**Технически университет – София**

Факултет по Компютърни системи и управление



Дипломна работа

Приложение за обучение по музика

Изготвил:

**Валентин Георгиев Александров** – фак.№ 121216074

Научен ръководител:

**доц. д-р инж. Аделина Алексиева**

София

2020

Тук би трябвало да бъде място предназначено за заданието. НЕ забравяй да го поставиш. Todo

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

**ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ**

Д Е К Л А Р А Ц И Я

**з а а в т о р с т в о**

*Долуподписаният Валентин Георгиев Александров фак. № 121216074 декларирам, че представената от мен дипломна работа на тема “*Приложение за обучение по музика" *е разработена самостоятелно. В работата съм се съобразил/а с авторските права на други източници или ресурси, които съм използвал/а и съм цитирал/а в текста. Не съм използвал/а други материали, обект на авторско право, освен посочените в раздела "Използвана литература". Декларирам, че дипломната работа не е представяна в рамките на друга дипломна защита.*

*Дата: Декларатор:*

*гр. София*

**Todo: Помисли дали не можеш да разработиш второ помощно приложение с помощта, на която да може да се вкарват песни в базата данни. Идеята на това приложение да е отговора на въпроса:  
- Как се вкарват песни?**

**Недей казва за него за да се сетят да попитат за него и да се възползваш от възможността да им покажеш още нещо(печелиш време, а те губят)!**

**Съдържание: (накрая -> добави страници)**

**Увод Todo**

**Първа глава – Описание на проблемната област. Цели и задачи.**

Въведение в проблема Todo

Актуалност на проблема Todo

Текущо състояние Todo

Мотиви за разработка Todo

Цели и Задачи Todo

Todo виж дали не може да се добавят няколко глави свързани с „проучване“

**Втора глава – Проектиране на приложението.**

**Целева група** Todo

**Системни изисквания Todo**

Функционални изисквания Todo

Нефункционални изисквания

**Концептуален дизайн на системата** Todo

Софтуерна архитектура на приложението Todo

**Концептуален модел на базата данни** Todo

**Използвани данни** Todo

Достъпване на функционалностите на потребителя Todo

**Описание на използваните езици, софтуерни средства и системи** Todo

JavaScript Todo

HTML Todo

CSS Todo

NodeJS Todo

Express Todo

MySQL Todo

Проектиране на схематичен дизайн и потребителско изживяване Todo

Схематичен дизайн на приложението Todo

Проектиране на потребителския интерфейс Todo

**Трета глава – Описание на програмната реализация.**

Подход при програмната реализация на приложението

Програмна реализация на сървърната част

Етапи от реализацията на сървърната част

Описание на програмната реализация на клиентската част на приложението

**Четвърта глава – Ръководство на потребителя. Todo**

**Todo – Добави подглави за това ръководство.**

**Заключение. Todo**

**Използвана литература. Todo**

**Втора глава**

**Проектиране на приложението**

**Целева група**

Todo:

**Системни изисквания**

**Функционални изисквания**

1. Потребителите да могат да си изберат песен, която искат да научат, и приложението да им покаже как да я свирят.
2. При обучение как се свири дадена песен, потребителите да могат да наблюдават как да свирят на няколко музикални инструмента.
3. Възможността клиентите да могат да търсят песен като я изсвирят(Todo: провери сега каква част от нея трябва да изсвирят за да си подготвен на защитата).
4. Опцията клиентите да могат да създават ноти като ги изсвирят.
5. Todo: провери дали можеш да имплементираш функционалността генерираните ноти от музиката на клиента да бъдат сваляни за да може на комисията да кажем – И може клиентите да си споделят нотите със свои приятелчета.
6. Todo: АКО остане време помисли дали не можеш да направиш за десерт(не за документацията) Optical Music Recognition (at the end of the demo. But wait, there is more…) ако не са били доволни, то тази функционалност ще ги накара поне да се замислят, че приложението не е за подценяване.

**Нефункционални изисквания**

1. Клиентската част на приложението да бъде реализирана на езика JavaScript за бизнес логиката, както HTML и CSS за оформянето и стилизирането на графичния интерфейс.
2. Сървърната част на приложението да бъде реализирана чрез NodeJS.
3. Приложението да се поддържа на съвременните браузери.
4. Приложението да има удобен и приятен за ползване графичен интерфейс.
5. Сървърната част да използва Express като библиотека за реализиране на rest услуги.
6. Сървърната част да съхранява информацията за песните в база от данни MYSQL.
7. Клиентската страна да използва abcjs библиотека за да генерира ноти.
8. Клиентската страна да използва sweetalert.js за да може да разпознава нотите, когато биват свирени.
9. За стилизирането на клиентската страна да се използва библиотеката Bootstrap.

**Концептуален дизайн на системата**

**Софтуерна архитектура на приложението**

Софтуерната архитектура показва как отделните подсистеми са свързани една с друга и как си комуникират. Системата се състои от три от три подсистеми:

* Клиентска подсистема, която работи в срядата на браузера.
* Сървърна подсистема, която работи в Windows среда.
* База от данни.

Todo: Добави снимка на архитектурата – MVC client -> js + html + css, server -> nodeJS + express, database -> MySQL.

Фиг. N

На фигура N се наблюдава архитектурата на приложението. Комуникацията между клиентската страна и сървърната става посредством HTTP(Hypertext Transfer Protocol) протокола реализирана чрез REST услуги. Информацията, която сървърът обработва, се съхранява в MySQL база данни. Трите подсистеми оформят MVC(Model View Controller) архитектурата, която осигурява лесна за поддържане и ефективна система. В приложението:

* Клиентската страна(браузера) отговоря на View.
* Сървърната страна(Express + NodeJS) отговоря на Controller.
* Базата данни отговоря на Model.

**Концептуален дизайн на системата**

Базата данни съхранява необходимите данни за песни и композитори, които се използват от приложението. На фигура N е илюстрирана ER диаграма на базата данни.

Todo: Вмъкни снимка на ER диаграмата на приложението

Фигура N. Entity-relationship (ER) диаграма на базата данни.

**Използвани данни**

За да реализира функционалностите по засичане на музиката изсвирена от клиента, музикалното приложение използва микрофона на клиента за да слуша за изсвирената музика. Необходимата информация за песните, която се използва за да се реализират функционалностите на приложението, се съхранява в MySQL база данни. Сървърът е отговорен да снабди тази информация от базата данни до клиента в подходящ формат чрез REST услуги, които използват HTTP протокола.

**Достъпване на функционалностите от потребителя**

На фигура N се демонстрира как се достъпват функционалностите от клиента:

Todo: добави диаграма

Фигура N. Диаграма на работния поток на приложението.

При стартиране на приложението потребителят може да избира от меню с опции, които му позволява да използва функционалностите на приложението.

При избор на опцията за научаване на песен, клиентът ще бъде отведен до екран, в който ще бъде помолен да си избере песен от падащо меню, която иска да научи. След като цъкне напред ще бъде отведен до екран, който има ясни картинки, които показват на потребителя коя е следващата нота от песента. Картинки показват, не само коя е следващата нота, но и как да бъде изсвирена на няколко музикални инструмента. След като клиента изсвири необходимата нота, приложението ще покаже коя е следващата нота от песничката.

Когато потребителят избере опцията търсене на песен чрез свирене, ще бъде отведен до екран, на който, когато клиента цъкне върне „напред“ ще може да почне да свири песничката, за която иска да разбере как се казва и кой е нейния композитор. Приложението ще се погрижи да върне намерения резултат след като потребителя посвири известно време.

Ако клиента избере последната опция, то той ще бъде отведен до екран, където след като цъкне напред на диалоговия прозорец, ще има възможността да свири и приложението да преведе тази музика на езика на нотите, визуализирайки крайния резултат върху екрана на потребителя.

**Описание на използваните езици, софтуерни средства и системи**

За програмното реализиране на приложението ще се използват следните програмни езици, системи и софтуерни инструменти:

**JavaScript**

Todo: добави описание за JavaScript

**HTML**

Todo: добави описание

**CSS**

Todo: добави описание

**NodeJS**

Todo: добави описание

**Express**

Todo: добави описание

***MySQL***

Todo: добави описание

Todo: Може да добавиш библиотеката за генериране на ноти и библиотеката, която обработва прихванатите тонове от микрофона на клиента. При тяхното добавяне се погрижи да обновиш и съдържанието на документацията.

**Проектиране на схематичен дизайн и потребителско изживяване**

**Схематичен дизайн на приложението**

Дсфдсф

**Проектиране на потребителския интерфейс**

Дсфсдф

**Трета глава**

**Описание на програмната реализация**

**Подход при програмната реализация на приложението**

Клиентската страна на приложението е реализирана посредством езика за програмиране JavaScript, както и HTML и CSS технологиите. Приложението си комуникира със сървър посредством REST услуги, които използват HTTP протокола. За генерирането на ноти, които да се визуализират на екрана на клиента, приложението използва библиотеката abc.js. A за да може да се отчита честотата на изсвирената музика се използва библиотеката sweetalert.js.

**Програмна реализация на сървърната част**

Сървърът е разработен посредством NodeJS. Използва библиотеката Express за да осигури REST услуги, които клиентската страна ще консумира. Сървърът има задачата да поема заявки относно песните:

* Заявка за имената на песните, от които клиента може да избира да се учи да ги свири на своя музикален инструмент.
* Заявка за нотите на конкретна песен.
* Заявка, която проверява коя песен отговоря на изпратените от клиентската страна ноти и връща информация за конкретната песен, като име на песента, нейния композитор и снимка на композитора й.

Тази система освен, че си комуникира с клиентското приложение, си комуникира и с MySQL база данни, където съхранява данните за различните песни и техните композитори.

Сървърът играе ролята на мост между базата данни и клиентското приложение, добавяйки възможността данните да бъдат валидирани и анализирани, както и имплементира алгоритъма за търсене на песен, спрямо това, което потребителя е изсвирил.

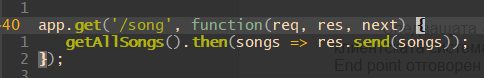
**Етапи от реализацията на сървърната част**

**Клониране на проект шаблон за NodeJS сървър**

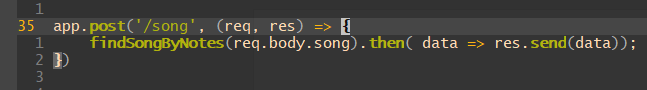
Open Source общността на JavaScript е огромна и осигурява много инструменти, с които да помогне при разработката на JavaScript приложения. В GitHub следния шаблонен проект node-mysql-express-template-v1 представлява проект, който е конфигуриран да работи като REST сървър с готова конфигурация за да се свърже към MySQL база данни.

**Създаване на маршрути**

Следващата стъпка е да се направят end points, които ще се използват от клиентската система.



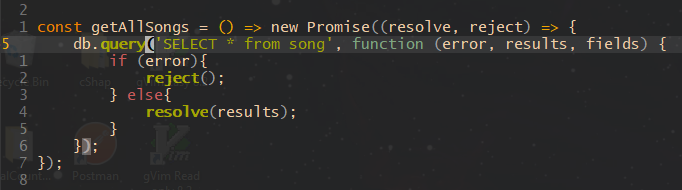
Фигура N. End point отговорен за връщането на имената на наличните песни



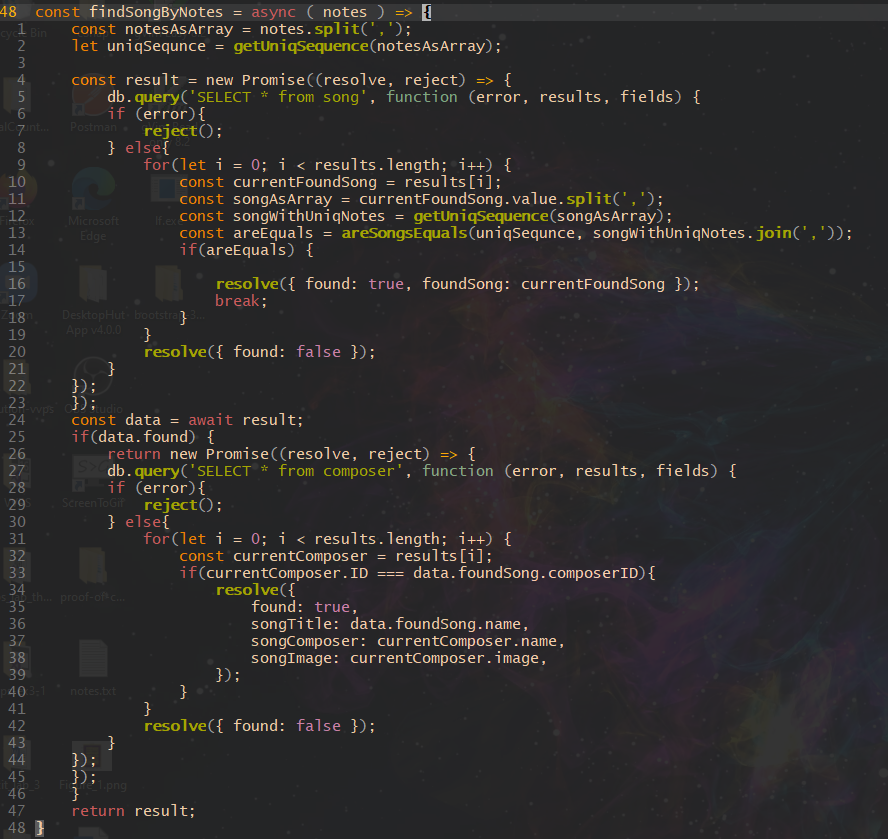
Фигура N. End point отговорен да приеме изсвирената песен от потребителя и да провери на коя песен отговаря песента, която клиента е изсвирил.

**Създаване на услуги**

Създаване на услуги, които да направят връзката към базата данни и да върнат обработена информацията.

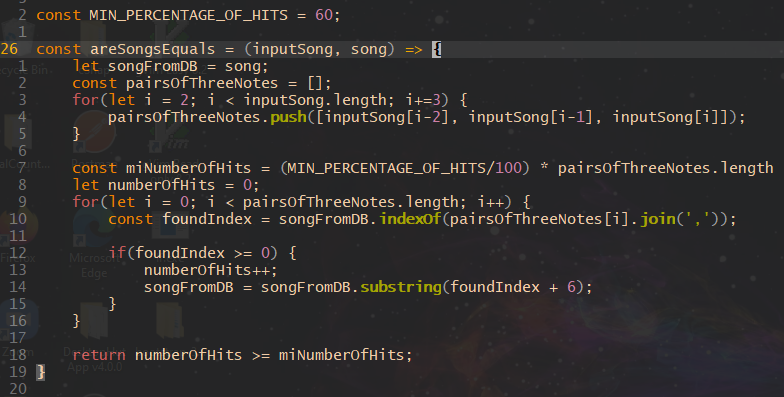


Фигура N. Услуга, която ще върне наличните песни.



Фигура N. Услуга, която ще провери дали изсвирената музика от потребителя съвпада с някоя песен от наличните в базата данни.

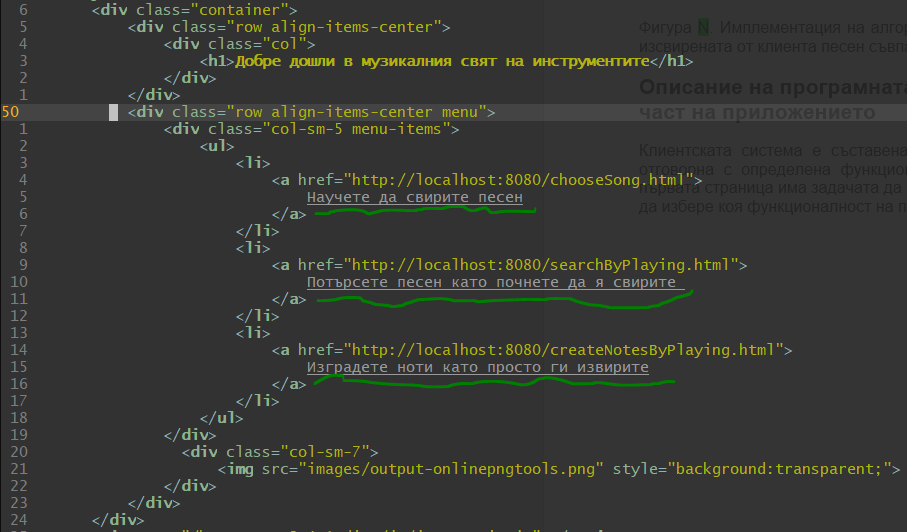
**Имплементиране на алгоритъм за търсене на песни**



Фигура N. Имплементация на алгоритъма, който ще се погрижи да провери дали изсвирената от клиента песен съвпада с песен от базата данни.

**Описание на програмната реализация на клиентската част на приложението**

Клиентската система е съставена от няколко на брой HTML страници, всяка отговорна с определена функционалност. При стартирането на приложението първата страница има задачата да визуализира меню, което позволява на клиента да избере коя функционалност на приложението иска да използва.



Фигура N. Навигацията на приложението за основните функционалности.

Приложението е съставено от 3 основни функционалности:

**Научаване на нова песен**

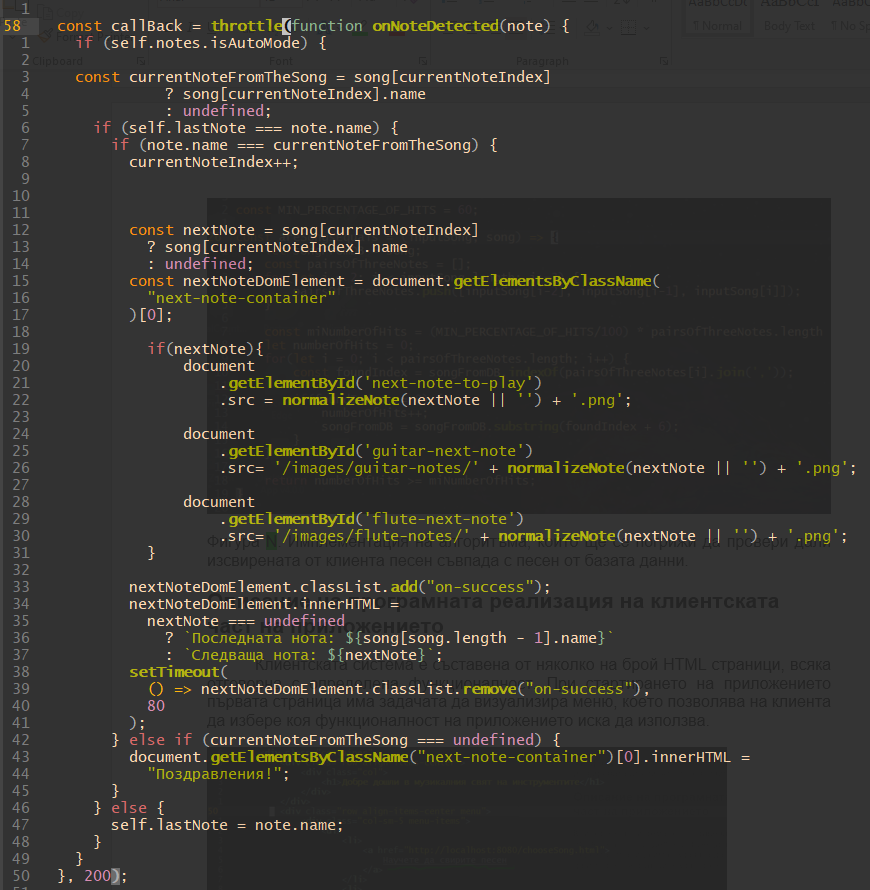
Потребителя разполага с възможността от падащо меню да си избере песен, която би искал да науча как да я свири на своя любим музикален инструмент. След избора се появява екран, който демонстрира как свири нота по нота избраната песен. Появяват се снимки на няколко музикални инструменти, които показват как да се изсвири необходимата нота.

Когато клиента изсвири дадена нота, приложението третира тази операция като събитие.



Фигура N. Илюстрация на събитието, което настъпва при засичането на тонове.

На фигура N се илюстрира как се подава callBack функция, която ще се погрижи да обработи засечената честота от музикалния инструмент. На фигура Р може да се види имплементацията на тази callBack функция:

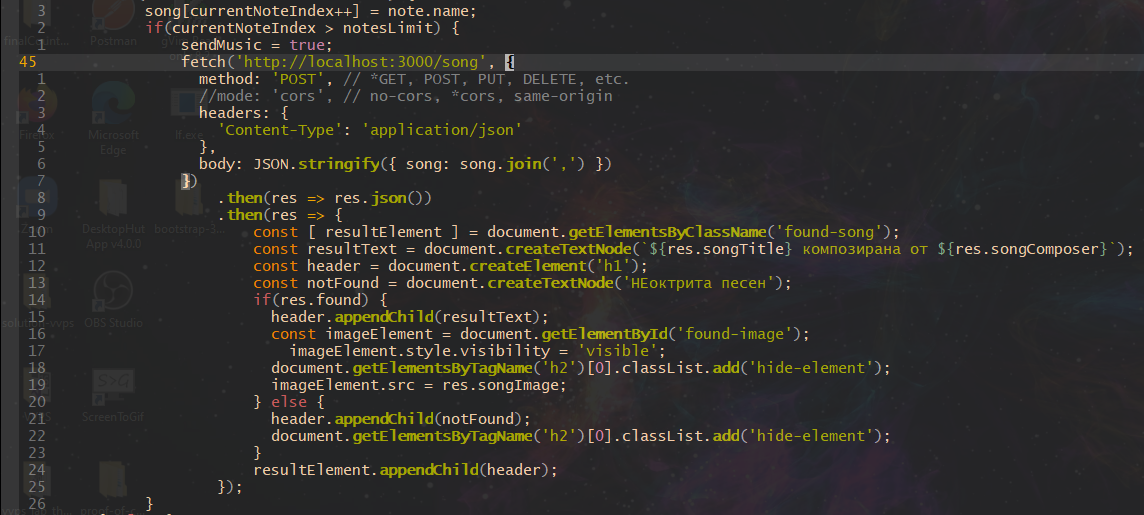


Фигура P. Илюстрация на имплементацията на callBack функцията.

За да се избегне засичането на една и съща нота многократно се използва throttle функция, която ще се погрижи callBack функцията да не се извика отново за по-малко от 200 милисекунди. Тази комбинация работи доста оптимално за нуждите на приложението. За да се прочете правилната нота, се прави проверка нотата да съвпада с предишната. Това гарантира по-точно засичане на изсвирения тон, като се пренебрегне „шума“. Мелодията на песента, която иска да бъде научена от потребителя се съхранява в масив. При съвпадение на поредната нота от песента, с това, което свири клиента, ще бъде изчислена следващата нота от песента. Когато това се случи, ще се променят илюстрациите на снимките, които показват как се свири на множество от музикални инструменти, както и за момент ще се изпълни анимация, която ще покаже на клиента, че правилно е изсвирил необходимия тон. Това се постига чрез добавяне и премахване на класове, които се закачат към елементи от HTML частта. Когато нотите от песента приключат, клиентът ще бъде поздравен за успешно изсвирената песен.

**Търсене на песен, чрез нейното изсвирване**

Тази функционалност се реализира като преизползва засичането на тонове от горната функционалност. Този път, засечените ноти се добавят в масив, който ще бъде запратен към сървъра за обработка и намиране на съответна песен. На илюстрация N може да се види, как се изпраща заявка към сървъра, когато потребителя е изсвирил достатъчно на брой ноти:

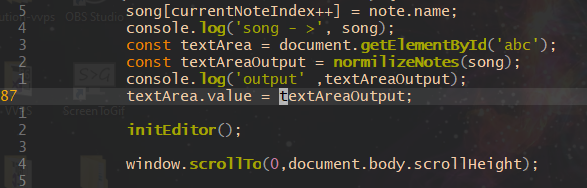


Фигура N. Илюстрация на бизнес логиката по запращането на изсвирената музика до сървъра.

На маркирания ред от фигура N може да се наблюдава пускането на заявка към сървъра. В тялото на заявката се изпраща изсвирената от клиента музика. Приложението ще се погрижи да извлече информацията от отговора от сървъра и съответно да визуализира на екрана името на песента, нейния композитор, както и снимка на композитора. В случай, че песента не е открита, клиента ще бъде информиран за това с кратко съобщение.

**Генериране на ноти чрез тяхното изсвирване**

Последната функционалност на приложението също преизползва техниката за разпознаване на изсвирените тонове от първата функционалност. Приложението държи всеки изсвирен тон в масив. Този масив бива трансформиран във формат, който да се разбере от abc.js.



Фигура N. Трансформиране на засечените тонове към ноти, които се визуализират на екрана.

На маркирания ред от фигура N е илюстрирано как трансформираните засечени тонове се присвоят на Dom елемент, който е невидим за потребителя, но се използва от abc.js за да получи новите ноти. initEditor() функцията ще се погрижи да прересува визуализираните ноти. Последния ред от фигура N ще се погрижи клиента да вижда последните ноти, които е добавил, в случай че песента стане по-голяма от размерите на екрана. normilizeNotes е функция, която ще трансформира засечените тонове във формат, който да бъде разбираем от библиотеката отговорна за визуализирането на нотите върху дисплея на потребителя.