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

Съдържание

Увод

За какво служи изкуството? Според писателката Елън Дисаянаке, негова основна характеристика е предоставянето на механизъм за създаване на обекти или събития, "съществуващи в измерение, различно от ежедневното".
Неслучайно музиката се нарича език на душата. За огромна част от човечеството тя (в много бройните си жанрове и стилове) е най-достъпната и разбираема форма на изкуство и забавление от векове насам [7].

Как ни влияе музиката? [8] Като едно от седемте изкуства, тя върви ръка за ръка с развитието на човечеството. Може да се приеме
едновременно като:

- специален начин за възприятие на света около нас
- уникална демонстрация на креативност
- висша форма на комуникация

Някои мислители отчитат общи ценности в музиката:

- израз на емоция
- нуждата от преизграждане на реалността
- споделяне с близки
- духовна удовлетвореност
- съхранение на култура във времето

Първа глава

Описание на проблемната област.

Цели и задачи.

Въведение в проблема

"Колкото повече е дадено на човек, толкова повече трябва да се труди." - Пьотр Чайковски [9]

За всеки начинаещ музикант изборът и упражнението на инструмент може да се стори като трудна, а понякога и привидно непосилна задача. Най-честите пречки са:

- Липса на правилни подход и/или средства

Малкото време и пари са водеща причина за демотивация още в самото начало на начинанието.

Друг източник е сблъсъкът с лош учител - заради него може коренно да се промени начинът, по който учещият се възприема като музикант, и тази преценка трудно се изменя.

- Кариера, семейни отговорности и социални задължения

Отдадеността към нещо ново винаги е предизвикателство, изискващо адаптация и приоритизация.

- Недостатъчно упражнения

Не съществува изобретателно занимание, което не ги изисква. Свиренето на инструмент е от най-видните примери за това.

- Нереалистични очаквания

Нагласата и целта са важна част от процеса на учене. Понякога, за съжаление, в стремежа си да достигне някакво ниво на познаване, нетърпението може да надделее, а и отчае начинаещия [10],[11].

Актуалност на проблема

Както при всяко хоби, причината за началото му пряко засяга посоката на развитие на бъдещия музикант, дали като любител или професионалист. Достъпът до информация никога не е бил толкова обширен, както сега. Следователно възможностите за научаване на нещо ново са безброй. Една от целите на голяма част от технологиите е спестяването на време и усилия, които биха могли да се инвестират в по-полезни действия.

Текущо състояние

Разнообразната гама от приложения предоставя множество възможности за учене и затвърждаване на знания от основна музикална теория до композиция на цели симфонии.

Алтернативи: [12]

1. Simply Piano

Целта на приложението е даването на обратна връзка. Осигурява и 5-минутни тренировки, съвети за четене на ноти и широко разнообразие от песни. Разработеното приложение в тази дипломна работа надгражда тази идея, като позволява потребителя да се обучава на множество инструменти, като му се показва как да свири на тях.

2. Yousician

Като най-известното приложение в тази сфера, Yousician е разработен за научаването на пиано, китара и укулеле. Включени са кратки видеа преди всеки урок, както и седмични предизвикателства.

3. Shazam

Представлява приложение за търсене на песни след като са изсвирени. Системата разработена в тази дипломна работа има почти същата функционалност, но е предвидена да засича песни, които се свирят от начинаещи и не е спазено напълно темпото. Това стимулира клиентите да разчитат повече на своите умения и слух, като се пробват да възпроизведат до колкото могат песента, която искат да открият.

4. Perfect Ear

Това приложение има за цел да тренира музикалния слух и чувството за ритъм на потребителя - важни умения за всеки музикант [8].

Мотиви за разработка

Проблемът при много начинаещи музиканти главно идва от незнанието и липсата на насоки по отношение на учене. Светът на музиката е голям и динамичен, затова не е необичайно да присъстват страх и притеснение в избора на инструмент и музикалната теория касаеща го. Целта на приложението е да ги обедини в едно така, че потребителят да добие представа какво се случва зад процеса на свирене и да изгради усет за различните мелодии.

Цели и Задачи

Целта на дипломната работа е да се работа е да се разработи приложение, което да помогне на хората да навлязат в музиката по-лесно по естествения начин. Сред задачите, които трябва да се направят е изследване към коя общност от хора това приложение ще бъде най-полезно и до каква степен може да им помогне в изграждането на основни музикални познания при свиренето на даден музикален инструмент. Трябва да се провери на пазара какво предлага конкуренцията, за да може приложението да надгради това, което вече е достъпно и да бъде максимално полезно за клиентите.

Проучване за желанието от страна на хората да научат музикален инструмент

Проучвания сочат, че 90 процента от децата имат желание да научат музикален инструмент. Изглежда, че интересът е в своя пик при малките децата. С порастването интереса им почва да намалява. Проучването е проведено на територията на Великобритания с деца във възрастовата група между 6 и 16 години. Данните сочат, че във възрастовата група между 10 и 14 години е намален броя на деца, които искат да научат музикален инструмент[13]. Анализирайки събраната информация от проучването изглежда, че приложение, което да помогне на начинаещи, особено на по-малка възраст би играло ключова роля в това повече деца да се научат да свирят на музикален инструмент.

Целева група

Целевата група на приложението е хора, които са начинаещи и искат да се научат да свирят на музикален инструмент и да се запознаят по-добре с чара и красотата на музикалното изкуство. Приложението е полезно и за напреднали музиканти, които искат да научат нов музикален инструмент. Това приложение също така е и подходящо за хора, които искат просто да се запознаят малко по-добре с музика (да видят какво представлява свиренето на музика). Приложението може и да се използва и от ученици, които изучават музика като допълнителен инструмент за усвояването на умението да свирят и да разбират от музика.

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.
4. Опцията клиентите да могат да създават ноти като ги изсвирят.
5. Клиентът да може да сваля генерираните ноти.

Потребителски истории

номер	като	искам да	за да
1	потребител	Да ми се показва как да изсвиря песен по избор	Мога да я науча
2	потребител	Да свиря някаква песен като цъкна търсачката	Мога да разбера как се казва песента, която свиря, кой е нейния композитор и как изглежда.
3	потребител	Да мога да създавам ноти като свиря	За да се науча кои тонове на кои ноти отговарят.
4	Потребител	Да има опцията за сваляне на генерираните ноти	За да мога да си ги запаметя и да ги споделям със своите приятели.
5	потребител	Получава наличните песни	За да мога да избирам коя искам да науча.

Фигура 0. Таблица с потребителските истории.

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 tuner за да може да разпознава нотите, когато биват свирени.
9. Клиентската страна да използва dom-to-image за конвертиране на генерираните ноти в снимка.
10. За стилизирането на клиентската страна да се използва библиотеката Bootstrap.

Втора глава

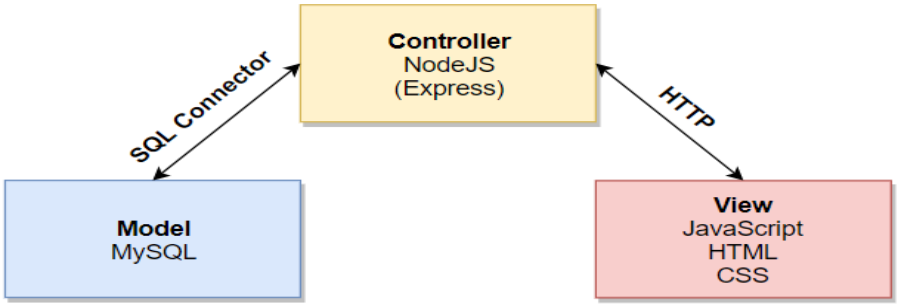
Проектиране на приложението

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.



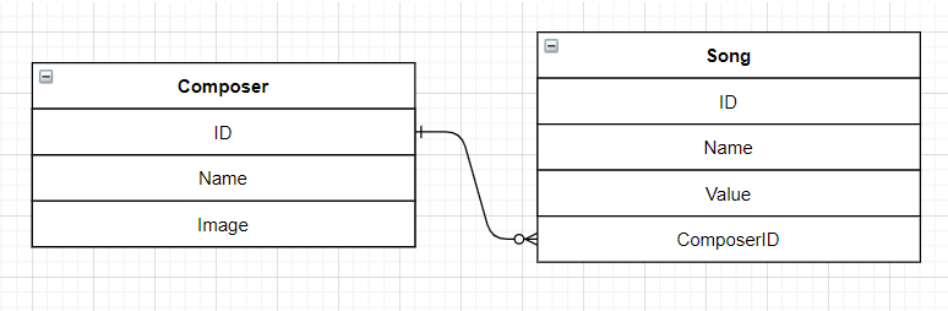
Фигура 1. Илюстрация на Софтуерната архитектура на приложението

На фигура 1 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 модел на базата данни

Базата данни съхранява необходимите данни за песни и композитори, които се използват от приложението. На фигура 2 е илюстрирана ER диаграма на базата данни.



Фигура 2. Entity-relationship (ER) диаграма на базата данни.

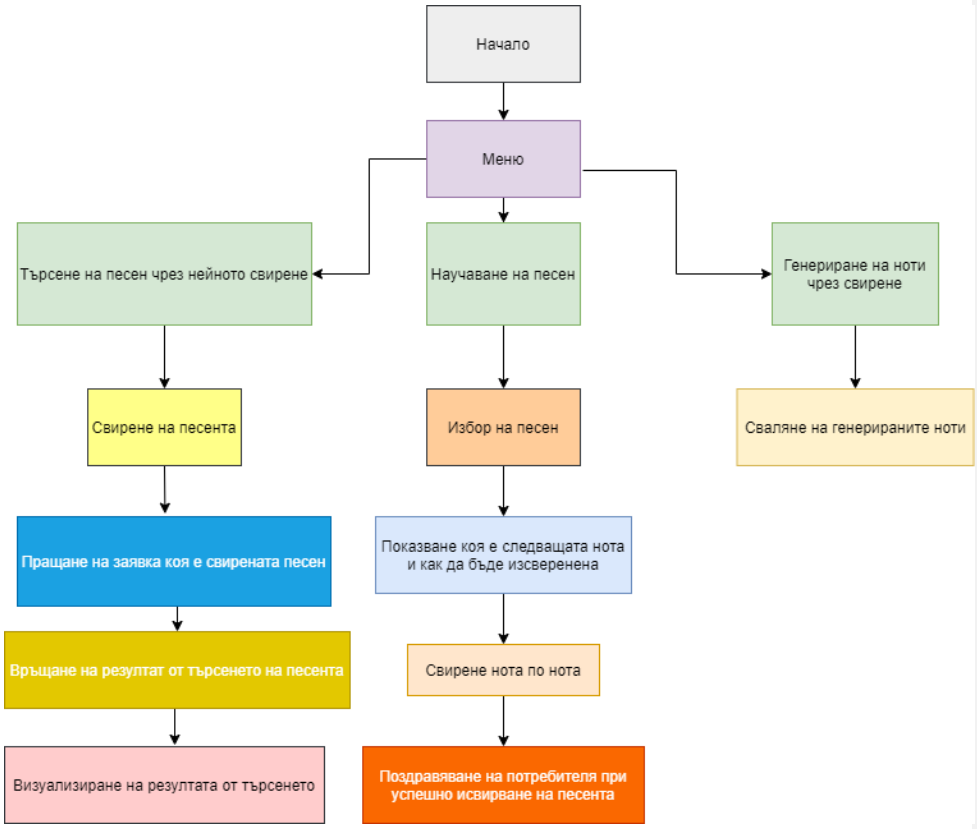
Използвани данни

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. За да може приложението да показва как потребителя да свири на различни инструменти дадена песен, имаме набор от снимки, които показва на няколко инструмента как се свирят ноти. Тук се включват снимки за пиано, китара и флейта. Тоест на практика имаме информация за:

- Духов музикален инструмент
- Клавишен музикален инструмент
- Струнен музикален инструмент

Достъпване на функционалностите от потребителя

На фигура 3 се демонстрира как се достъпват функционалностите от клиента:



Фигура 3. 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. Потребителя има възможността да изтегли генерираните ноти.

Описание на използваните езици, софтуерни средства и системи

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

JavaScript

Езикът е сред най-популярните скриптови езици за програмиране в Интернет. Той добива своята популярност, защото е единствения програмен език, който може да се изпълнява в браузера [1]. Това приложение го използва в разработката и на клиентската страна и на сървърната част. Еко системата на езика е пълна с удобни и подходящи библиотеки за разработка на ежедневни задачи. Езикът е мулти-парадигма, състояща се от различни стилове на писане като прототипно ООП(Обектно-ориентирано програмиране) и функционално програмираме. Един от страхотните инструменти, които езикът ни осигурява, е възможността да се правят асинхронни заявки към някакво API. В разработката на приложението е използвана тази техника за да може да се реализират функционалностите свързани с намирането на изсвирена песен, както и функционалността свързана с взимането на наличните песни в системата. Езикът може да се използва и за писането на сървърно приложение и сървърната система се възползва от тази опция, позволявайки клиентската страна да извлича данни от базата данни чрез сървърната система.

HTML

HTML(**H**yper**T**ext **M**arkup **L**anguage) е основния маркиращ език за описание и дизайн на уеб страници. Той е ядрото на HTML5 платформата и дава възможност

да се създават Уеб приложения, които да могат да работят на множество платформи като настолен компютър, смарт телефон, таблет и други. Основното предимство на HTML е че документите, оформени по този начин, могат да се разглеждат на различни устройства. Езика може да прикрепя JavaScript скриптове, което променя поведението на уеб страницата. Също така може да се използва и Cascading Style Sheets (CSS) [2].

CSS

CSS представлява език за описване на презентацията и стиловете на елементите в един HTML/XML документ. CSS е една от основните технологии, използвани в Уеб, редом с HTML и JavaScript [3]. Приложението използва CSS за да оформи графичния интерфейс.

Bootstrap

Bootstrap е библиотека за CSS, която позволява за по-прецизно разполагане на HTML елементите върху екрана и тяхното стилизиране.

NodeJS

NodeJS представлява JavaScript, който може да бъде стартиран извън браузера и позволява да се разработва различни приложения като сървъри. Едно от огромните му предимства е че позволява клиентската страна да се пише на същия език като сървърната и да се преизползват библиотеките. NodeJS гарантира качество, защото е Open Source и има огромна общност от програмисти, които го поддържат и развиват. Може да работи огромно разнобразие от платформи като например Window, MacOS, Linux[4]. Сървърната система на приложението се възползва от тези качества на NodeJS имплементирайки REST услуги, които клиентската система на приложението да може да консумира.

Express

Express представлява библиотека за NodeJS, която позволява да се изгражда бързи и минималистични REST услуги. Библиотеката е тествана дълго време и се е доказала като една от най-добрите в света на NodeJS [5]. Тя позволява да се разпишат лесни за четене и конфигуриране REST услуги, които с лекота да позволяват на сървърите да си вършат работата. Приложението се възползва от библиотеката като я използва за контролерите, които са отговорни за обработват заявките изпратени от клиентската страна.

MySQL

MySQL представлява многопоточна, многопотребителска, SQL система за управление на бази данни. MySQL е релационна база данни, която може да поеме огромно количество от данни и да ги управлява без рискове [6]. Приложението има за задача да работи с песни, които могат да бъдат многобройни и MySQL е отличния инструмент, който позволява приложението да се държи стабилно дори с много голямо количество от песни. Сървърната система се свързва с базата данни и се грижи да извлече информацията, от която има нужда приложението.

ABCJS

Представлява JavaScript библиотека, която има възможността да генерира ноти в браузера.

Tuner

Библиотека, която засича тонове и ги трансформира в ноти.

dom-to-image

JavaScript библиотека, давайки възможността DOM елементи да бъдат трансформирани в снимки. Приложението се възползва от тази опция, за да конвертира изведените ноти на екрана, към снимка, която клиента има възможността да свали и да си я запази или да я сподели със свои приятели и близки.

Проектиране на схематичен дизайн и потребителско изживяване

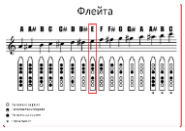
Схематичен дизайн на приложението

Всички екрани от приложението използват фона от фигура 4.

Фигура 4. Илюстрация на фона, който се използва във приложението.

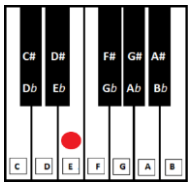
За да се осигури по-качествено обучение за потребителите на музикалното приложение са подготвени множество от снимки, които показват как различните ноти се свирят на съответните музикални инструменти.

Фигура 5. Илюстрация на нотата E за китара



Commented [u1]: Малко по-малки изображенията...

Фигура 6. Илюстрация на нотата E за флейта



Фигура 7. Илюстрация на нотата E за пиано

Музиката е изкуство и важна част от проектирането на графичния интерфейс на приложението е избора на подходящ снимка за началото на приложението, която да демонстрира чара на приложението. Изборът на това важно изображение е илюстрирано на фигура 8:

Фигура 8. Картината, която се използва за началния екран на приложението

Важно изискване при подбора на картината от фигура 8 е тя да бъде във формат .png и да бъде с опция за прозрачност за да може да се слее по елегантен начин с фона на приложението.

Проектиране на потребителския интерфейс

В процеса по проектирането на потребителския интерфейс е взето под внимание, че ще се използва grid системата на Bootstrap библиотеката, която позволява много прецизно да се разположат елементи от графичния интерфейс. Потребителският интерфейс е замислен по такъв начин, че да създаде приятна емоция при неговото ползване като осигури лесна за ползване платформа, която дава необходимата информация на потребителя. Важен етап от проектирането на потребителския интерфейс е избора на цвят за фон, и цвят за текстовете. Решения за това са взети на база лична преценка коя комбинация изглежда добре и се вписва добре с допълнителните снимки и ресурси, които използва приложението. Шрифтовете на текста използват по подразбиране шрифтовете, които са предефинирани от Bootstrap библиотеката.

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. А за да може да се отчита честотата на изсвирената музика се използва библиотеката tuner.

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.

```
40  app.get('/song', function(req, res, next) {
41     getAllSongs().then(songs => res.send(songs));
42  });
```

Фигура 9. End point отговорен за връщането на имената на наличните песни

```
35  app.post('/song', (req, res) => {
36      findSongByNotes(req.body.song).then( data => res.send(data));
37  })
```

Фигура 9. 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.

```
const getAllSongs = () => new Promise((resolve, reject) => {
    db.query('SELECT * from song', function (error, results, fields) {
        if (error){
            reject();
        } else{
            resolve(results);
        }
    });
});
```

Фигура 10. Услуга, която ще върне наличните песни.

```
const findSongByNotes = async ( notes ) => {
    const notesAsArray = notes.split(',');
    let uniqSequnce = getUniqSequence(notesAsArray);

    const result = new Promise((resolve, reject) => {
        db.query('SELECT * from song', function (error, results, fields) {
        if (error){
            reject();
        } else{
            for(let i = 0; i < results.length; i++) {
                const currentFoundSong = results[i];
                const songAsArray = currentFoundSong.value.split(',');
                const songWithUniqNotes = getUniqSequence(songAsArray);
                const areEquals = areSongsEquals(uniqSequnce, songWithUniqNotes.join(','));
                if(areEquals) {

                    resolve({ found: true, foundSong: currentFoundSong });
                    break;
                }
            }
            resolve({ found: false });
        }
    });
```

Фигура 11. Услуга, която ще провери дали изсвирената музика от потребителя съвпада с някоя песен от наличните в базата данни.

Commented [u2]: Това е много дълго – не може ли да се съкрати за да мине на горния ред?

Имплементиране на алгоритъм за търсене на песни

```
24  const MIN_PERCENTAGE_OF_HITS = 60;
25
26  const areSongsEquals = (inputSong, song) => {
27      let songFromDB = song;
28      const pairsOfThreeNotes = [];
29      for(let i = 2; i < inputSong.length; i+=3) {
30          pairsOfThreeNotes.push([inputSong[i-2], inputSong[i-1], inputSong[i]]);
31      }
32
33      const miNumberOfHits = (MIN_PERCENTAGE_OF_HITS/100) * pairsOfThreeNotes.length
34      let numberOfHits = 0;
35      for(let i = 0; i < pairsOfThreeNotes.length; i++) {
36          const foundIndex = songFromDB.indexOf(pairsOfThreeNotes[i].join(','));
37
38          if(foundIndex >= 0) {
39              numberOfHits++;
40              songFromDB = songFromDB.substring(foundIndex + 6);
41          }
42      }
43
44      return numberOfHits >= miNumberOfHits;
45  }
```

Фигура 12. 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.

```
44        <div class="container">
45            <div class="row align-items-center">
46                <div class="col">
47                    <h1>Добре дошли в музикалния свят на инструментите</h1>
48                </div>
49            </div>
50            <div class="row align-items-center menu">
51                <div class="col-sm-5 menu-items">
52                    <ul>
53                        <li>
54                            <a href="http://localhost:8080/chooseSong.html">
55                                Научете да свирите песен
56                            </a>
57                        </li>
58                        <li>
59                            <a href="http://localhost:8080/searchByPlaying.html">
60                                Потърсете песен като почнете да я свирите
61                            </a>
62                        </li>
63                        <li>
64                            <a href="http://localhost:8080/createNotesByPlaying.html">
65                                Изградете ноти като просто ги изсвирите
66                            </a>
67                        </li>
68                    </ul>
69                </div>
70                  <div class="col-sm-7">
71                      <img src="images/output-onlinepngtools.png" style="background:transparent;">
72                </div>
73            </div>
74        </div>
```

Фигура 13. Навигацията на приложението за основните функционалности.

Приложението е съставено от 3 основни функционалности:

Научаване на нова песен

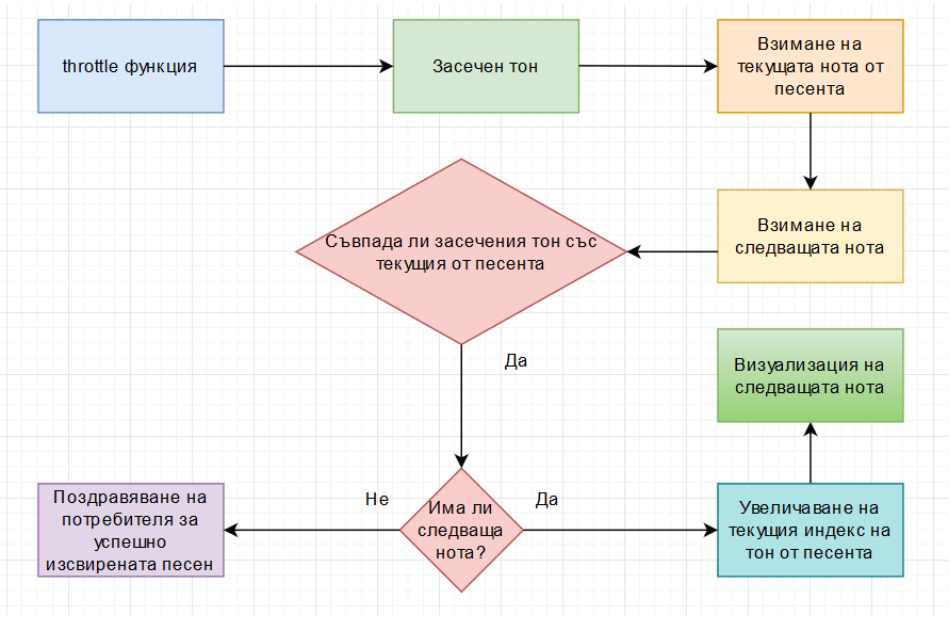
Потребителя разполага с възможността от падащо меню да си избере песен, която би искал да науча как да я свири на своя любим музикален инструмент. След избора се появява екран, който демонстрира как свири нота по нота избраната песен. Появяват се снимки на няколко музикални инструменти, които показват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

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

```
110     this.tuner.onNoteDetected = callBack;
```

Фигура 14. Илюстрация на събитието, което настъпва при засичането на тонове.

На фигура 14 се илюстрира как се подава callBack функция, която ще се погрижи да обработи засечената честота от музикалния инструмент. Имплементацията на тази callback функция е илюстрирана в приложение 2. На фигура 15 се наблюдава диаграма на работния поток на функционалността, която обработва засечените тонове.



Фигура 15. Диаграма на работния поток при обработка на засечете тонове.

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 16 може да се види, как се изпраща заявка към сървъра, когато потребителя е изсвирил достатъчно на брой ноти:

```
42          song[currentNoteIndex++] = note.name;
43          if(currentNoteIndex > notesLimit) {
44              sendMusic = true;
45              fetch('http://localhost:3000/song', {
46                method: 'POST', // *GET, POST, PUT, DELETE, etc.
47                //mode: 'cors', // no-cors, *cors, same-origin
48                headers: {
49                  'Content-Type': 'application/json'
50                },
51                body: JSON.stringify({ song: song.join(',') })
52              })
53                  .then(res => res.json())
54                  .then(res => {
55                      const [ resultElement ] = document.getElementsByClassName('found-song');
56                      const resultText = document.createTextNode(`${res.songTitle} композирана от ${res.songComposer}`);
57                      const header = document.createElement('h1');
58                      const notFound = document.createTextNode('НЕоктрита песен');
59                      if(res.found) {
60                        header.appendChild(resultText);
61                        const imageElement = document.getElementById('found-image');
62                          imageElement.style.visibility = 'visible';
63                        document.getElementsByTagName('h2')[0].classList.add('hide-element');
64                        imageElement.src = res.songImage;
65                      } else {
66                        header.appendChild(notFound);
67                        document.getElementsByTagName('h2')[0].classList.add('hide-element');
68                      }
69                      resultElement.appendChild(header);
70                  });
71          }
```

Фигура 16. Илюстрация на бизнес логиката по запращането на изсвирената музика до сървъра.

На ред 45 от фигура 16 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

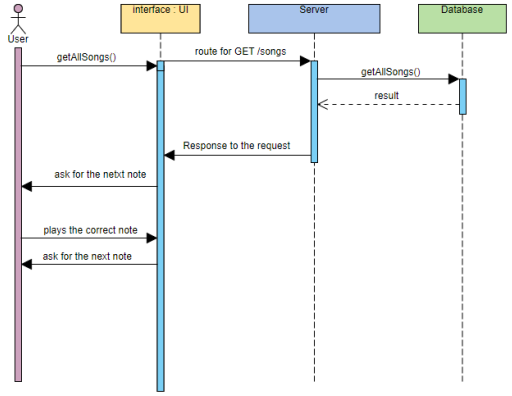
Последната функционалност на приложението също преизползва техниката за разпознаване на изсвирените тонове от първата функционалност. Приложението държи всеки изсвирен тон в масив. Този масив бива трансформиран във формат, който да се разбере от abc.js.

```
90          song[currentNoteIndex++] = note.name;
91          const textArea = document.getElementById('abc');
92          const textAreaOutput = normilizeNotes(song);
93          textArea.value = textAreaOutput;
94          initEditor();
95          window.scrollTo(0,document.body.scrollHeight);
96          document.getElementById('download-notes').style.visibility = 'visible';
```

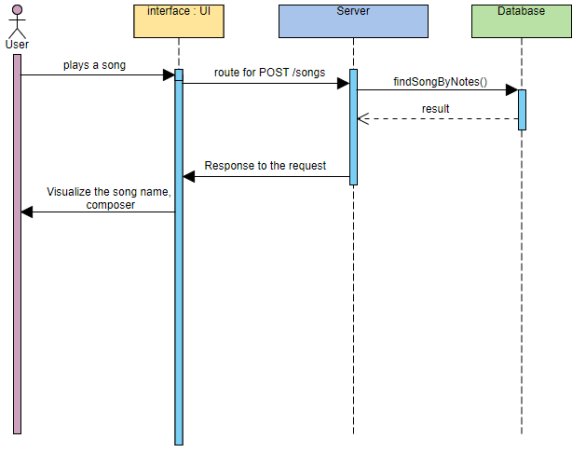
Фигура 17. Трансформиране на засечените тонове към ноти, които се визуализират на екрана.

На ред 93 от фигура 17 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 пресува визуализираните ноти. Последния ред от фигура 17 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.

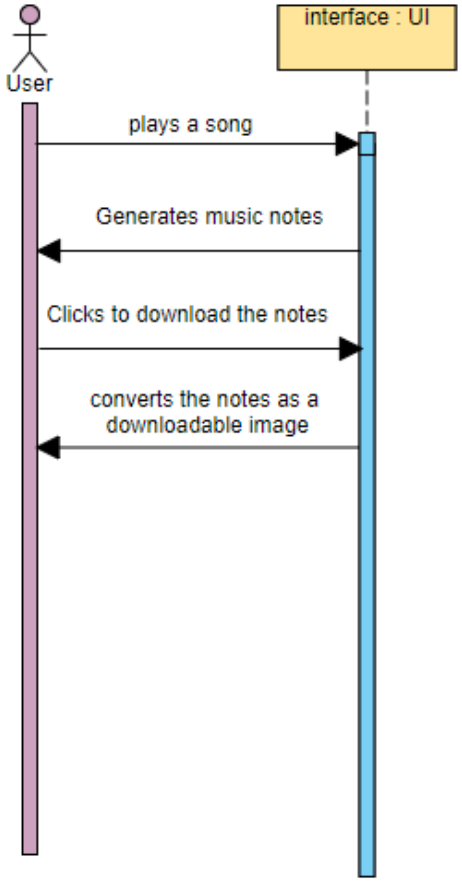
Диаграми на последователността



Фигура 18. Диаграма на последователността за функционалността свързана с обучение на нова песен.



Фигура 19. Диаграма на последователността за функционалността отговорна за намирането на изсвирената песен.



Фигура 20. Диаграма на последователността за функционалността, която позволява на потребителя да генерира ноти.

Четвърта глава

Ръководство на потребителя

Главно меню

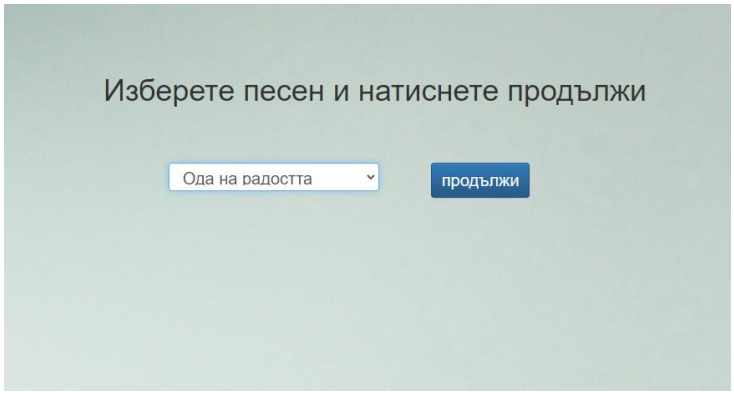
Главното меню посреща клиента при стартирането на приложението. Предоставя възможността клиента да избере една от следните три опции:

- Научете да свирите песен
- Потърсете песен като почнете да я свирите
- Изградете ноти като просто ги изсвирите.

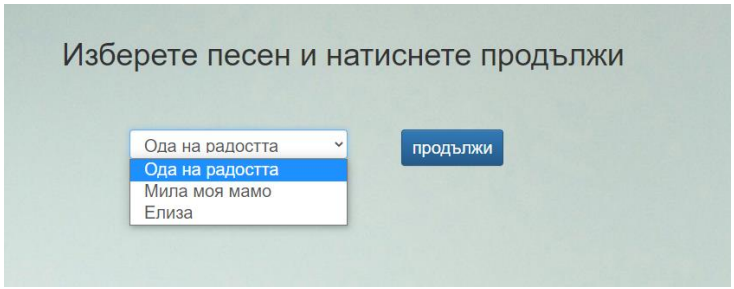
Фигура 21. Илюстрация на главното меню на приложението

Избор на песен за научване

Предоставя възможността потребителя да си избере коя песен иска да научи.



Фигура 22. Илюстрация на страницата за избор на песен за научаване

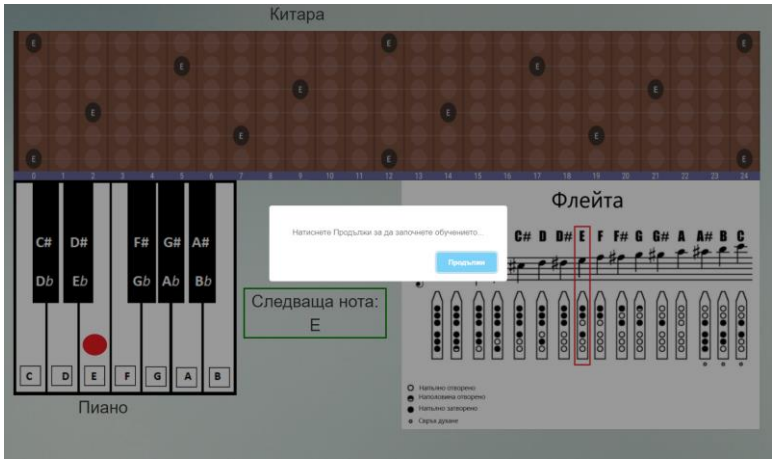


Фигура 23. Илюстрация на падащото меню за избор на песен за научаване

След като песента е избрана от падащото меню, както е илюстрирано на фигура 23, потребителят може да цъкне върху „продължи“ бутона.

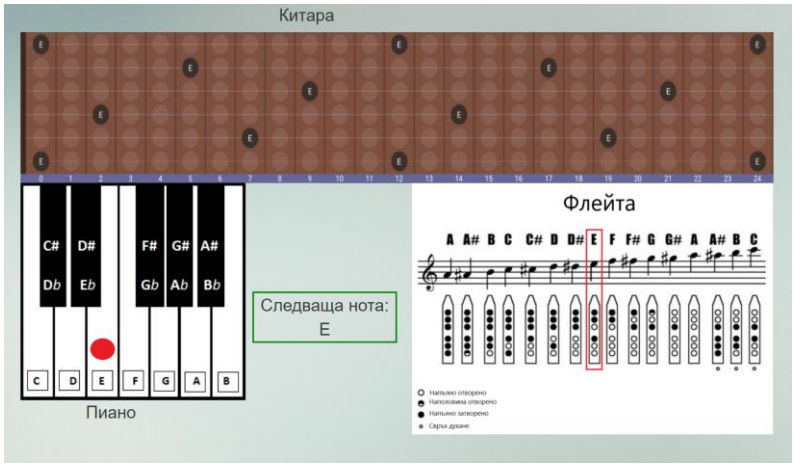
Обучение по избрана песен

Преди да започне обучението, потребителят трябва да цъкне върху „продължи“ бутона от екрана илюстриран на фигура 24.



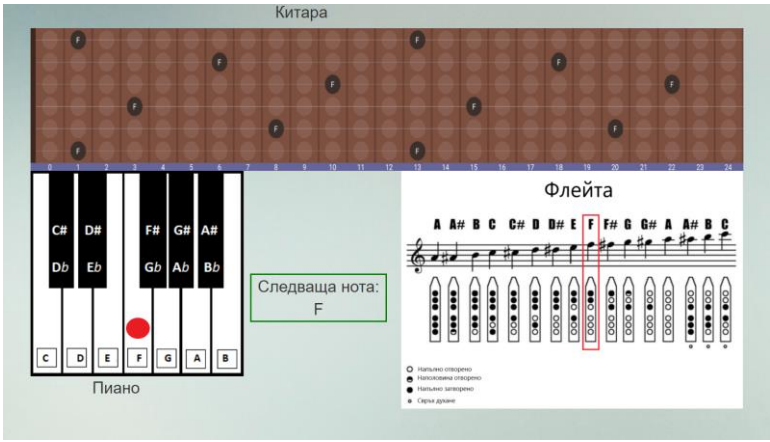
Фигура 24. Предупреждение, че предстои започване на урок.

След като избере „продължи“, пред клиента ще се появи екрана от фигура 25



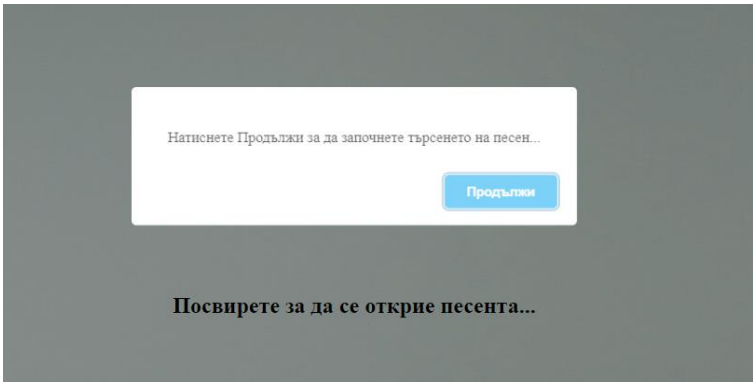
Фигура 25. Екранът, отговорен за обучението по свирене на дадена песен.

При успешно изсвирване на съответната нота, пред очите на клиента ще се покаже коя е следващата нота, която трябва да бъде изсвирена от песента. На фигура 26 може да се види следващата нота, която се очаква да бъде изсвирена:

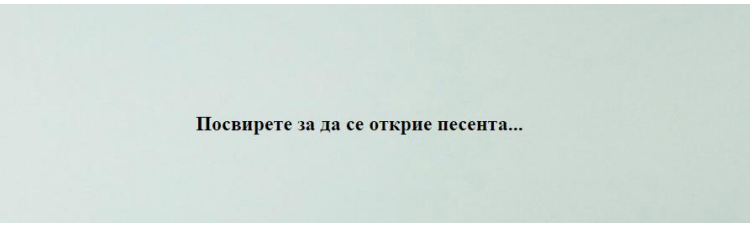


Фигура 26. Илюстрация на следващата нота от избраната песен.

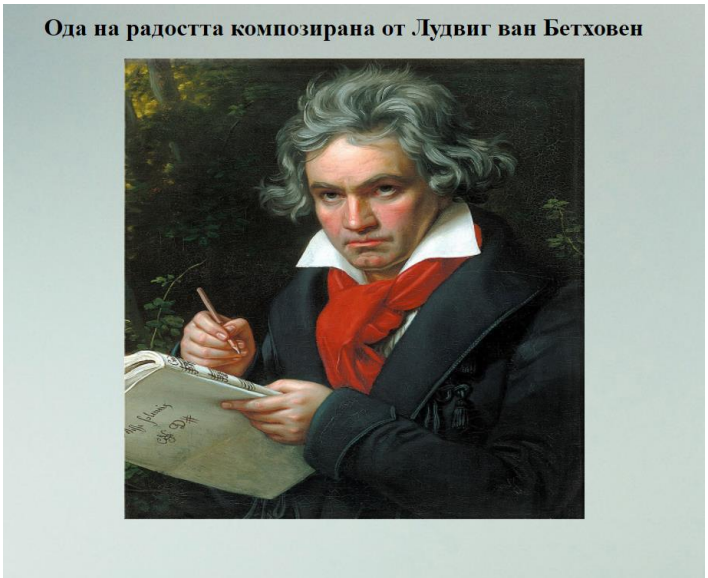
Търсене на песен чрез свирене й



Фигура 27. Предупреждение, че предстои търсене на песен.

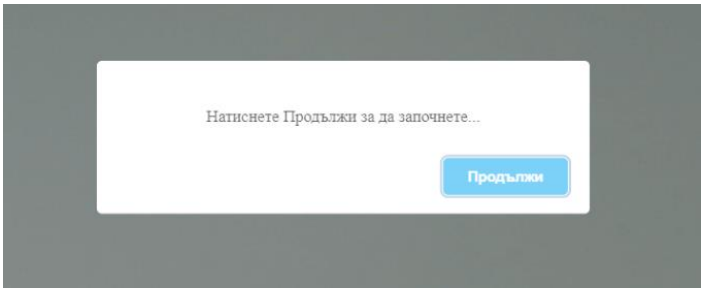


Фигура 28. Екран в процес на слушане на песента, която клиента търси.

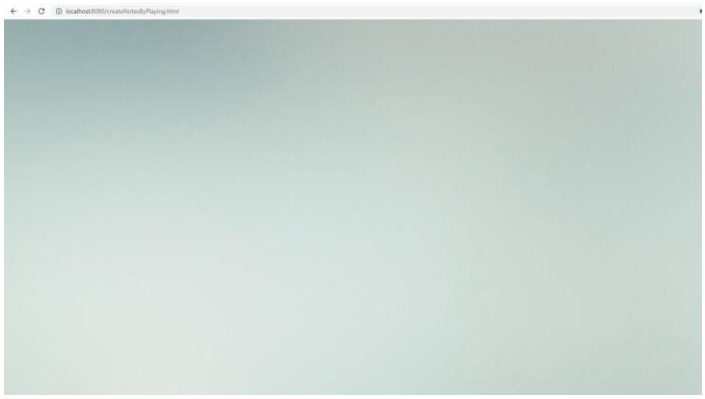


Фигура 29. Намерена песен.

Генериране на ноти чрез тяхното изсвирване

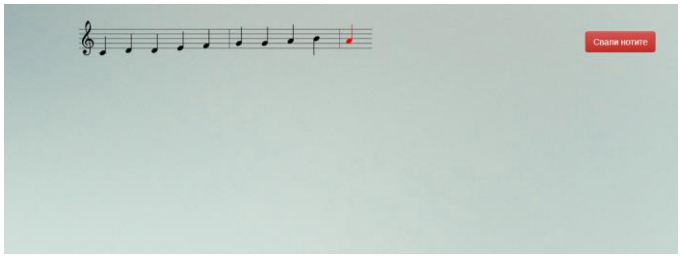


Фигура 30. Предупреждение, че предстои стартиране на функционалността за генериране на ноти.



Фигура 31. Начален екран за функционалността по генериране на ноти.

На фигура 32 се илюстрира как приложението рисува изсвирените от клиента тонове.



Фигура 32. Екран, демонстриращ вече изсвирени ноти.

След като потребителя изсвири своята музика, той има опцията да свали нотите.

Заключение

Приложението има готовност за експлоатация, където обратната връзка от клиентите ще помогне да бъде направено още по-хубаво и полезно. Предстои базата данни да се напълни с огромни количества от информация, така че да посрещне нуждите на колкото се може повече клиенти.

Системата е изцяло Open Source, което позволява откриване на бъгове и нежелани дефекти, които да бъдат бързо отстранени с цел да се достави оптималното музикално приложение за хора, които стартират в своето музикално пътешествие.

Приложението успешно посреща своите клиенти. Една от основните задачи е да се направи приложението да бъде просто за използване за малките деца, които проучването е посочило, че над 90 процента от тях искат да се научат да свирят на музикален инструмент.

Приложението предлага интересна алтернатива в лицето на конкуренцията и се цели към децата, докато интереса им към това е най-голям, както е посочено от проучването. Функционалностите, които предлага система имат потенциала да изградят добри основи в потребителите, демонстрирайки им първите стъпки в изучаването на музикален инструмент. Приложението е подходящо и за представителите на по-възрастната аудитория и е достъпно на поредица от устройства като таблет, настолен компютър, лаптоп, смарт телефон и други.

Използвана литература

1. Какво е JavaScript? [Онлайн] [Цитирано: 07 09 2020г.] https://speedflow.bg/blog/what-is-javascript-what-it-is-used-for/
2. Какво представлява HTML? [Онлайн] [Цитирано: 07 09 2020г.] https://bg.wikipedia.org/wiki/HTML
3. Какво е CSS? [Онлайн] [Цитирано: 07 09 2020г.] https://help.superhosting.bg/what-is-css.html
4. Какво е NodeJS? [Онлайн] [Цитирано: 07 09 2020г.] https://bg.wikipedia.org/wiki/Node.js
5. Описание на Express. [Онлайн] [Цитирано: 07 09 2020г.] https://expressjs.com/
6. Какво е MySQL [Онлайн] [Цитирано: 07 09 2020г.] https://bg.wikipedia.org/wiki/MySQL
7. За какво служи изкуството? [Онлайн] [Цитирано: 07 09 2020г.] https://www.az-jenata.bg/a/8-svobodno-vreme/18320-vazdeistvieto-na-muzikata-varhu-dushata-i-saznanieto-ni/
8. Описание на източника. [Онлайн] [Цитирано: 07 09 2020г.] https://nafme.org/wp-content/files/2015/12/6-WhyDoHumansValueMusic-by-Bennett-Reimer.pdf
9. Стихотворение на Чайковски. [Онлайн] [Цитирано: 07 09 2020г.] https://skif.bg/index.php/edno-kam-edno/5092-pyootr-chaikovski-krasotata-v-muzikata-e-v-prostotata-i-estestvenostta
10. Защо хората се отказват от свиренето. [Онлайн] [Цитирано: 07 09 2020г.] https://www.lifehack.org/articles/productivity/10-reasons-why-people-give-learning-musical-instruments-too-easily.html
11. Извинения за отказване научаването на музикален инструмент. [Онлайн] [Цитирано: 07 09 2020г.] https://www.joytunes.com/blog/learn-to-play/excuses-learning-to-play-an-instrument/#:~:text=The%20most%20common%20reason%20people,they%20lack%20the%20right%20tools.&text=Today%20there%20are%20loads%20of,piano%20faster%20than%20ever%20before.
12. Алтернативи. [Онлайн] [Цитирано: 07 09 2020г.] https://www.pastemagazine.com/tech/apps/learn-a-new-musical-instrument-with-these-10-great/#4-uberchord-free--
13. Проучване за хората, които искат да научат музикален инструмент. [Онлайн] [Цитирано: 07 09 2020г.] https://www.classicfm.com/music-news/children-learning-musical-instruments-survey/
14. Perfect Ear. [Онлайн] [Цитирано: 07 09 2020г.] https://apps.apple.com/us/app/perfect-ear-ear-trainer/id1440768353#:~:text=Perfect%20Ear%20-%20Ear%20Trainer%204%2B&text=Perfect%20Ear%20provides%20you%20with,make%20you%20a%20better%20musician.

Приложения

Приложение 1

Хранилище(repository) на приложението:

- https://github.com/ValkaHonda/musician-guide-tool

Приложение 2

Програмен код

Контролерът на приложението:

```
var express = require('express');
const cors = require('cors');
var app = express();
var path = require('path');
var bodyParser = require('body-parser');

/* Getting all the db querys */
const {saveUser,findUser, UpdateToken} = require('./models/user');
const { getAllSongs, findSongByNotes } = require('./models/song');
/* This middle ware checks if the token given by the user is right */
const {authenticate} = require('./middleware/authenticate');

// The code below allows the node js to find the public directory with the index.html file
const publicPath = path.join(__dirname, './public');
// Node js is using port 3000/ and when you push to cloud it will use process.env.PORT
const port = process.env.PORT || 3000; // The port of the api

// Bodyparser for using json data
app.use(bodyParser.json());
app.use(cors());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static(publicPath));


/* GET index page */
app.get('/', function(req, res, next) {
   res.render('index');
});

```

```
app.post('/song', (req, res) => {
    findSongByNotes(req.body.song).then( data => res.send(data));
})


app.get('/song', function(req, res, next) {
   getAllSongs().then(songs => res.send(songs));
});

/* This function saves a user to the db
        It uses promises.
 */
app.post('/createUser',(req,res,next)=>{
    saveUser(req.body).then((result)=>{
        return res.header('x-auth', result.token).send({email : result.email});
    }).catch((e)=>{
        return res.status(400).send(e);
    });
});

/* When the user from the front-end wants to use a function,
 The below code is an example of using the word authenticate to see if the
 user is actually authenticated
*/
app.get('/get/me',authenticate,(req,res,next)=>{
    res.send(req.user);
});

app.listen(port, () => {
  console.log(`Server is up on ${port}`);
});
```

Връзката с базата данни:

```
var mysql = require('mysql');

var db_config = {
    host     : '127.0.0.1', // Your host - either local or cloud
    user     : 'root', // your username
    password : '1234', // your password
    database : 'db_music' // database name
};

var connection;

function handleDisconnect() {
    connection = mysql.createConnection(db_config); // Recreate the connection, since
                                                    // the old one cannot be reused.

    connection.connect(function(err) {              // The server is either down
        if(err) {                                     // or restarting (takes a while sometimes).
            console.log('error when connecting to db:', err);
            setTimeout(handleDisconnect, 2000); // We introduce a delay before attempting to reconnect,
        }                                     // to avoid a hot loop, and to allow our node script to
    });                                     // process asynchronous requests in the meantime.
                                            // If you're also serving http, display a 503 error.
    connection.on('error', function(err) {
        console.log('db error', err);
        if(err.code === 'PROTOCOL_CONNECTION_LOST') { // Connection to the MySQL server is usually
            handleDisconnect();                         // lost due to either server restart, or a
        } else {                                      // connnection idle timeout (the wait_timeout
            throw err;                                  // server variable configures this)
        }
    });
}

handleDisconnect();

module.exports = connection;
```

Услугите на сървърната система:

```
var db = require('../db');


const getAllSongs = () => new Promise((resolve, reject) => {
    db.query('SELECT * from song', function (error, results, fields) {
        if (error){
            reject();
        } else{
            resolve(results);
        }
    });
});

const getUniqSequence = (notesAsArray) => {
    let uniqSequnce = [notesAsArray[0]];
    for(let i = 1; i < notesAsArray.length; i++){
        if(notesAsArray[i-1] !== notesAsArray[i]) {
            uniqSequnce.push(notesAsArray[i]);
        }
    }
    return uniqSequnce;
}

const MIN_PERCENTAGE_OF_HITS = 60;

const areSongsEquals = (inputSong, song) => {
    let songFromDB = song;
    const pairsOfThreeNotes = [];
    for(let i = 2; i < inputSong.length; i+=3) {
        pairsOfThreeNotes.push([inputSong[i-2], inputSong[i-1], inputSong[i]]);
    }

    const miNumberOfHits = (MIN_PERCENTAGE_OF_HITS/100) * pairsOfThreeNotes.length
    let numberOfHits = 0;
    for(let i = 0; i < pairsOfThreeNotes.length; i++) {
        const foundIndex = songFromDB.indexOf(pairsOfThreeNotes[i].join(','));

        if(foundIndex >= 0) {
            numberOfHits++;
            songFromDB = songFromDB.substring(foundIndex + 6);
        }
    }
```

```
const findSongByNotes = async ( notes ) => {
    const notesAsArray = notes.split(',');
    let uniqSequnce = getUniqSequence(notesAsArray);

    const result = new Promise((resolve, reject) => {
        db.query('SELECT * from song', function (error, results, fields) {
        if (error){
            reject();
        } else{
            for(let i = 0; i < results.length; i++) {
                const currentFoundSong = results[i];
                const songAsArray = currentFoundSong.value.split(',');
                const songWithUniqNotes = getUniqSequence(songAsArray);
                const areEquals = areSongsEquals(uniqSequnce, songWithUniqNotes.join(','));
                if(areEquals) {

                    resolve({ found: true, foundSong: currentFoundSong });
                    break;
                }
            }
            resolve({ found: false });
        }
    });
    });
    const data = await result;
    if(data.found) {
        return new Promise((resolve, reject) => {
        db.query('SELECT * from composer', function (error, results, fields) {
        if (error){
            reject();
        } else{
            for(let i = 0; i < results.length; i++) {
                const currentComposer = results[i];
                if(currentComposer.ID === data.foundSong.composerID){
                    resolve({
                        found: true,
                        songTitle: data.foundSong.name,
                        songComposer: currentComposer.name,
                        songImage: currentComposer.image,
                    });
                }
            }
            resolve({ found: false });
        }
    });
    });
    }
    return result;
}
```

```
module.exports = {
    getAllSongs,
    findSongByNotes,
};
```

Помощни библиотеки за сървърната система:

```
{
  "name": "mysql-express-mysql",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "engines": {
    "node": "6.5.0"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.17.2",
    "cookie-parser": "~1.4.3",
    "cors": "^2.8.5",
    "debug": "~2.6.3",
    "express": "~4.15.2",
    "jsonwebtoken": "^8.1.0",
    "mysql": "^2.14.1"
  }
}
```

HTML за генериране на ноти и тяхното изтегляне:

```
    <textarea name="abc" autofocus id="abc" cols="80" rows="15">
    </textarea>
    <div id="warnings"></div>
    <div id="paper-wrapper">
        <div id="paper"></div>
    </div>
    <button id="download-notes" class="btn btn-danger" onclick="downloadNotes()">
        Свали нотите
    </button>
    <script src="/bootstrap-3.4.1-dist/js/jquery.min.js"></script>
    <script src="/bootstrap-3.4.1-dist/js/bootstrap.min.js"></script>
    <script src="domToImage.js"></script>
    <script src="abcjs.min.js" type="text/javascript"></script>
    <script src="sweetalert.min.js"></script>
    <script src="aubio.js"></script>
    <script src="tuner.js"></script>
    <script src="createNotesByPlaying.js"></script>
```

HTML за обучението на песен, която се свири на музикален инструмент:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width,initial-scale=1,user-scalable=no"
    />
    <title>Музикално обучение</title>
    <link rel="stylesheet" href="./bootstrap-3.4.1-dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="./bootstrap-3.4.1-dist/css/bootstrap.min.css.map" />
    <link rel="stylesheet" href="./bootstrap-3.4.1-dist/css/bootstrap-theme.min.css" />
    <link rel="stylesheet" href="style.css" />
    <style>
        body {
            background-image: url("images/blur-background-1.jpg");
        }
    </style>
    <link rel="shortcut icon" href="" />
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-sm-12">
              <p class="instrument-name">Китара</p>
            <img height="300px" width="1500px" id="guitar-next-note">
        </div>
      </div>
      <div class="row">
          <div class="col-sm-4">
              <img  id="next-note-to-play">
              <p class="instrument-name">Пиано</p>
          </div>
          <div class="col-sm-4">
              <div class="next-note-container font-weight-bold"></div>
          </div>
          <div class="col-sm-4">
              <img  id="flute-next-note" height="500px" src="images/flute-notes/A.png">
        </div>
      </div>

      <div class="row">

      </div>
    </div>
    <script src="/bootstrap-3.4.1-dist/js/jquery.min.js"></script>
    <script src="/bootstrap-3.4.1-dist/js/bootstrap.min.js"></script>
    <script src="sweetalert.min.js"></script>
    <script src="aubio.js"></script>
    <script src="tuner.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

HTML за търсенето на песен, чрез нейното изсвирване:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width,initial-scale=1,user-scalable=no"
    />
    <title>Музикално обучение</title>
    <link rel="stylesheet" href="style.css" />
    <style>
        #found-image {
            visibility: hidden;
        }
        img {
            margin-left: 35%;
        }
        .hide-element {
            display: none;
        }

        body {
            background-image: url("images/blur-background-1.jpg");
        }
        h1 {
            text-align: center;
        }
        h2 {
            text-align: center;
        }
    </style>
    <link rel="shortcut icon" href="" />
  </head>
  <body>
    <div class="found-song"></div>
    <img id="found-image" width="600px" height="600px">
    <h2>Посвирете за да се открие песента...</h2>
    <script src="sweetalert.min.js"></script>
    <script src="aubio.js"></script>
    <script src="tuner.js"></script>
    <script src="searchSongByPlaying.js"></script>
  </body>
</html>
```

Обработка на засечения тон

```
const callBack = throttle(function onNoteDetected(note) {
  if (self.notes.isAutoMode) {

  const currentNoteFromTheSong = song[currentNoteIndex]
            ? song[currentNoteIndex].name
            : undefined;
    if (self.lastNote === note.name) {
      if (note.name === currentNoteFromTheSong) {
        currentNoteIndex++;



        const nextNote = song[currentNoteIndex]
          ? song[currentNoteIndex].name
          : undefined;
        const nextNoteDomElement = document.getElementsByClassName(
          "next-note-container"
        )[0];

          if(nextNote){
              document
                  .getElementById('next-note-to-play')
                  .src = normalizeNote(nextNote || '') + '.png';

              document
                  .getElementById('guitar-next-note')
                  .src= '/images/guitar-notes/' + normalizeNote(nextNote || '') + '.png';

              document
                  .getElementById('flute-next-note')
                  .src= '/images/flute-notes/' + normalizeNote(nextNote || '') + '.png';
          }

        nextNoteDomElement.classList.add("on-success");
        nextNoteDomElement.innerHTML =
          nextNote === undefined
            ? `Последната нота: ${song[song.length - 1].name}`
            : `Следваща нота: ${nextNote}`;
        setTimeout(
          () => nextNoteDomElement.classList.remove("on-success"),
          80
        );
      } else if (currentNoteFromTheSong === undefined) {
        document.getElementsByClassName("next-note-container")[0].innerHTML =
          "Поздравления!";
      }
    } else {
      self.lastNote = note.name;
    }
  }
}, 200);
```

Помощни функции:

```
function downloadNotes() {
        window.domtoimage.toPng(document.getElementById('paper-wrapper'))
            .then(function (dataUrl) {
                var link = document.createElement('a');
                link.download = 'my-image-name.jpeg';
                link.href = dataUrl;
                link.click();
            })
}

function selectionCallback(abcelem) {
        var note = {};
        for (var key in abcelem) {
                if (abcelem.hasOwnProperty(key) && key !== "abselem")
                        note[key] = abcelem[key];
        }
        var el = document.getElementById("selection");
        el.innerHTML = "<b>selectionCallback parameter:</b><br>" + JSON.stringify(note);
}

function initEditor() {
        new ABCJS.Editor("abc", { paper_id: "paper",
                generate_warnings: true,
                warnings_id:"warnings",
                abcjsParams: {
                        generateDownload: true,
                        clickListener: selectionCallback
                }
        });
}
```

```
const normalizeNote = (note) => (note.length > 1) ? note[0] + 'sharp' : note;
```

```
  const app = new Application();
  const songFromStorage = localStorage.getItem('song')
  song = songFromStorage.replace(/"/g,'').split(',').map(note => ({ name: note }));
  app.start();
```

```
function throttle(callback, limit) {
  var waiting = false; // Initially, we're not waiting
  return function () {
    // We return a throttled function
    if (!waiting) {
      // If we're not waiting
      callback.apply(this, arguments); // Execute users function
      waiting = true; // Prevent future invocations
      setTimeout(function () {
        // After a period of time
        waiting = false; // And allow future invocations
      }, limit);
    }
  };
}

const normilizeNotes = (notes) => {
    const tranformed = notes.map(note => {
        if(note.length > 1) {
            return '^' + note[0];
        }
        return note;
    }).map(note => note + '2');
    let output = '';
    for(let i = 0; i < tranformed.length; i++){
        output = output + tranformed[i];
        if(i !== 0 && i % 4 === 0){
            output = output + '|';
        }
        if(i !== 0 && i % 16 === 0) {
            output = output + '\r\n';
        }
    }
    return output;
}
```