МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

Выпускная квалификационная работа

Проектирование и разработка музыкального секвенсора на веб-платформе

Обучающегося 4 курса
Корчинского Сергея Олеговича

Научный руководитель:
Кандидат педагогических наук, доцент
Государев Илья Борисович

Санкт-Петербург
2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Музыка – неотъемлемая часть нашей жизни в 21 веке. Раньше написанием шедевров занимались талантливые люди, имеющие релевантное образование и большой потенциал. Но времена меняются, и в наш век цифровых технологий создать хит может каждый. Прогресс далеко шагнул вперед: теперь не обязательно иметь полноценную аналоговую студию с физическими инструментами, достаточно ноутбука и специального программного обеспечения – музыкального секвенсора.

Хотя затраты для занятий музыкой снизились, не каждый может позволить себе приобретение полноценного DAW (Digital audio workstation – цифровая звуковая рабочая станция) и оборудования для самой популярной операционной системы среди музыкантов – macOS, эксклюзивно поддержкой которой занимаются многие производители музыкальных секвенсоров. Необходимость создания эффективной замены для DAW, доступной на разных платформах, определяет **актуальность** данной ВКР.

Цель данной работы – проектирование и разработка музыкального секвенсора на веб-платформе.

Для достижения поставленной цели были решены следующие **задачи**:

- проанализированы существующие музыкальные секвенсоры, доступные на разных платформах и был составлен список их основных возможностей
- проанализированы и на основе анализа отобраны наиболее подходящие инструментальные средства и технологии
- спроектирована модель системы в соответствии с требованиями, выявленными при анализе существующих музыкальных секвенсоров.
- в соответствии с разработанной моделью был создан и опробован программный продукт

ГЛАВА 1. СПОСОБЫ РЕАЛИЗАЦИИ МУЗЫКАЛЬНЫХ СЕКВЕНСОРОВ

1.1. Исследование существующих музыкальных секвенсоров

Музыкальный секвенсор (англ. sequencer, от англ. sequence – «последовательность») – аппаратное устройство или прикладная программа для записи, редактирования и воспроизведения «последовательности MIDI-данных» [1]. Аппаратное устройство выглядит как микшер с ограниченными возможностями расширения. Популярность таких устройств снизилась после появления DAW в виде прикладной программы, но они по прежнему используются для записи живого звука на радио или телевизионных станциях. В данном разделе представлен обзор наиболее известных и доступных секвенсоров, которые облегчают процесс записи музыкального произведения.

В процессе исследования были проанализированы разные возможности существующих решений на рынке для выявления основного списка требований к DAW. Всего было рассмотрено подробно 6 программ:

- Image-Line FL Studio
- Ableton Live
- Steinberg Cubase
- Apple GarageBand
- Apple Logic Pro X
- ProTools

На диаграмме в рис. 1 можно посмотреть, какой процент рынка занимает каждое из решений [2].

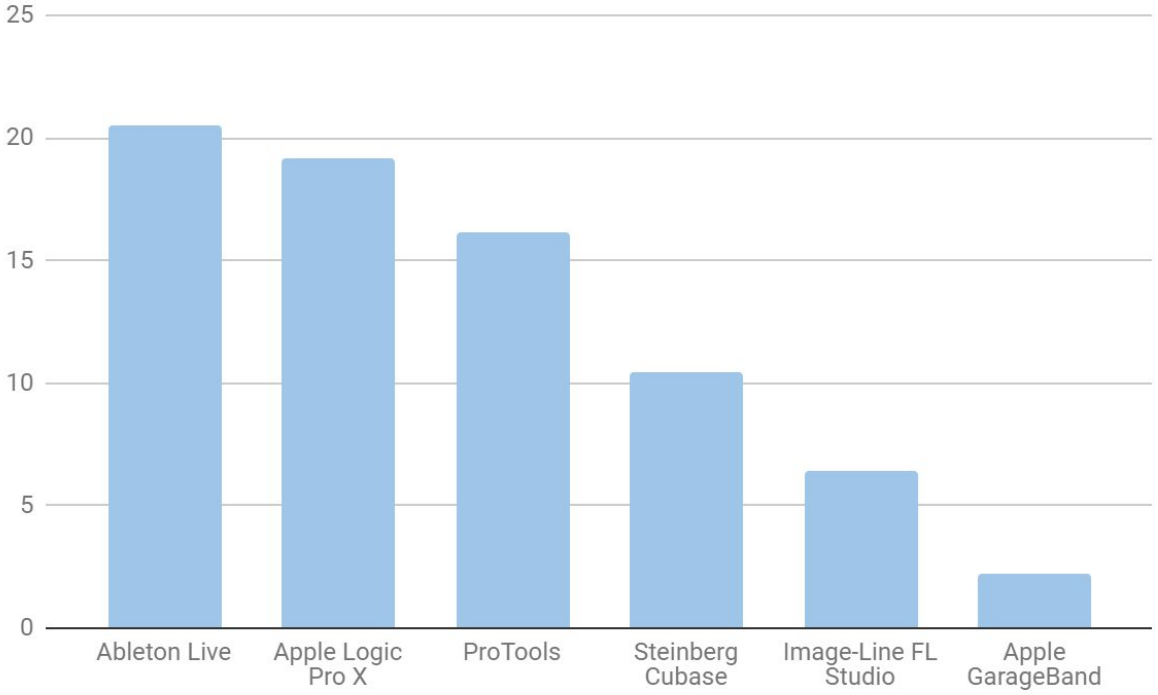


Рисунок 1 – Статистика самых используемых DAW

Ниже представлено описание каждого из секвенсоров.

Image-Line FL Studio

Один из наиболее известных DAW на рынке, хоть и является выбором всего 6,4% опрошенных респондентов[2]. С недавнего времени доступен как на macOS, так и на Windows. Путь от запуска секвенсора до сохранения проекта самый короткий относительно других программ: для работы с FL Studio понадобятся минимальные навыки в написании музыки.

FL Studio визуально сильно отличается от внешнего вида других секвенсоров. Он имеет приятный и удобный интерфейс, представленный на рис. 2, но в тоже время обладает всеми необходимыми функциями. Данным инструментом пользуются многие известные артисты, в том числе Мартин Гаррикс, Avicii, Tyler The Creator или музыкант-видеоблогер Enjoykin.

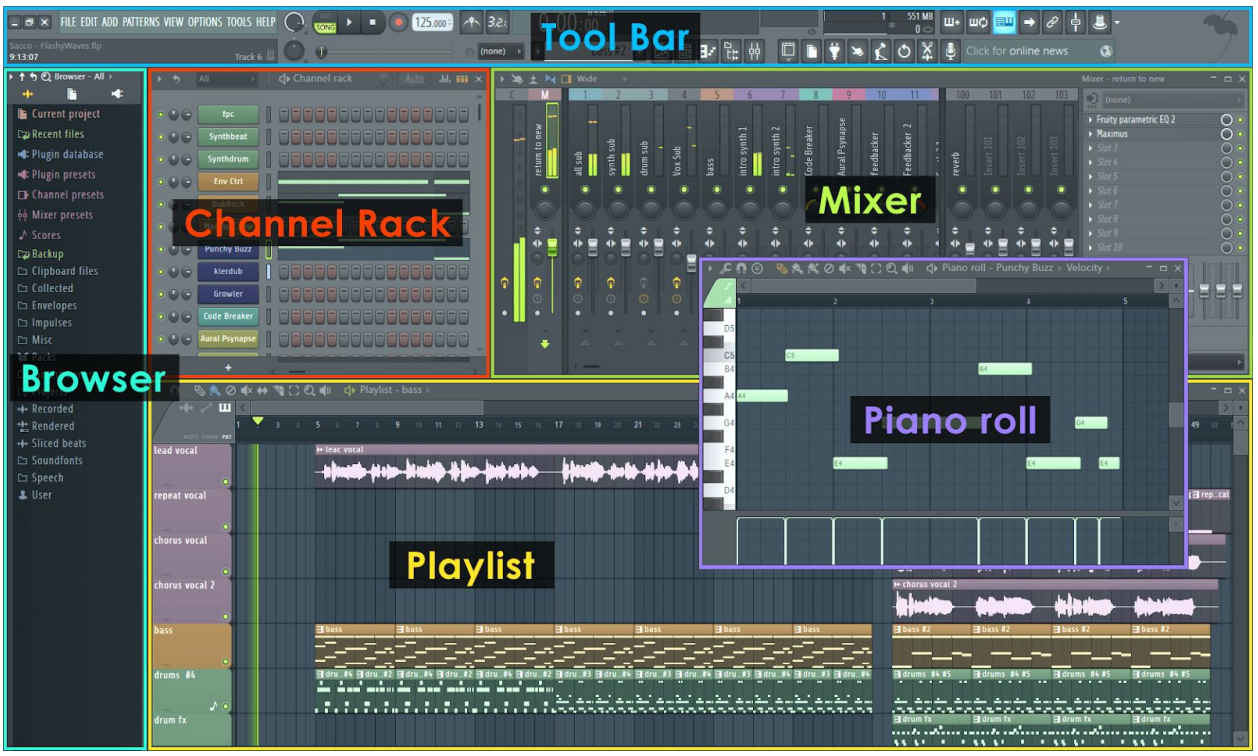


Рисунок 2 – Интерфейс FL Studio

FL Studio плохо подходит для записи живого звука, но зато можно легко работать с композицией трека, сэмплами, MIDI (Musical Instrument Digital Interface – цифровой интерфейс музыкальных инструментов), VST (Virtual Studio Technology – формат зависимых от среды выполнения плагинов реального времени) и писать музыкальные партии в Piano Roll. Piano Roll (пианоролл) это графическое представление MIDI-данных с помощью которого можно вручную менять тон звука, громкость и длительность нот. Также в комплекте с полной версией программы идут дополнительные инструменты и эффекты. Например, Gross Beat, который является одним из наиболее известных VST, с помощью которого можно управлять воспроизведением звукового потока, высотой тона, позицией и громкостью в режиме реального времени.

За FL Studio пользователю придется заплатить всего один раз, после чего все дальнейшие обновления будут доступны бесплатно.

Помимо десктопного приложения у Image-Line есть мобильные версии секвенсора для iOS и Android устройств. Проекты оттуда можно экспортировать в полноформатную FL Studio.

Ableton Live

По данным опроса, проведённого изданием Ask.Audio, Ableton Live в 2017 году обогнал все остальные секвенсоры: 20,52% респондентов используют его в качестве основной DAW-программы [2].

С помощью режимов Arrangement View для аранжировки и сведения, и Session View для импровизаций и диджеинга, Ableton Live можно использовать как в качестве классического секвенсора, так и контрольной станции для живых выступлений.

Arrangement View напоминает интерфейс большинства других секвенсоров, но с важным преимуществом: в рабочей области, демонстрируемой на рис. 3, расположено все только самое необходимое. Панель не перегружена кнопками и регуляторами, в следствии чего им можно пользоваться даже на ноутбуках с 13-дюймовым экраном.

Рисунок 3 – Интерфейс Ableton Live

Помимо Live версии, разработчики выпустили еще одну версию контроллера – Ableton Push. Ableton использует собственный протокол синхронизации Ableton Link, которые многие предпочитают устаревшему MIDI

Sync. Данный протокол создавался с расчетом на использование секвенсора с MIDI контроллерами, поэтому в Ableton Live очень просто записывать звук используя подобные контроллеры.

В Ableton также удобно работать с сэмплами и лупами (готовые повторяющиеся звуковые дорожки и файлы – loops), настраивать автоматизацию и добавлять различные плагины. Им пользуются многие известные артисты, например M83, kedr livanskiy и известный музыкальный блогер Эндрю Хуан (Andrew Huang).

Cubase

Один из наиболее старых представителей музыкальных секвенсоров. Cubase используется 10,43% пользователей [2]. Steinberg, компания-разработчик данного продукта, стояла у истоков технологий которые теперь присутствуют в каждой современной DAW: временной варпинг (возможность растягивать звуковые фрагменты с сохранением тональности) и VST-плагины [3]. Долгие годы Cubase находится на лидирующих позициях в списках среди популярных секвенсоров.

Данной DAW пользуются такие исполнители как Chvrches, Dub FX, New Order и даже Ханс Циммер.

Одна из особенностей Cubase состоит в том, что исполнители, работающие с данным DAW, могут воспользоваться специальным облаком, которое позволяет работать с проектами в режиме реального времени.

Многие пользователи считают его неудобным из-за перегруженного несколькими рабочими областями интерфейса, наблюдаемых на рис. 4, которые затрудняют работу с ним. Также в последнее время наблюдается некоторые застой в развитии программы: обновления выходят каждый год, но действительно стоящих нововведений не так много. Помимо десктопной версии, также доступен урезанный вариант для iPad.

Рисунок 4 – Интерфейс Cubase

GarageBand

GarageBand – бесплатный секвенсор от компании Apple, используемый в качестве единственного инструмента 2,22% опрошенных [2]. Возможно, он отстаёт от полноформатных станций по техническим характеристикам, но появление GarageBand – важное событие в мире любительской музыки. Благодаря данному продукту, теперь любой владелец техники от Apple имеет возможность записать собственное произведение, вне зависимости от навыков работы с DAW-программами.

GarageBand не стоит сравнивать с простыми программами: здесь приходится не просто комбинировать готовые лупы, здесь необходимо самостоятельно сочинять партии для каждого инструмента отдельно. Компактная версия DAW, доступная на iPhone и iPad, поможет реализовать идею и впоследствии с легкостью перенести свои наработки в GarageBand и Logic Pro X на macOS.

GarageBand обладает огромным количеством встроенных инструментов и эффектов с интуитивно понятным интерфейсом как на рис. 5. Также помимо этого доступен виртуальный барабанщик Drummer, который имеет огромных набор готовых партий барабанов.

Рисунок 5 – Интерфейс GarageBand

GarageBand (естественно, не в качестве основной DAW-программы) пользуются многие известные исполнители, например Рианна, Nine Inch Nails и Ноэль Галлахер.

Apple Logic Pro X

Logic Pro X – это уже более серьезный секвенсор родом из Купертино, и переход на него – естественный эволюционный этап после работы в GarageBand. Так сделали 19,2% опрошенных респондентов [2].

Один из главных преимуществ перед другими секвенсорами – принадлежность к экосистеме от Apple. Все, что работает с macOS, работает и с

Logix Pro X. Так же как и GarageBand, он умеет работать с сенсорной панелью MacBook “из коробки”.

К сожалению VST-плагины не поддерживаются – доступны только AU-плагины (Audio Units plugins). Впрочем, многие отмечают что Logic Pro X и так имеет уже все нужные при работе инструменты и эффекты.

Из минусов программы можно отметить высокую стоимость и рабочую область, видимую на рис. 6, которая не адаптирована под устройства с маленькими экранами. Несмотря на эти недостатки, Logic Pro X находится почти в любой серьезной студии звукозаписи, а пользуются ей многие знаменитые исполнители и группы. Среди них Disclosure и Foster the People и Tesla Boy.

Рисунок 6 – Интерфейс Apple Logic Pro X

ProTools

Семейство программно-аппаратных комплексов студий звукозаписи для Mac и Windows от компании Digidesign. Данному решению принадлежит целых 16,13% рынка, но в силу обязательного наличия аппаратного комплекса в связке с

секвенсором, можно не рассматривать подробно этот вариант [2]. Все ранее перечисленные DAW идут без физических устройств в комплекте и не подлежат сравнению.

Вывод

На основании качественного анализа каждого музыкального секвенсора из предоставленного списка можно отметить что все они имеют схожий функционал. Список основных функций DAW, которыми обладают самые популярные секвенсоры рынка:

- Метроном
- Банк пресетов с различными инструментами и эффектами
- Пошаговый секвенсор для записи барабанной партии
- Поддержка клавиатуры как MIDI контроллера
- Поддержка внешних MIDI контроллеров
- Piano Roll для записи и редактирования партий виртуальных инструментов
- Рабочая область для расположения записанных партий

1.2 Обзор средств и технологий реализации

Одно из требований к музыкальному секвенсору – кросс-платформенность. Это позволит иметь доступ к DAW с любого устройства, поэтому в качестве среды выполнения идеально подойдет браузер. При таком подходе программу не придется портировать на различные платформы, ведь все что требуется для запуска приложения – это веб-обозреватель.

Ниже представлены базовые технологии, каждая из которых была проанализирована и было проведено сравнение всех вариантов.

WebAssembly

WebAssembly или wasm – это низкоуровневый формат байт-кода для клиентских скриптов на стороне браузера [4]. На практике WebAssembly реализуется разработчиками браузеров на основе существующего JavaScript-движка. Первоначальная цель WebAssembly – замена JavaScript как целевого языка. Например, компилировать TypeScript код в WebAssembly, минуя этап компиляции в JavaScript. Данная технология только недавно пришла в мир веб-разработки и, хоть сейчас есть уже готовые реализации для всех основных браузеров, она по прежнему находится на этапе активной разработки и довольно сложна в обучении.

TypeScript

Не всем командам и разработчикам может подойти JavaScript: одним он кажется слишком свободным, другим не хватает дополнительных возможностей, которые не включены в стандарт ECMAScript. Требования и проекты у всех разные, поэтому в последние годы появилось множество языков, которые добавляют различные возможности «поверх» JavaScript [5]. Один из наиболее известных – TypeScript. Основная особенность языка это добавление типов в JavaScript.

Статическая типизация очень важна при разработке и поддержке крупных проектов с большими командами, но в случае когда проект разрабатывает один человек это может только замедлить разработку.

JavaScript

Но чаще всего для разработки под браузеры используют JavaScript [6]. Данный язык изначально создавался именно с этой целью и на сегодняшний день является одним из самых популярных языков программирования. По результатам

опроса stackoverflow.com, представленного на рис. 7, JavaScript занимает первое место среди разработчиков [7].

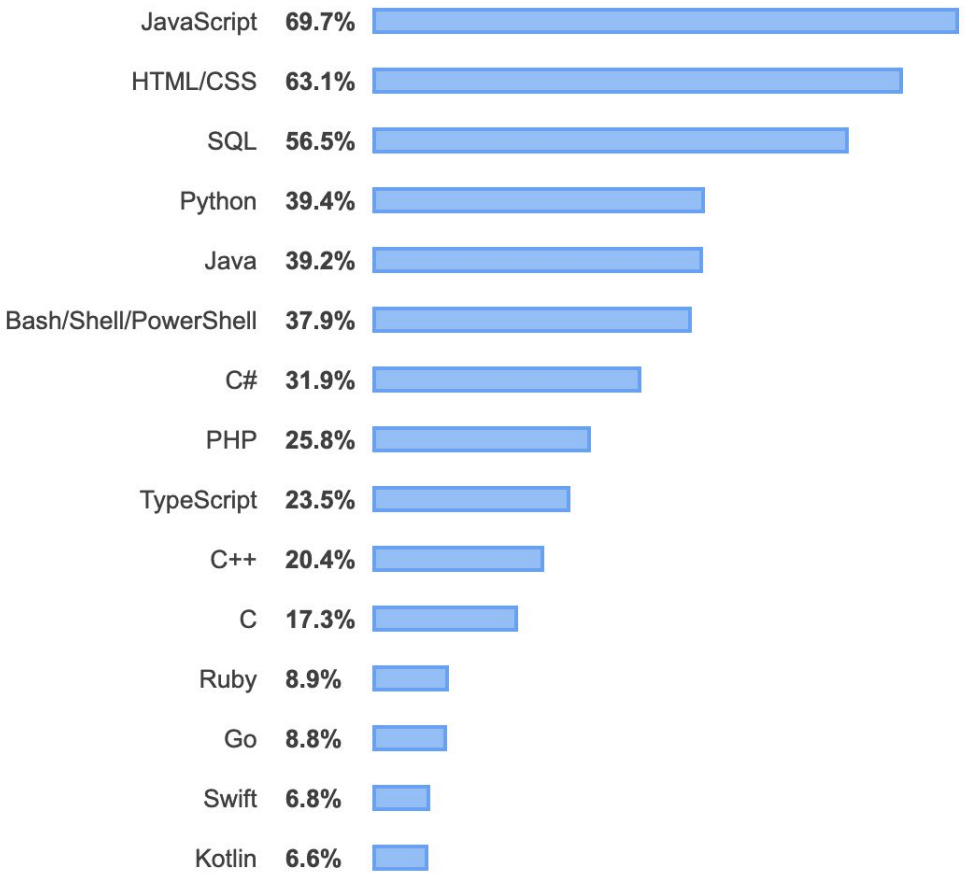


Рисунок 7 – Результаты опроса StackOverflow Survey 2019

Об этом же говорит и количество репозиториев с исходным кодом на GitHub, отображенных на рис. 8 [8].

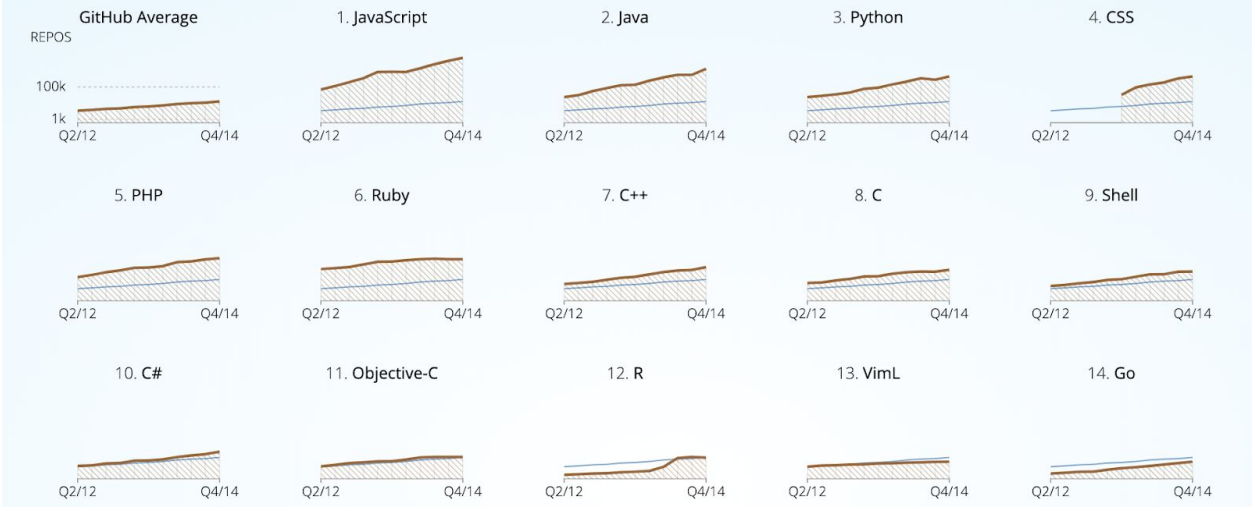


Рисунок 8 – Статистика самых используемых языков в репозиториях GitHub

JavaScript является высокоуровневым языком программирования и широко используется в веб-среде. С помощью JavaScript можно использовать различные подходы к написанию программ, так как сам язык является мультипарадигменным.

Он поддерживает следующий стили:

- объектно-ориентированный
- императивный
- функциональный

Основными сферами применения JavaScript являются:

- клиентская часть веб-приложения
- асинхронный обмен данными браузера с веб-сервером
- серверные приложения
- пользовательские скрипты

Несмотря на то, что JavaScript был создан более чем 20 лет назад, он по прежнему развивается. Каждый год члены комитета TC39 обновляются стандарт ECMAScript, которому следует сам JavaScript [9].

JavaScript обладает тремя уникальными особенностями:

- Полная интеграция с HTML/CSS
- Поддерживается всеми распространенными браузерами и включён по умолчанию
- Простые задачи можно реализовать без сложных архитектурных решений

Этими тремя вещами не обладает ни одна браузерная технология на сегодняшний день. Он остается самым распространенным средством создания браузерных интерфейсов и веб-приложений, поэтому в качестве основного

языкового средства для реализации дипломного проекта (программного продукта) был выбран язык JavaScript.

1.3 Анализ библиотек и фреймворков

Библиотекой является набор кода для решения той или иной задачи с понятным и доступным интерфейсом для взаимодействия с ней. Чаще всего библиотеки нужны для того, чтобы использовать уже готовые и проверенные решения. При вызове библиотечной функции, передав в неё параметры, библиотека выполнит её и вернёт элемент управления.

Фреймворк можно представить как некий каркас для создания приложений на определенном языке программирования. Он состоит из набора библиотек и правил, по которым следует вести разработку приложения. Помимо этого, одна из целей фреймворка – предоставление такой среды разработки, которая позволяет безболезненно расширять и дорабатывать функционал проекта постепенно.

Благодаря пакетному менеджеру npm и его команде JavaScript обрел богатую экосистему библиотек, которые с легкостью можно интегрировать в собственный проект.

В процессе анализа инструментальных средств, рассматривались следующие фреймворки:

- Vue.js
- Angular
- ReactJS

Vue.js

Название фреймворка созвучно со словом view, то есть с представлением (видом), если мы говорим о модели MVC (Model-View-Controller – «Модель-Вид-Контроллер»). Vue.js используется для решения задач именно

уровня представления, поэтому его можно с легкостью интегрировать с другими библиотеками и проектами.

Центральная концепция Vue.js – это концепция компонентов. Компоненты это небольшие элементы интерфейса, которые можно переиспользовать повторно в разных частях проекта. Один компонент может состоять из других, более маленьких компонентов, образуя древовидную иерархию, как отображено на рис. 9. Таким образом, все приложение состоит из частей-компонентов.

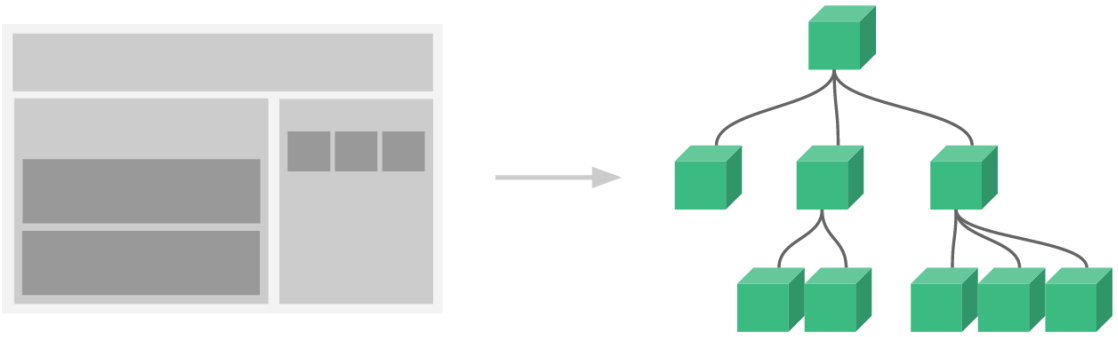


Рисунок 9 – Древовидная иерархия компонентов Vue.js

Достоинства Vue.js:

- размер – после применения метода сжатия gzip, вес библиотеки составляет всего 18 кБ
- простота в интеграции
- простота в изучении
- подробно описанная документация
- компонентный подход
- cli-утилита для быстрой развертки прототипа приложения
- браузерные расширения для разработчиков
- концепция VirtualDOM (Virtual Document Object Model – легковесная копия представления структурного документа с помощью объектов)

Недостатки:

- небольшое сообщество программистов

- маленькая экосистема библиотек
- Vue.js развивается очень быстро, из-за чего некоторые примеры и библиотеки уже устарели и не поддерживаются
- в некоторых случаях Vue.js дает недостаточный контроль, из-за чего могут возникнуть трудности

Angular

Angular это фреймворк для создания пользовательских интерфейсов с открытым кодом, поддерживаемый компанией Google. Свое начало данный фреймворк берет с 2009 года, когда двое инженеров из Google создали инструмент под названием AngularJS. В марте 2014 года началась разработка AngularJS 2.0. Новая версия была полностью переписана на TypeScript и отличалась настолько сильно от предыдущей, что было принято решение вести разработку отдельно от AngularJS под другим названием названием – Angular [10].

Изначально Angular разрабатывался для создания огромных бизнес-приложений, поэтому он получился очень многословным и сложным. Он сильно ограничивает программиста в выборе инструмента и подходов к разработке. Это основное отличие от таких инструментов как React.js и Vue.js. С последними гораздо быстрее начать разработку и предоставить продукт конечному пользователю, но в дальнейшем не так легко поддерживать как с Angular.

Достоинства Angular:

- двусторонняя привязка данных позволяет синхронизировать Модель и Представлению между собой без особых усилий
- компонентный подход
- легко поддерживать код в дальнейшем, так как фреймворк дает унифицированный подход к разработке

- поддержка TypeScript
- внедрение зависимостей упрощает тестирование и переиспользование компонентов
- большое сообщество программистов
- поддержка от Google

Недостатки:

- сообщество не довольно тем, как развивается Angular
- Angular сложный и многословный
- неполная документация
- сложность в изучении
- плохая производительность

ReactJS

При разработке ReactJS, команда Facebook пыталась решить основную проблему – разработка динамических интерфейсов с высокой производительностью. В частности, перед ними стояла задача чтобы лента новостей обновлялась мгновенно даже во время того когда пользователи используют мини-чат.

Чтобы решить эту проблему, им нужно было оптимизировать сам процесс разработки, и один из программистов предложил внедрить XHP, их собственный шаблонизатор для PHP, в JavaScript [11]. Эта идея казалась невозможной, но в 2011 году они выпустили ReactJS.

ReactJS – фреймворк, который изначально был создан разработчиками Instagram и на сегодняшний день поддерживается командой из Facebook. Согласно опросу State Of JavaScript 2018: React.js, начиная с 2016 года и по сей день, остается самым популярным инструментом (рис. 10) среди разработчиков [12].

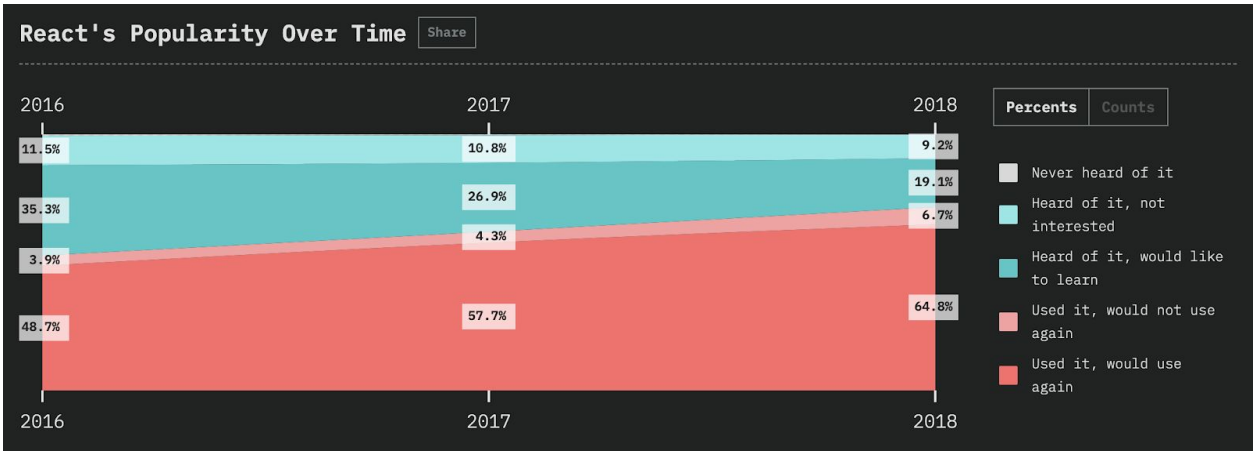


Рисунок 10 – График популярности ReactJS за 2016-2018 год

Для достижения высокой производительности ReactJS использует концепцию VirtualDOM (VDOM) – абстрактная копия реального DOM. VDOM хранится в памяти и синхронизируется с «настоящим» DOM. Благодаря такому подходу все что требуется разработчику, это указать в каком состоянии должен находится пользовательский интерфейс, а React проследит за тем, чтобы реальный DOM соответствовал этому состоянию. Более наглядно концепцию VDOM можно проследить на рис. 11.

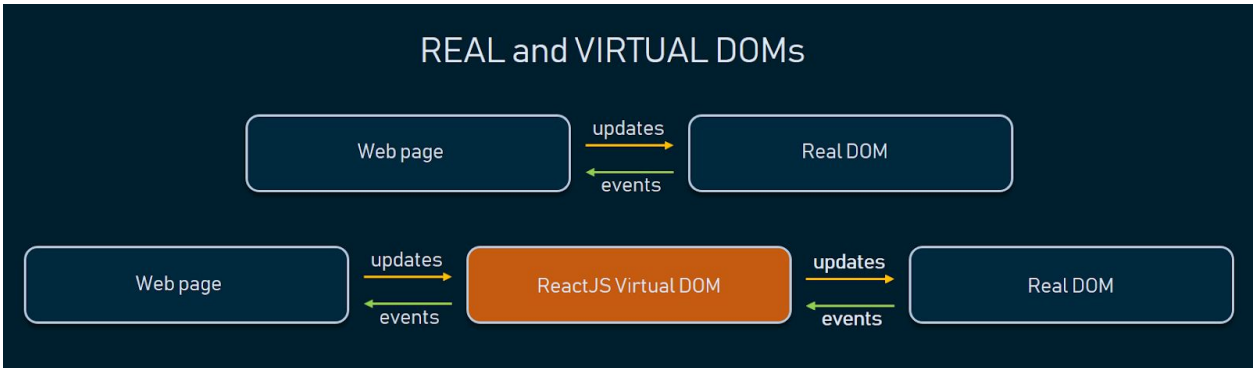


Рисунок 11 – Разница между Real DOM и Virtual DOM

Особенности ReactJS:

- в основе лежит концепция VirtualDOM
- переиспользование компонентов на любом уровне
- однонаправленный поток данных уменьшает количество ошибок при разработке
- поддержка от Facebook
- огромное сообщество программистов

- богатая экосистема библиотек и утилит

Недостатки:

- проблемы с SEO-оптимизацией
- сложен в изучении
- требует изучения JSX синтаксиса
- нет готового подхода для разработки, поэтому в некоторых ситуациях сложно решать архитектурные вопросы

Элементы – мельчайшие частицы из которых собирается React-приложение. Элемент описывает то, что должно быть выведено на экран, пример можно наблюдать на рис. 12 [13].

Рисунок 12 – Пример элемента в React.js

Чтобы разбить интерфейс на независимые части используются компоненты. Их можно переиспользовать и совмещать вместе. Объявить компонент в React.js можно двумя способами:

- с помощью обычной функции
- с помощью класса, который наследуется от React.Component

Компоненты, которые объявлены как функция, называются “функциональными”. Такие функции (рис. 13) получают данные в одном объекте и возвращают React-элемент.

Рисунок 13 – Пример функционального компонента в React.js

Важно чтобы компоненты назывались с заглавной буквы, иначе React их воспримет как DOM-тег. Например div – это тег из HTML, а Welcome – компонент.

У каждого компонента может быть собственное внутреннее состояние. Для того, чтобы добавить состояние в функциональный компонент используется специальный инструмент – хук useState. Хуки это нововведение в React 16.8, которое позволяет использовать состояние и другие возможности React без написания классов. useState возвращает значение состояния и функцию для его обновления.

Во время первоначальной отрисовки компонента значение состояния совпадает с переданным значением в useState, то есть initialState, как показано на рис. 14.

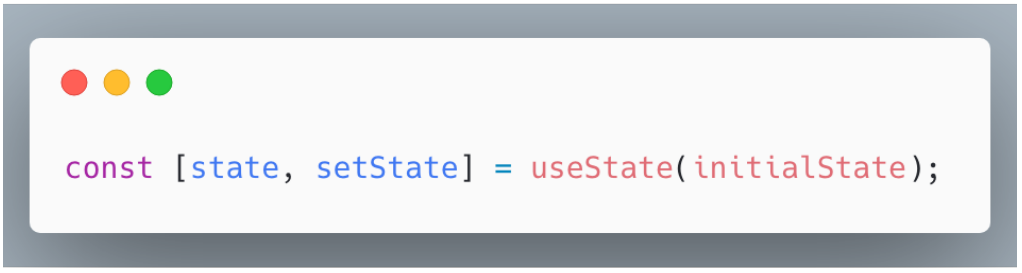


Рисунок 14 – Пример хука useState в React.js

Для обновления значения состояния используется функция setState. Она принимает следующее значение состояния и планирует повторную отрисовку

компонента. В последующих перерисовках компонанта state будет иметь самое последнее значение после применения обновлений.

Вывод

Учитывая все факторы, перечисленные в таблице 1, React.js представляется наиболее подходящим для решения поставленной задачи. Данный фреймворк обладает следующими преимуществами:

- Обширная экосистема библиотек
- Большое сообщество программистов
- Не является полноценным фреймворком, из-за чего прост в интеграции с другими библиотеками
- Полноценная документация всех возможностей

Таблица 1

	Vue.js	Angular	ReactJS
Владелец/Спонсор	Спонсируется Laravel и Alibaba	Принадлежит Google	Принадлежит Facebook
Сообщество и популярность	3	2	1
Полноценный фреймворк	Нет	Да	Нет
Компонентный подход	Да	Да	Да
Простота в	1	2	3

изучении			
Скорость разработки	2	3	1
Экосистема	3	2	1
Хорошая документаця	Да	Нет	Да
Command Line Tool	Да	Да	Да

Web Audio API

Web Audio API предоставляет мощную и универсальную систему для управления аудио составляющей в веб-приложениях. Данный инструмент дает возможность разработчикам выбирать аудио источники (звуковые файлы, потоки аудио), добавлять к ним эффекты и визуализировать их [14].

Web Audio API обрабатывает звуковые операции внутри специального контекста – AudioContext. AudioContext, который схематически изображен на рис. 15, представляет из себя изолированное окружение с аудио-узлами (AudioNode), которые соединены между собой.

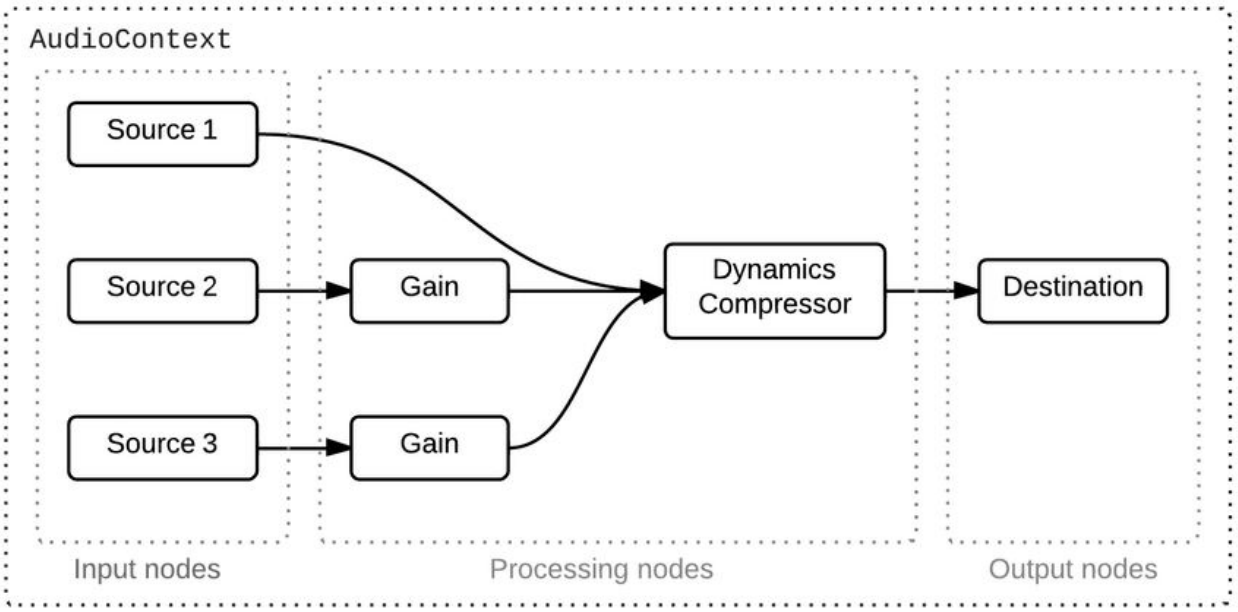


Рисунок 15 – Схематическое изображение AudioContext

Типичный порядок действий при работе с Web Audio API представлен ниже:

1. Создается аудио контекст
2. Внутри контекста создаются источники (input nodes).
3. Объявляются узлы с аудио эффектам (processing nodes). В качестве примера аудио-эффекта может выступать компрессия.
4. Создаются конечные точки (output nodes) аудио сигнала (например системные динамики).
5. Источники связываются с узлами эффектов, а узлы эффектов подключаются к конечным точкам.

В роли источников могут выступать:

- Звуки, которые генерируются непосредственно в JavaScript с помощью аудио-узла (например осциллятора);
- Данные звуковых файлов и видео файлов;
- Данные считанные с HTML медиа элементов (например <video> или <audio>);

- Захват медиа-потока с устройств (например микрофона или веб-камеры);

Tone.js

Tone.js – это фреймворк для создания интерактивных музыкальных приложений в браузере. Архитектура Tone.js спроектирована так, чтобы интерфейс был привычен как для веб-разработчиков, так и для музыкантов. Данный фреймворк предоставляет возможности планирования событий на временной шкале, синтезирования звуков и эффектов, высокоуровневую абстракцию над Web Audio API с интуитивно понятным интерфейсом для создания аудио узлов [15]. Все элементы Tone.js могут быть скомпонованы в маленькие блоки, которые впоследствии можно просто интегрировать с компонентным подходом React.js.

На сегодняшний день это единственный аудио-фреймворк для работы с Web Audio API, который имеет поддержку от сообщества программистов, обладает полной документацией и подробными примерами работы с данной технологией.

ВЫВОДЫ ПО ПЕРВОЙ ГЛАВЕ

1. На основе качественного анализа популярных музыкальных секвенсоров был выявлен список их основных возможностей.
2. Проведен обзор языковых средств для реализации веб-приложения. В качестве языка программирования был выбран JavaScript.
3. Проведен анализ библиотек и фреймворков. В результате был выбран React.js для отрисовки интерфейса и Tone.js для работы с аудио.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МУЗЫКАЛЬНОГО СЕКВЕНСОРА НА ВЕБ-ПЛАТФОРМЕ

2.1 Функциональные требования к продукту

Так как создание полноценного музыкального секвенсора, который мог бы конкурировать с перечисленными DAW из первой главы, требует огромных ресурсов, было принято решение реализовать работоспособный прототип. Для этого был составлен список ключевого функционала базовой версии, В дальнейшем планируется провести его тестирование, по результатам которого будут расставлены приоритеты для поэтапного внедрения новых возможностей в будущем.

Требования к прототипу продукта:

- Пошаговый секвенсор для записи барабанной партии
- Банк предустановленных инструментов
- Piano Roll для записи и редактировании партий виртуальных инструментов

Пошаговый секвенсор для записи барабанной партии

Инструмент для записи повторяющихся рисунков ударных. Каждая строка сетки управляет воспроизведением определенного звука ударных (элемента ударной установки), а каждая колонка представляет собой отдельный бит в рисунке ударных. Когда секвенсор активен, на каждом шаге (рис. 16) воспроизводится активные элементы ударной установки.

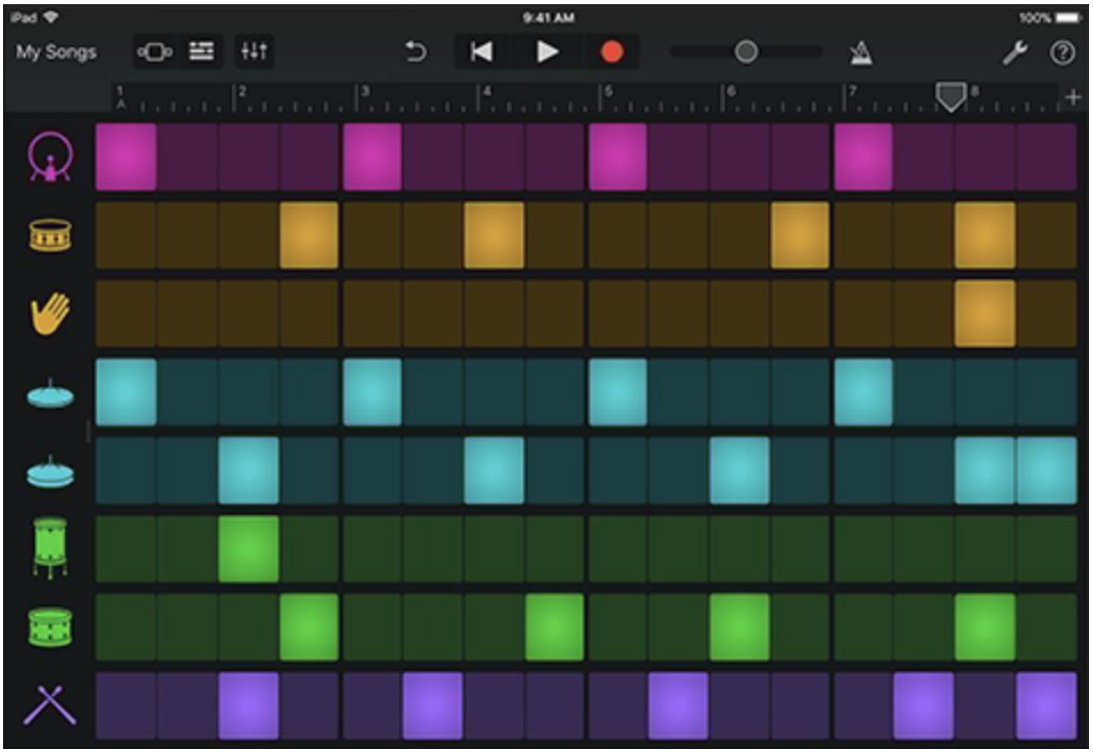


Рисунок 16 – Пример пошагового секвенсора для записи барабанной партии

Piano Roll для записи и редактировании партий виртуальных инструментов

Piano Roll (пианоролл) это графическое представление MIDI-данных с помощью которого можно вручную менять тон звука, громкость и длительность нот. Высота тона нот отображается по вертикальной оси, а время по горизонтальной. Пример реализации показан на рис. 17. Пользователь сам может выбрать разрешение для сетки Piano Roll.

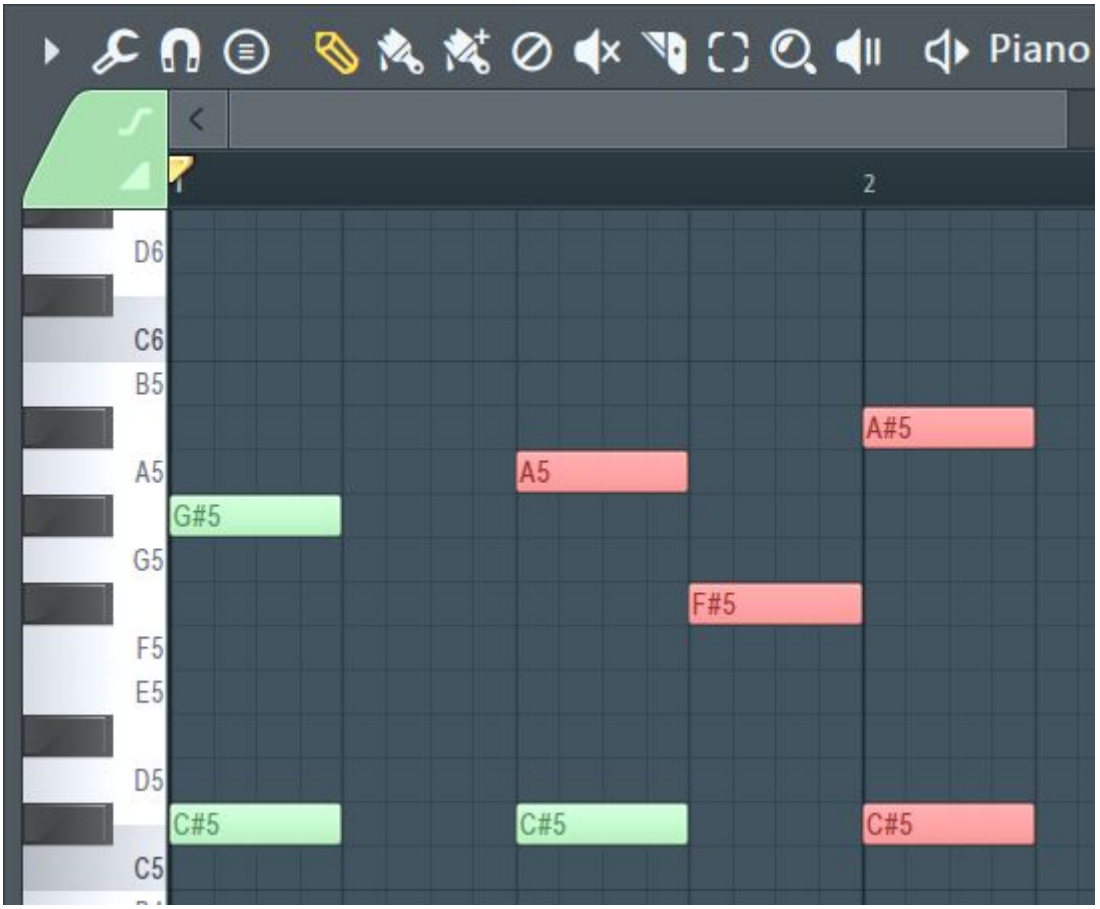


Рисунок 17 – Пример реализации Piano Roll

2.2 Модель приложения

Для прототипа было решено хранить все данные и состояние приложения на стороне пользователя. Это позволяет уменьшить затраты на аренду серверов под базы данных, сократить время разработки и при необходимости запустить приложение локально, без доступа к интернету, на компьютере пользователя.

В дальнейшем планируется доработать серверную часть, которая будет отвечать за авторизацию пользователя в системе и хранения его данных.

Ядром системы является фреймворк Tone.js, который предоставляет удобный интерфейс для работы с Web Audio API. В числе возможностей Tone.js:

- синтезирование звуков
- добавление звуковых эффектов
- синхронизация звуковых дорожек по времени
- интуитивно-понятная абстракция над Web Audio API

Расположение звуков и событий на временной шкале

Для выстраивания звуков и событий по времени используется Tone.Transport. Он предоставляет абстрактную временную линию (рис. 18) на которой располагаются события всего приложения.

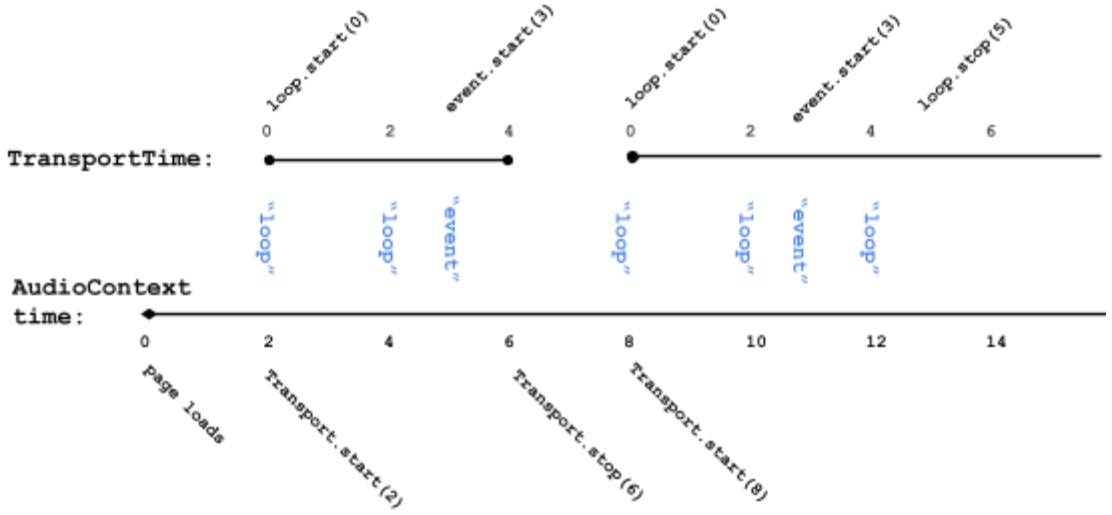


Рисунок 18 – Абстракция TransportTime над AudioContextTime

Для планирования событий используются следующие методы:

- schedule(callback, time)
- scheduleRepeat(callback, interval, startTime, duration)
- scheduleOnce(callback, time)

schedule(callback, time)

Tone.Transport.schedule позволяет запланировать вызов функции в конкретное время на временной шкале, и когда Tone.Transport достигнет этого положения, он ее вызовет.

scheduleRepeat(callback, interval, startTime, duration)

Планирует событие которое будет вызываться с установленным интервалом начиная со startTime с заранее заданной продолжительностью. Если startTime не

указан, тогда интервал начнется с текущей позиции в которой находится Tone.Transport или с начала, если Tone.Transport остановлен. Если продолжительность не указана, тогда событие будет повторяться бесконечно.

scheduleOnce(callback, time)

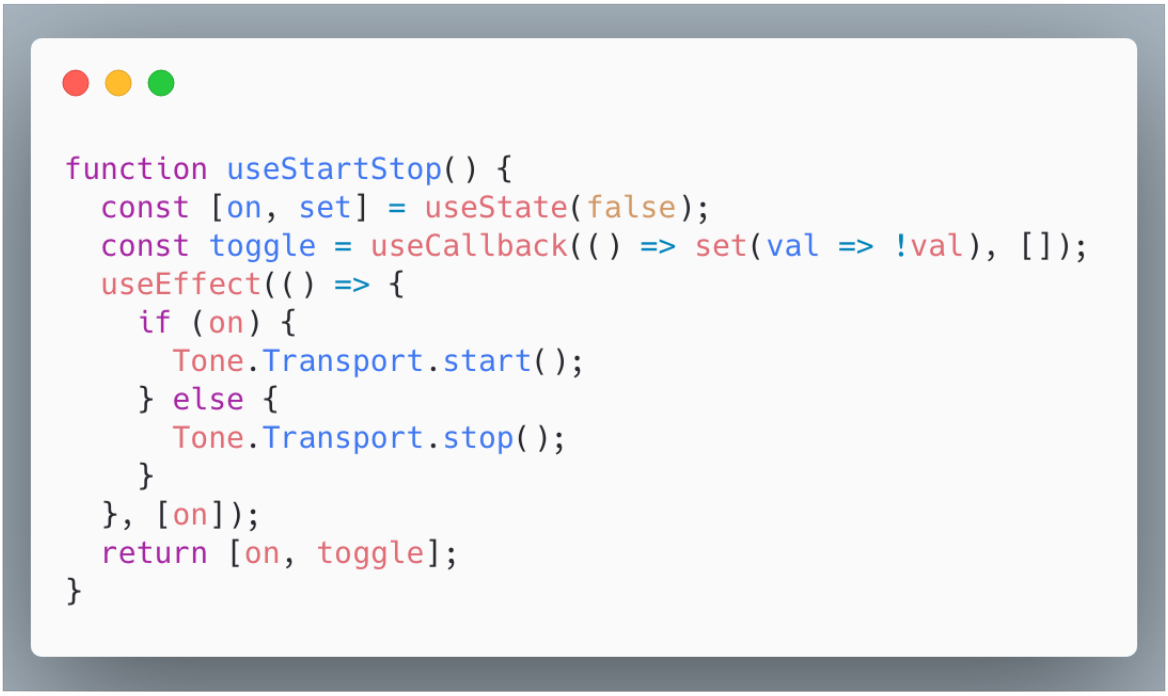
Запланированное событие будет вызвано всего один раз, после чего оно будет сразу же удалено со шкалы времени. Если указано время раньше, чем позиция Tone.Transport, оно будет вызвано немедленно.

Все эти методы возвращают идентификатор с помощью которого можно отменить событие и удалить его с помощью метода Tone.Transport.clear(eventID).

Без Web Audio API синхронизация по времени в JavaScript является не такой точной. Например стандартный вызов setTimeout(callback, 100) вызывается чуть позже чем 100 миллисекунд, а приложения работающие со звуком требуют точности в микросекундах. Чтобы решить эту проблему Tone.Transport вызывает функцию callback чуть раньше запланированного времени и передает точное время в которое должно произойти событие. Такой подход позволяет более точно синхронизировать звуки и события по времени [15].

Ни одно из событий не сможет выполниться, если Tone.Transport находится в остановленном состоянии. Для запуска Tone.Transport используется метод Tone.Transport.start.

Для удобства работы с остановкой/запуском временной шкалы был написан отдельных хук. Рассмотрим реализацию остановки/запуска на рис. 19:



```
function useStartStop() {
  const [on, set] = useState(false);
  const toggle = useCallback(() => set(val => !val), []);
  useEffect(() => {
    if (on) {
      Tone.Transport.start();
    } else {
      Tone.Transport.stop();
    }
  }, [on]);
  return [on, toggle];
}
```

Рисунок 19 – Запуск и остановка Tone.Transport

1. useState возвращает текущее состояние on в виде булевого значения: true – запущен, false – остановлен, и функцию set, которая выставляет это значение.
2. useCallback нужен для оптимизации, чтобы при каждом изменении состояния не создавалась новая функция, а использовалась та же самая. Вторым аргументом useCallback принимает массив переменных при изменении которых функция будет создана заново. В случае с toggle функция будет оставаться неизменной в любом случае, так как не зависит от внешнего лексического окружения, поэтому вторым аргументом передается пустой массив.
3. useEffect, так же как и useCallback, принимает вторым аргументом массив из переменных при изменении которых будет функция-обработчик будет вызвана заново. useEffect следит за тем, чтобы при каждом изменении состояния он запускал или останавливал Tone.Transport.

Управление темпом композиции

Параметр Tone.Transport.bpm отвечает за темп композиции. Для управления темпом используется хук useBPM, который демонстрируется на рис. 20:

```
function useBPM(initialValue = 128) {
  const [bpm, setBPM] = useState(initialValue);
  useEffect(() => {
    Tone.Transport.bpm.value = bpm;
  }, [bpm]);
  return [bpm, setBPM];
}
```

Рисунок 20 – Хук useBPM

Так же как и в примере с остановкой/запуском временной шкалы, useState хранит текущее значение BPM, и как только переменная bpm изменяется – срабатывает useEffect, так как вторым аргументом ему передан массив с переменной bpm, которая присваивается в качестве значения Tone.Transport.bpm.

Пошаговый секвенсор

Для воспроизведения звуковых файлов используется Tone.Players. При создании экземпляра Tone.Players принимает карту в которой ключи это названия, а значения – ссылки на файл. Карта хранит записи в произвольном порядке, поэтому изначально все элементы ударной установки хранятся в массиве. Такое решение позволяет сохранить последовательность дорожек в пошаговом секвенсоре.

Tone.Sequence позволяет проигрывать последовательность событий. В качестве аргументов он принимает:

- функцию для обработки события
- массив событий

- метр

Метр определяет координатную сетку для событий. В качестве события выступает один шаг секвенсора. На каждый шаг вызывается функция, которая проверяет активные элементы ударной установки на текущем шаге и воспроизводит их. Помимо текущего шага в функцию также передается время с которым вызывается метод воспроизведения звука.

Рассмотрим реализацию пошагового секвенсора. Ключевая логика программы заключена в хуке на рис. 21 под названием useSequence:

```
function useSequence({ drumKit, stepsLength }) {
  const initialStepState = getInitialStepState(drumKit.length, stepsLength);
  const [stepsState, setStepsState] = useState(initialStepState);
  const [currentStep, setCurrentStep] = useState(0);
  const toggleStep = useCallback((drumIndex, stepIndex) => {
    setStepsState(
      produce(draft => {
        const prevValue = draft[drumIndex][stepIndex];
        draft[drumIndex][stepIndex] = prevValue ? 0 : 1;
      })
    );
  }, []);
  const drumKitRef = useRef();
  const drumPositionsRef = useRef();
  useEffect(() => {
    drumPositionsRef.current = drumKit.map(({ name }) => name);
    drumKitRef.current = getTonePlayers(drumKit);
    drumKitRef.current.toMaster();
    return () => drumKitRef.current.dispose();
  }, [drumKit]);
  const stepStateRef = useRef();
  useEffect(() => {
    stepStateRef.current = stepsState;
  }, [stepsState]);
  useEffect(() => {
    const loop = new Tone.Sequence(
      (time, column) => {
        const stepStateColumn = stepStateRef.current.map(row => row[column]);
        stepStateColumn.forEach((value, drumIndex) => {
          if (value) {
            const drumName = drumPositionsRef.current[drumIndex];
            drumKitRef.current.get(drumName).start(time);
          }
        });
        Tone.Draw.schedule(() => setCurrentStep(column));
      },
      Array.from({ length: stepsLength }).map((_, index) => index),
      '8n'
    );
    loop.start(0);
    return () => loop.dispose();
  }, [stepsLength]);
  return [currentStep, stepsState, toggleStep];
}
```

Рисунок 21 – Хук useSequence

Он принимает звуки барабанной установки в виде массива, как описывалось выше, и длину секвенсора в виде количества шагов.

Состояние пошагового секвенсора хранится в виде двумерного массива. Строками матрицы двумерного массива являются дорожки для каждого элемента ударной установки, а столбцы – каждый шаг в секвенсоре. Еще один useState используется для хранения текущего шага, который подсвечивается в пользовательском интерфейсе. В текущей версии шаг может находится в двух состояниях – активный и не активный. Далее планируется расширить этот функционал добавив возможность менять громкость каждого шага и настраивать повтор нот, как это реализовано в GarageBand для iPhone.

Для активации/деактивации шага используется функция toggleStep, которая в качестве аргументов принимает drumIndex и stepIndex – индекс элемента ударной установки и индекс шага.

Основная логика работы находится в useEffect в котором создается экземпляр Tone.Sequence. В качестве событий ему передается массив состоящий из индексов шагов. На каждый шаг обработчик берет колонку из матрицы и перебирает ее по элементам ударной установки, если значение равно 1, тогда берется звук по названию элемента ударных из экземпляра Tone.Players и проигрывается с помощью метода start, который принимает время события. Важно чтобы состояние активного шага менялось не в обработчике Tone.Sequence, так как он не синхронизирован с внутренними перерисовками браузера, что может вызвать коллизии в интерфейсе. Для таких задач Tone.js предоставляется специальный интерфейс – Tone.Draw. Он помогает запланировать отрисовку интерфейса синхронно со звуком. После создания экземпляра Tone.Sequence, мы сразу его ставим в начало Tone.Transport с помощью метода start.

При создании новой последовательности или удаления старой, мы должны отключить ее от Tone.Transport и удалить из памяти. Для этого используется метод dispose. Чтобы сделать это, функция переданная в useEffect возвращает

функцию очистки. Она будет вызвана до удаления компонента из пользовательского интерфейса или перед выполнением следующего эффекта.

Чтобы при каждом изменении состояния не пересоздавать последовательность, состояние и текущая ударная установка передаются по ссылке.

Синтезирование звуков

Для синтезирования звуков используется Tone.Synth. Он представляет собой простой синтезатор состоящий из осциллятора для генерации электронного сигнала и ADSR-огибающей (Attack-Decay-Sustain-Release) для описания изменения громкости звука [15].

Для синтезирования звуков в основном используются четыре вида формы волн:

1. Синусоидальная
2. Треугольная
3. Пилообразная
4. Прямоугольная

Синусоидальная форма

Синусоидальная форма волны является самой простой. Такой сигнал дает мягкое и чистое звучание с небольшой размытостью.

Треугольная форма

Треугольная форма волны является промежуточным этапом между синусоидальной и прямоугольной формами. Эта форма волны имеет похожий мягкий звук, но уже более ярко выраженный, то есть без размытости, которая присуща синусоидальной форме.

Пилообразная форма

Пилообразная форма волны имеет резкий, режущий, едкий звук, который напоминает жужжание.

Прямоугольная форма

Прямоугольная форма волны дает более жесткий и давящий звук.

В реальном мире каждый инструмент имеет свои особенности изменения громкости. Например, орган при нажатой клавише соответствующей ноты играет ее с постоянной громкостью, а гитара воспроизводит звук максимально громко только в момент удара по струне, после чего он плавно затухает. Для духовых и струнных инструментов свойственно достижение максимальной громкости звука не сразу, но через некоторое время после взятия ноты.

ADSR-огибающая позволяет описывает изменения громкости звука как 4 последовательные фазы, представленные на рис. 22 [16]:

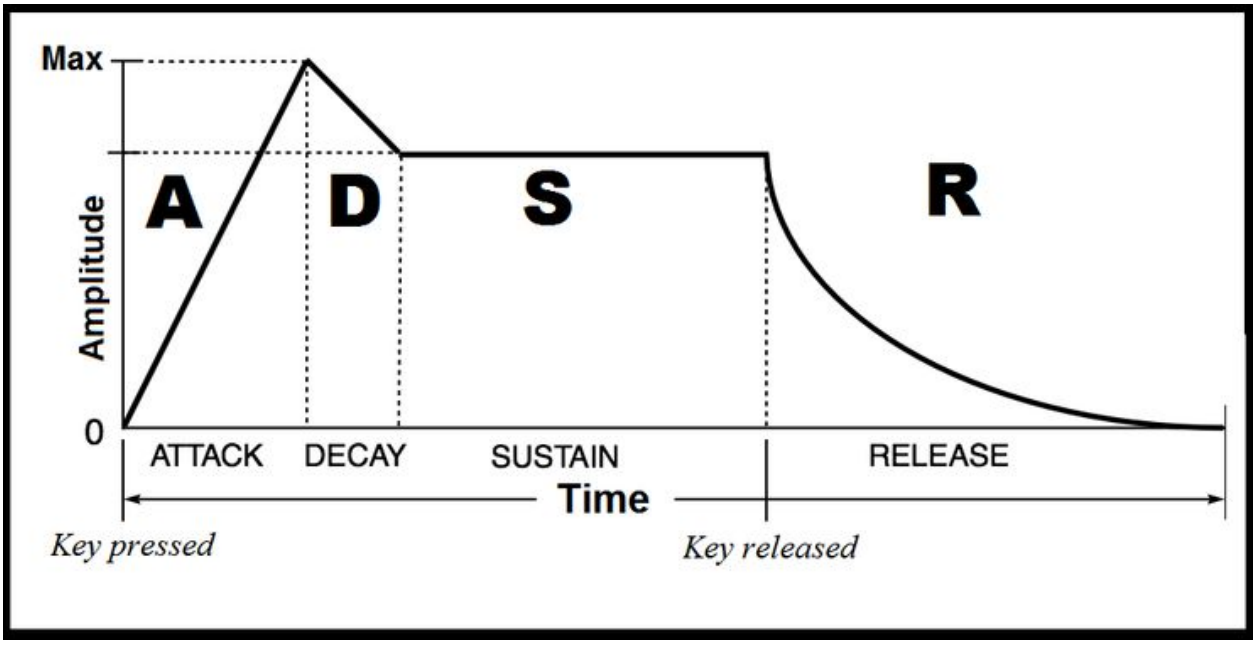


Рисунок 22 – ADSR огибающая

Attack (Атака) – фаза возрастания сигнала от полной тишины до максимального уровня громкости сигнала. Время атаки – время, нужное для того,

чтобы громкость сигнала достигла своего максимального уровня. У любого сигнала есть атака, так как амплитуда в реальной жизни изменяется непрерывно, без скачков.

Decay (Спад) – фаза спада сигнала до уровня, определенном в фазе Sustain. Аналогично, параметром является время спада.

Sustain (Удержание) – фаза "стабильного звучания" с постоянной заданной амплитудой. Если описывать нажатие клавиши на пианино, то фаза Sustain длится до того момента, пока клавиша остается нажатой. Конечно, в настоящем пианино, если долго держать клавишу нажатой, звук постепенно стихнет. Если же рассмотреть синтезатор, то он может продолжать генерировать ноту, и в этой фазе мы будем слышать постоянный, не меняющийся по амплитуде сигнал.

Release (Затухание) – фаза затухания сигнала. Определяет время нужное для окончательного спада амплитуды сигнала до нуля.

Для воспроизведения синтезированного звука каждый экземпляр Tone.Synth имеет метод triggerAttackRelease. Это комбинация из двух других методов: triggerAttack и triggerRelease. triggerAttack идентичен нажатию клавиши на инструменте. После вызова этого метода громкость ноты нарастает до установленного значения sustain в огибающей. Громкость останется на том же уровне пока не будет вызван метод triggerRelease. После вызова tiggerRelease громкость начнет снижаться в соответствии с огибающей. В качестве параметров tiggerAttachRelease принимает ноту и длительность нажатия клавиши [15].

2.3 Интерфейс пользователя

Интерфейс – это набор средств, используемый для взаимодействия двух систем. Под системами могут подразумеваться как вычислительные устройства (или их компоненты), а также сам человек. Таким образом происходит обмен данными и получение требуемой информации.

Интерфейс пользователя – интерфейс, обеспечивающий передачу информации между пользователем и программно-аппаратными компонентами системы.

Для пользовательской работы с секвенсором был реализован веб-интерфейс, который является совокупностью отображаемой на странице браузера информации и элементов управления, через которые мы можем взаимодействовать с DAW.

Рассмотрим интерфейс веб-приложения, представленный на рис. 23.

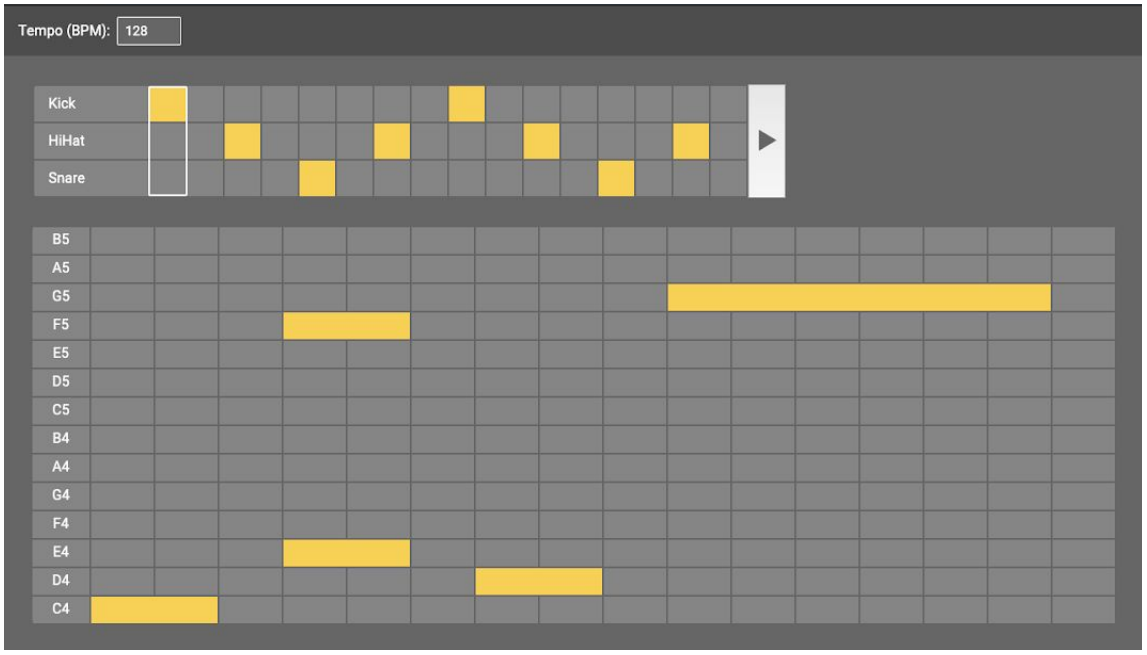


Рисунок 23 – Интерфейс реализованного секвенсора

Для управления темпом композиции используется числовое поле со значением BPM (beat per minute – удары в минуту). Пользователь может вводить произвольное целое значение. По умолчанию это значение равно 128 BPM.

Пошаговый секвенсор выглядит в виде таблицы, в которой строки это дорожки элементов ударной установки, а столбцы – шаги секвенсора. Каждый элемент ударных имеет свое название – kick, hihat, snare. Если шаг активен, он подсвечивается акцентным цветом. Текущий шаг подсвечиваются курсором в виде рамки вокруг столбца проигрываемого шага.

Чуть ниже пошагового секвенсора располагается рабочая область с Piano Roll. Здесь можно выбрать один из предустановленных синтезаторов, добавлять и удалять ноты, редактировать их тональность и длину воспроизведения.

По нажатию на кнопку Play начнется воспроизведение записанных дорожек. После запуска дорожек, кнопка Play меняется на кнопку Stop. По нажатию на нее будет произведена остановка воспроизведения.

ВЫВОДЫ ПО ВТОРОЙ ГЛАВЕ

При создании музыкального секвенсора было реализовано следующее:

1. Осуществлено проектирование модели и разработка прототипа конечного продукта.
2. Проведено тестирование прототипа с целью выявления дальнейших путей развития и утверждения выявленных требований к конечному продукту в первой главе.

ЗАКЛЮЧЕНИЕ

При выполнение данной дипломной работы был разработан прототип музыкального секвенсора, который позволяет записывать небольшие кусочки композиции.

В ходе выполнения работы были решены следующие задачи:

- Проанализированы существующие музыкальные секвенсоры, выделены основные возможности DAW и сформирован список функциональных требований;
- Проведен анализ языковых средств, библиотек и фреймворков. На основе полученных результатов выбран оптимальный язык программирования и библиотека для решения поставленной задачи.
- Спроектирована модель приложения. Описаны основные архитектурные особенности программного продукта.
- Разработан рабочий прототип продукта.

Таким образом, поставленные задачи полностью реализованы, а цель дипломной работы достигнута.

В будущем планируется доработка существующего прототипа по утвержденным требованиям и запуск с публичным доступом.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Kefauver, Alan P.; Patschke, David: Fundamentals of Digital Audio, New Edition / John Strawn, James Zychowicz: A-R Editions, 2007. 133 с.
2. Rounik Sethi The Top 11 Most Popular DAWs [Электронный ресурс]. – Режим доступа https://www.macprovideo.com/article/presonus-studio-one/the-top-11-most-popular-daws-you-voted-for/ Дата обращения: 10.05.2019
3. Derek Johnson, Debbie Poyser Steinberg Cubase VST [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20150914175804/http://www.soundonsound.com/sos/1996_articles/jul96/steinbergcubase3.html Дата обращения: 10.05.2019
4. Спецификация WebAssembly [Электронный ресурс]. – Режим доступа: https://webassembly.github.io/spec/core/syntax/conventions.html Дата обращения: 10.05.2019
5. Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/intro#yazyki-poverh-javascript Дата обращения: 10.05.2019
6. Флэнаган Д. JavaScript. Карманный справочник. Сделайте веб-страницы интерактивными! / Сысонюк А.: Вильямс, 2015. 320 с.
7. Most Popular Technologies: Programming, Scripting, and Markup Languages [Электронный ресурс]. – Режим доступа: https://insights.stackoverflow.com/survey/2019#technology-_-programming-scripting-and-markup-languages Дата обращения: 10.05.2019
8. A small place to discover languages in github [Электронный ресурс]. – Режим доступа: https://githut.info/ Дата обращения: 10.05.2019
9. Zakas N. What is JavaScript? Professional JavaScript for Web Developers / USA, Canada: Wiley Publishing, 2009. 840 с.

10. Блог команды Angular [Электронный ресурс]. – Режим доступа: https://blog.angularjs.org/2017/01/branding-guidelines-for-angular-and.html Дата обращения: 10.05.2019
11. Ответ разработчика из команды React.js [Электронный ресурс]. – Режим доступа: https://www.quora.com/React-JS-Library/How-was-the-idea-to-develop-React-conceived-and-how-many-people-worked-on-developing-it-and-implementing-it-at-Facebook/answer/Bill-Fisher-17 Дата обращения: 10.05.2019
12. Опрос State of JavaScript [Электронный ресурс]. – Режим доступа: https://2018.stateofjs.com/front-end-frameworks/react/ Дата обращения: 10.05.2019
13. Документация React.js [Электронный ресурс]. – Режим доступа: https://reactjs.org/docs/rendering-elements.html Дата обращения: 10.05.2019
14. Сайт веб-документации MDN [Элетронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API Дата доступа: 10.05.2019
15. Documentation Tone.js [Электронный ресурс]. – Режим доступа: https://tonejs.github.io/ Дата обращения: 10.05.2019
16. Teach me Audio: Sound Envelopes [Электронный ресурс]. – Режим доступа: https://www.teachmeaudio.com/recording/sound-reproduction/sound-envelopes/ Дата обращения: 10.05.2019