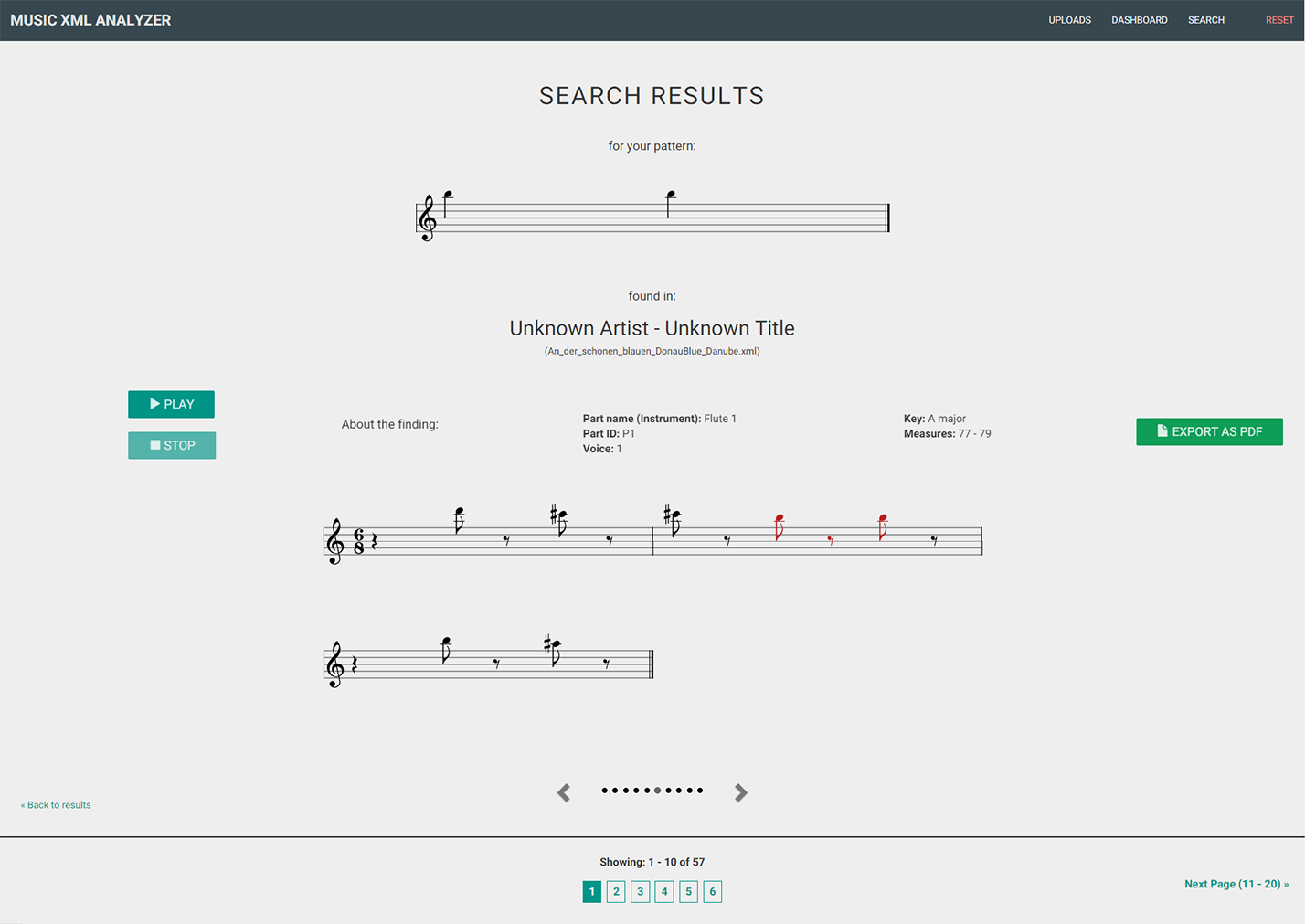


Abbildung : Searchresult Details (Screenshot)

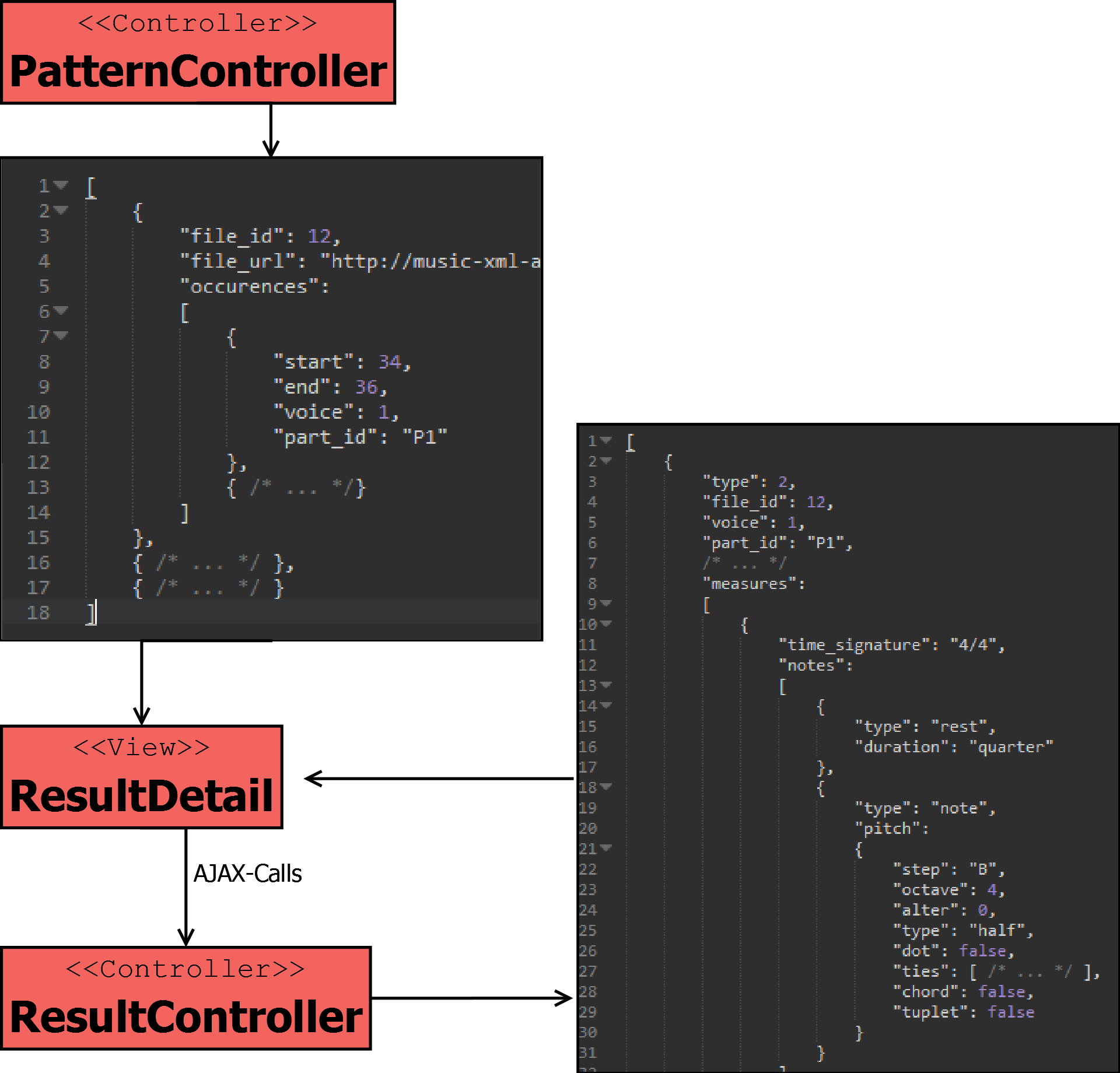


Abbildung : Generierung von Ergebnissauschnitten(eigene Grafik)

Die vom *PatternController* erhaltenen Suchergebnisse geben an, in welchem XML-Dokument Treffer gefunden wurden und bieten Aufschluss über den Index der Start- und Endnote, sowie darüber, in welchem Part und in welcher Stimme sich das Ergebnis befindet. Diese Objekte werden zunächst an den View *ResultDetail* weitergegeben. Aus Performancegründen werden von dort aus für jedes Element im oben genannten Carousel nacheinander die Ausschnitte der Partitur nachgeladen.

Im *ResultController* werden dazu mithilfe der PHP-Implementierung von *DOMDocument*[[11]](#footnote-11)und *DOMXPath*[[12]](#footnote-12) die jeweiligen Abschnitte extrahiert und das gefundene Pattern farbig markiert. Dabei besteht die Komplexität darin, dass im Grunde alle möglichen XML-Elemente des MusicXML-Dokuments richtig interpretiert werden müssen, um ein Ausgabeobjekt zu erhalten, das die Ausgabe der Noten mit der Javascript-Rendering-Bibliothek ermöglicht. Im Rahmen dieses Projekts beschränkte sich dies jedoch auf die wichtigsten Elemente zur Darstellung unterschiedlichster Notenlängen, Tonhöhen und rhythmischer Besonderheiten. Einige Spezialfälle mussten aus Zeitgründen leider Entfallen, wodurch das Ergebnis aber lediglich geringfügig beeinflusst wird.

# Architektur und Implementierung

## Projektablauf

Die Projektanforderungen wurden zunächst in Form von User Stories erfasst, welche in Anhang A User Stories einzusehen sind. Um den Projektablauf zu organisieren, wurde das Issue-System von Github benutzt. Hierbei repräsentierten die Meilensteine jeweils eine User Story. Diesen wurden anschließend kleinere Teilaufgaben, die Issues, zugewiesen. Eine zusätzliche Erweiterung für Google Chrome namens ZenHub[[13]](#footnote-13) erlaubte die Anordnung der Issues in Form eines Boards mit verschiedenen Pipelines. Das verwendete Board wurde, angelehnt an das Projektmanagement-Framework Kanban, in „Backlog“, „Nice to have“, „To-Do“, „In Progress“ und „Closed“ unterteilt.

Zu Beginn des Projekts konnten die Zuständigkeiten relativ klar aufgeteilt werden. Während Lukas Lamm für die Implementierung des verwendeten Web-Frameworks und die daran gebundene Datenbankarchitektur zuständig war, kümmerte sich Tobias Semmelmann um die Frontend-Seite des Systems und dabei unter anderem um die grafische Aufbereitung der Analyseergebnisse. David Lechler wurde mit der Traversierung, Analyse und Entwicklung der Suchalgorithmen für die MusicXML-Dateien betraut. Matthias Schneider kümmerte sich um die Eingabe der Suchpatterns, deren Visualisierung und Weitergabe an die Suchkomponente. In späteren Projektphasen verwischten die Grenzen der Zuständigkeiten immer häufiger. Dadurch übernahm Lukas Lamm auch die Darstellung der Suchergebnisdetails in Form von Takten und Noten und kümmerte sich um die beiden Export-Funktionen. David Lechler übernahm schließlich die Wiedergabe der Ergebnisse mit Midi.js.

## Frameworks und Bibliotheken

Als Grundlage der Anwendung wurde das PHP Framework Laravel[[14]](#footnote-14) in Version 4.2 verwendet, welches auf dem Model-View-Controller-Pattern basiert. Dieses bietet Möglichkeiten, das Routing zwischen den einzelnen Controllern und Views zu handhaben, sowie die Gestaltung der Views durch Templates. Zudem konnte mit den in Laravel integrierten Models das Datenbankschema in SQL und die einfache Abfrage realisiert werden.

Zur Realisation interaktiver Funktionen und Vereinfachung von Javascript-Code wurde das Framework jQuery[[15]](#footnote-15) verwendet. Zudem wird jQuery vom eingesetzten CSS-Framework Twitter Bootstrap[[16]](#footnote-16) und weiteren hier genannten Bibliotheken benötigt. Um einen modernen Look im Material Design zu erzielen, wurde zudem das Theme Material Design for Bootstrap[[17]](#footnote-17) eingebunden. Die Darstellung der Graphen erfolgt mit der Bibliothek D3.js[[18]](#footnote-18), die Ausgabe der Noten hingegen wird über die API Vexflow[[19]](#footnote-19) abgewickelt. Die Visualisierung von Statusnachrichten wird mit dem Plugin Typed.js[[20]](#footnote-20) animiert. Die Darstellung und Abwicklung von hochgeladenen Dateien erfolgt durch die Bibliothek Dropzone.js[[21]](#footnote-21) im Zusammenspiel mit dem Framework Laravel. Die Audioausgabe von eingegebenen Pattern zur Suche oder von Ergebnisausschnitten wird mit dem Framework MIDI.js[[22]](#footnote-22) abgewickelt, während die Exportfunktion der Ergebnisse als PDF mit dem Plugin jsPDF[[23]](#footnote-23) erfolgt.

## Systemarchitektur

Abbildung 9 zeigt den Aufbau des Laravel-Frameworks im Zusammenspiel mit den jeweiligen Javascript-Komponenten. Die Laravel-Komponente kümmert sich dabei um die serverseitige Logik der Anwendung, wie die Auslieferung von HTML-Seiten über eine integrierte Templating-Engine und die Kommunikation mit der Datenbank zur persistenten Speicherung. In Abbildung 9 wird auch der Aufbau nach dem Model-View-Controller-Pattern deutlich, demzufolge die grafische Repräsentation (View) von der Logik (Controller) und der Datenschicht (Model) getrennt ist. Die Javascript-Komponente der Anwendung ist ebenso nach dem MVC-Pattern aufgebaut und kümmert sich hauptsächlich um die Interaktivität der Anwendung. Beispielsweise werden die Suchmuster in Javascript vorerst clientseitig zwischengespeichert und erst beim Absenden der Suchanfrage an den Server gleitet. Weitere Aufgabengebiete sind die grafische Aufbereitung der Analysedaten und Partiturausschnitte, sowie die Wiedergabe einzelner Sequenzen.

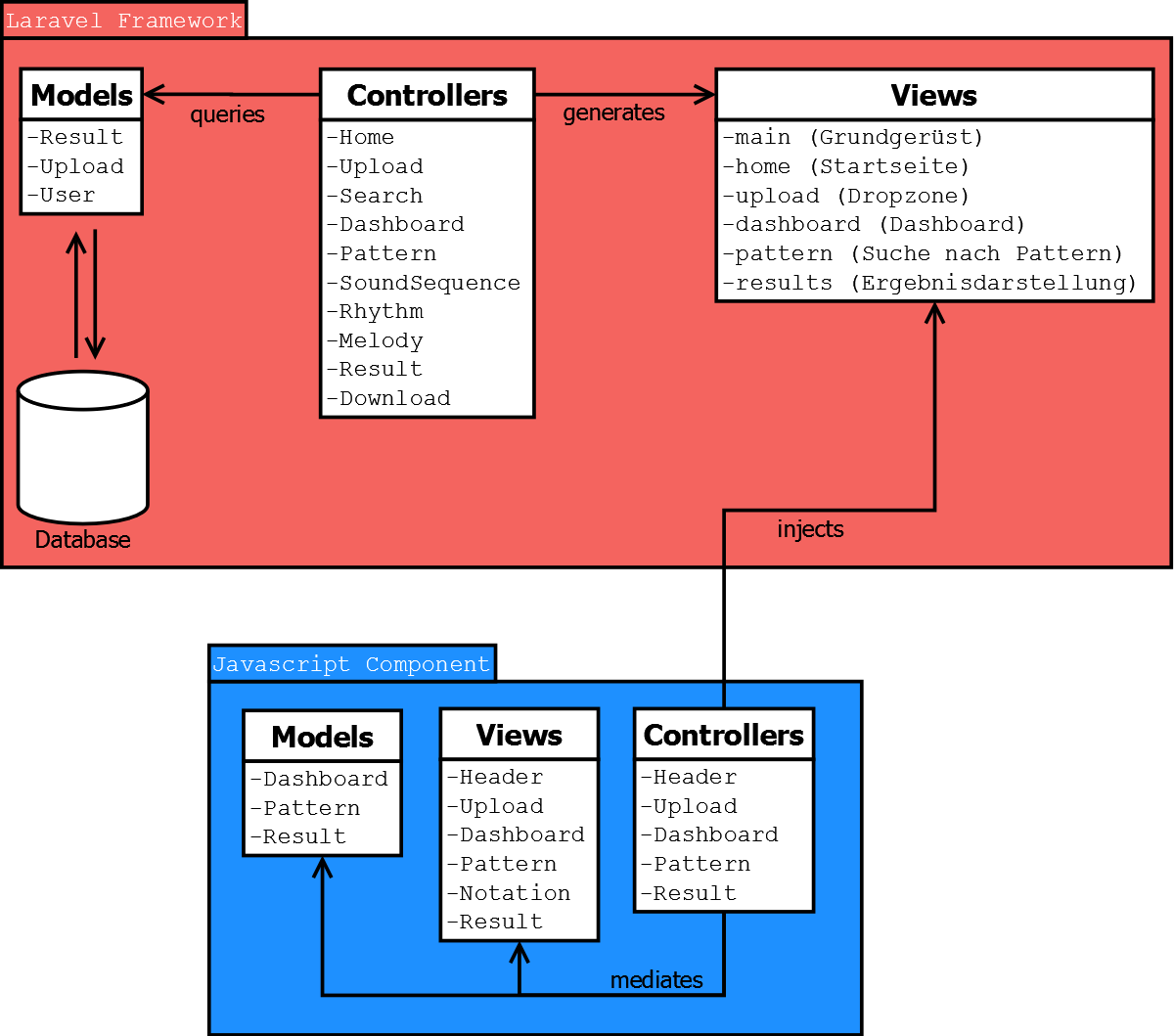


Abbildung : Systemarchitektur (eigene Grafik)

Um dem Nutzer die Möglichkeit zu bieten, seine Arbeit ohne zusätzliche Speicherung immer aktuell zu halten, wurde ein einfaches Datenbankschema entwickelt, welches in Abbildung 10 als ERM-Model aufgeführt ist. Den Nutzern wird dabei beim ersten Betreten der Anwendung eine ID zugewiesen, die gleichzeitig als Cookie beim Nutzer im Browser hinterlegt wird. Auf den Server geladene Dateien werden entsprechend mit dem Nutzer referenziert, ebenso wie einem *Upload*-Objekt nach erfolgreicher Analyse dessen Ergebnisse zugeordnet werden. Bei der Konzeption wurde bewusst auf eine aufwendige Nutzerverwaltung mit Registrierung und Anmeldung verzichtet, da dies den schnellen Einstieg oft erheblich behindert.

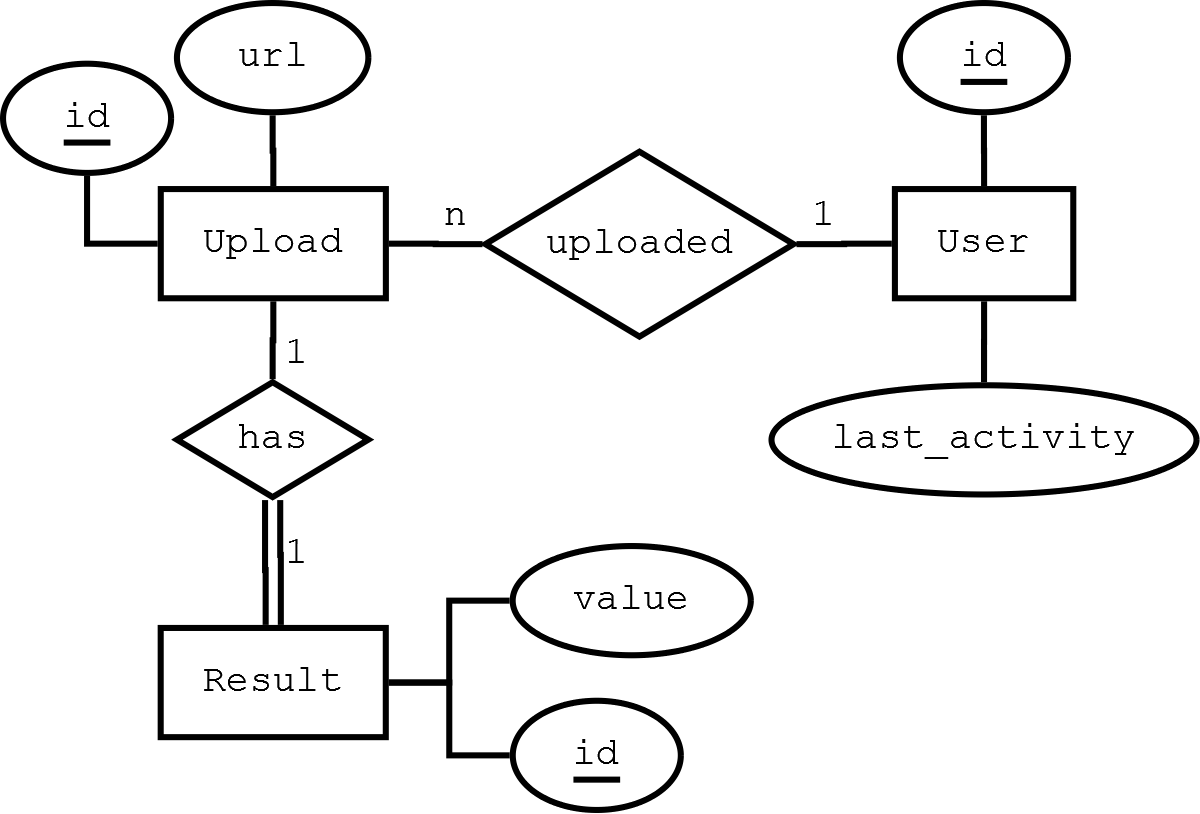


Abbildung : Datenbankschema (eigene Grafik)

# Ausblick

Im Folgenden werden einige Funktionen und Erweiterungsmöglichkeiten aufgezeigt, die nicht mehr implementiert wurden beziehungsweise als *nice-to-have* deklariert wurden. Der Grund hierfür ist, dass bei den Tests gegen Ende des Projektes noch einige Bugs gefunden wurden und die verbleibende Zeit damit verbracht wurde, diese zu beheben.

Als erste Funktion, die nicht mehr implementiert wurde ist das Abspielen einzelner, hochgeladener Musikstücke zu nennen.

Eine Erweiterungsmöglichkeit des Music-XML-Analyzer wäre, die Beschränkung auf das Format Music-XML aufzuheben, um so andere Dateitypen hochladen und analysieren zu können.

Zudem wurde bei umfangreichen Tests der Suchkomponente des Music-XML-Analyzers schnell klar, dass die aufwendige Generierung von Partituren basierend auf Daten des umfangreichen MusicXML-Formats teilweise erheblich Ressourcen benötigt. Hierbei spielt auch die Konzeption in PHP und den beiden Implementierungen zur Handhabung von xPath-Ausdrücken eine entscheidende Rolle. Eine Überlegung hierfür wäre die Realisierung einer XML-Datenbank und das dementsprechende Durchsuchen und Aufbereiten der Daten mit xQuery.

Auch die Darstellung der Suchergebnisse kann noch detaillierter gestaltet werden. Im Moment werden lediglich ein Violinschlüssel und die Notenzeile mit den Suchergebnissen angezeigt, um den Benutzer nicht zu erschlagen. Weitere Angaben wie beispielsweise Geschwindigkeit, Lautstärke oder Wiederholungszeichen können hier noch ergänzt werden. Weiterhin werden auch nur diejenigen Stimmen angezeigt, in denen ein Treffer vorhanden ist. Alle weiteren Stimmen werden aus oben genanntem Grund außen vor gelassen. Altschlüssel und Bassschlüssel werden ebenfalls nicht dargestellt, sondern in die jeweilige Darstellung im Violinschlüssel umgewandelt, also um die entsprechenden Oktaven verschoben, um die Darstellung konsistent zu halten.

Ein weiteres Feature wäre ein unscharfer Filter für alle drei Suchmöglichkeiten. Im Falle der Notenfolge (*MELODY*) beispielsweise die Möglichkeit, oktavenunabhängig nach der Notenfolge C-D-E zu suchen, um weitere Auswertungsmöglichkeiten zu erhalten.

Auch ein eigener Filter, der aus einem oder mehreren Stücken die jeweiligen Instrumente heraussucht und die Ergebnisse deutlich verfeinert, wäre als Erweiterung denkbar.

Eine letzte Erweiterung für den Music-XML-Analyzer wäre die Möglichkeit, neben Triolen und punktierten Noten auch Akkorde eingeben zu können und danach suchen zu lassen.

# Anhang A User Stories

1. „As a user i want to upload music xml files, so that i’m free to determine the data“

* Acceptance criteria:
  + visual feedback is given to the user
  + file is stored on the server

1. „As a user i want to receive immediate information (automatic analyzation) about my uploaded data, so that i get a short overview about that data“

* Acceptance criteria:
  + visual feedback while server is analyzing uploaded data
  + visualized results (s.u.)

1. „As a user i want to be able to upload multiple files at once, so that i can analyse multiple files easily“

* Acceptance criteria:
  + visual feedback while uploading
  + files are stored on the server

1. „As a user i want to be able to search for sound sequences, so that i can find occurences“

* Acceptance criteria:
  + user can enter sound sequences
  + sound sequences are visualized immediately according to user input
  + uploaded data is analysed
  + results are visualized
  + user can determine the clef

1. „As a user i want to be able to search rhytm, so that i can find rhythmic motives“

* Acceptance criteria:
  + user can defince rhytmic figures
  + uploaded data is analysed
  + results are visualized
  + user can determine the clef

1. „As a user i want to be able to search for melodies, so that i can find their occurences“

* Acceptance criteria:
  + user can define medolic patterns
  + uploaded data is analysed
  + results are visualized
  + user can determine the clef

1. „ As a user i want to be able to enter patterns interactively with a computer mouse, so that i can easily handle the creation of patterns“

* Acceptance criteria:
  + input field is visualized as a stave
  + user is able to enter notes on the stave with a computer mouse
  + user can determine the note value/duration by selection
  + user can determine the octave by placing notes on the stave
  + user can determine the clef
  + user can determine accidental and naturals
  + NICE TO HAVE: triplets, dotted notes, rests

1. „As a user i want to see a clear list of occurences of my search, so that i get an overview of where that pattern was found in my uploaded file(s)“

* Acceptance criteria:
  + user can review his/her search pattern
  + google like result list including title, interpret, number of occurences inside the file(s)

1. „As a user i want to view certain results in more detail, so that i can see where results occure in a file“

* Acceptance criteria:
  + user sees an extract of the file, containing highlighted results embedded in a few staves
  + user can jump to next/previous occurences inside the same file
  + user gets general information about this track (same as in automatic analysation after upload)

1. „As a user i want to listen to extracts of certain results, so that i can to get an impression of that certain pattern inside of its context.“

* Acceptance criteria:
  + user can play, pause, stop the playback
  + current playback position is visualized

# Anhang B Digitaler Anhang

Inhalt:

* Quellcode (Projektdateien und Ressourcen)
* Material (Während des Projektes entstandenes Material)
  + Logoentwürfe
  + Screenshots
  + Datenbankschema (Grafik)
  + Systemarchitektur (Grafik)
  + Sonstiges

1. http://www.klemm-music.de/makemusic/finale/ 27.03.2015 [↑](#footnote-ref-1)
2. https://musescore.org/de 27.03.2015 [↑](#footnote-ref-2)
3. http://my.vexflow.com/ 27.03.2015 [↑](#footnote-ref-3)
4. https://www.noteflight.com/ 27.03.2015 [↑](#footnote-ref-4)
5. http://www.musicxml.com/de/ 27.03.2015 [↑](#footnote-ref-5)
6. https://de.wikipedia.org/wiki/MusicXML 27.03.2015 [↑](#footnote-ref-6)
7. mscx, mscz, xml, mxl, mid, ogg, wav, flac, pdf, ps, png, svg und ly [↑](#footnote-ref-7)
8. http://www.w3.org/TR/xpath/ 27.03.2015 [↑](#footnote-ref-8)
9. https://php.net/ 27.03.2015 [↑](#footnote-ref-9)
10. https://php.net/manual/de/book.simplexml.php 27.03.2015 [↑](#footnote-ref-10)
11. http://php.net/manual/de/class.domdocument.php - 01.04.2015 [↑](#footnote-ref-11)
12. http://php.net/manual/de/class.domxpath.php [↑](#footnote-ref-12)
13. https://www.zenhub.io/ 27.03.2015 [↑](#footnote-ref-13)
14. http://laravel.com/ 27.03.2015 [↑](#footnote-ref-14)
15. https://jquery.com/ 27.03.2015 [↑](#footnote-ref-15)
16. http://getbootstrap.com/ 27.03.2015 [↑](#footnote-ref-16)
17. https://fezvrasta.github.io/bootstrap-material-design/ 27.03.2015 [↑](#footnote-ref-17)
18. http://d3js.org/ 27.03.2015 [↑](#footnote-ref-18)
19. http://www.vexflow.com/ 27.03.2015 [↑](#footnote-ref-19)
20. http://www.mattboldt.com/demos/typed-js/ 27.03.2015 [↑](#footnote-ref-20)
21. http://www.dropzonejs.com/ 27.03.2015 [↑](#footnote-ref-21)
22. http://mudcu.be/midi-js/ 27.03.2015 [↑](#footnote-ref-22)
23. https://parall.ax/products/jspdf 27.03.2015 [↑](#footnote-ref-23)