Санкт-Петербургский государственный университет

Данилов Александр Глебович

Выпускная квалификационная работа

Разработка визуального редактора песен и сборников песен с буквенно-цифровой нотацией аккордов

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
доцент кафедры системного программирования, к.т.н. Ю. В. Литвинов

Консультант:
ст. преп. кафедры системного программирования И. В. Зеленчук

Рецензент:
программист-разработчик ООО «В Контакте» Е. Н. Котляров

Санкт-Петербург
2023

Saint Petersburg State University

Aleksandr Danilov

Bachelor's Thesis

Visual editor for songs with noted chords and songbooks development

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2019 "Software and Administration of Information Systems"*

Scientific supervisor:
C.Sc., System Programming chair docent Y. V. Litvinov

Consultant:
System Programming chair sr. lecturer I. V. Zelenchuk

Reviewer:
Program developer at "VK" E. N. Kotlyarov

Saint Petersburg
2023

Оглавление

Введение **5**

1. Постановка задачи **6**

2. Введение в предметную область **7**

3. Обзор **8**

3.1. Запись в текстовом формате 8

3.2. Формат ChordPro 9

3.3. Прочие форматы 10

4. Требования **11**

4.1. Постановка проблемы 11

4.2. Портрет пользователя 11

4.3. Итоговые требования 13

5. Архитектура **14**

5.1. Главный сценарий использования продукта 14

5.2. Стек технологий 16

5.3. Внутреннее представление песен 17

5.4. Диаграмма компонент 19

5.5. Интерфейс 22

6. Особенности реализации **26**

6.1. Сохранение изменений во внутреннем представлении . . 26

6.2. Сохранение и загрузка сохранённых песен 27

6.3. Реализация экспорта в PDF 28

6.4. Реализация импорта из текстового формата 29

6.5. Доработки редактора песен 30

6.6. Настройка CI/CD 31

7. Апробация **33**

Заключение **35**

Список литературы **36**

Введение

Музыка всегда занимала значимое место в жизни людей. Как и природного происхождения в виде, например, птичьего пения, так и исполняемая человеком. Для исполнения музыки чаще всего используются либо человеческий голос, либо специальные музыкальные инструменты. Существует множество музыкальных произведений, но для каждого из них, как правило, характерной отличительной чертой является собственная мелодия. Для освоения музыкального произведения можно либо, услышав его исполнение, постараться освоить его по памяти, либо же воспользоваться записью, также называемой музыкальной нотацией.

Одним из наиболее знакомых современному человеку музыкальных инструментов является акустическая гитара. Среди различных способов исполнения музыкальных произведений на гитаре, существует такой способ, как игра аккомпанементом. Этот способ подразумевает, что при помощи гитары ведётся фоновая мелодия, тогда как основная мелодия исполняется иным образом - голосом или же на отдельном инструменте.

Распространённым способом музыкальной нотации для аккомпанирующей голосу гитары является запись текста песни дополненная буквенно-цифровой нотацией исполняемых аккордов.

Достаточно часто возникает запрос на создание свёрстанных сборников песен в такой нотации. Существуют и разнообразные способы и средства создания таких сборников. Все эти способы отличаются трудоёмкостью на этапе вёрстки.

Данная работа является продолжением работы, выполненной автором в рамках учебной практики 3 курса, по итогам которой был разработан визуальный редактор песен с буквенно-цифровой нотацией аккордов.

В настоящей работе было предложено развить созданный редактор в продукт "ChordWriter" - визуальный редактор для песен и сборников песен с буквенно-цифровой нотацией аккордов.

1. Постановка задачи

Целью данной работы является развитие продукта «ChordWriter», представляющего собой десктопный WYSIWYG редактор песен и сборников песен с буквенно-цифровой нотацией аккордов. Для достижения этой цели были поставлены следующие задачи:

1. Сформулировать требования к приложению;
2. Реализовать новую функциональность редактора согласно требованиям;
3. Произвести апробацию полученного приложения.

2. Введение в предметную область

Основными составляющими записи песен с буквенно-цифровой нотацией аккордов являются текст песни и определённым образом расположенные аккорды, записанные в буквенно-цифровой нотации.

Приведём пример[20]:

```
          C      G              C
Мы соль земли, мы украшенье мира,
        A7                  Dm
Мы полубоги - это постулат,
```

Аккорд - это созвучие из трёх и более звуков[18]. Существует два основных метода аккомпанирующей игры на гитаре - бой и перебор. Оба из них чаще всего подразумевают использование аккордов. Аккорды зажимаются на грифе гитары и, как правило, меняются в ходе исполнения.

Одним из способов записи аккордов является буквенно-цифровая нотация. Буквенно-цифровая нотация - это система обозначений латинскими буквами, цифрами и арифметическими знаками '+', '-', '/'. Иногда, вместо знаков '+' и '-' используются знаки '#' - диез и 'b' - бемоль, соответственно[21].

Важной составляющей такой записи песен является местоположение аккорда. Оно отражает, в какой момент песни следует сменить зажатый аккорд, сопоставляя этот момент с пропеваемым слогом из текста песни.

3. Обзор

В рамках работы были проанализированы различные подходы к записи песен с буквенно-цифровой нотацией аккордов. Исследование позволило обнаружить следующие способы:

- запись в формате обычного текста;
- запись при помощью компилируемых форматов:
  - формат ChordPro;
  - язык разметки HTML;
  - система вёрстки TEX.

Рассмотрим подробнее перечисленные подходы.

3.1. Запись в текстовом формате

Наиболее часто встречающимся форматом записи оказалась запись в виде обычного текста. В частности, таким образом предлагается записывать песни на сайтах AmDm.ru[1], Ultimate Guitar[8], в популярном Android-приложении "Песни под гитару Rus"[11], приложениях Opensong[6] и Songcraft.io[7]. Набор песен при таком подходе сводится к следующему:

1. Записывается текст песни;
2. Над строчками песни, для которых необходимо указать аккорды, добавляются пустые строки;
3. В полученных пустых строках записываются аккорды. Позиционирование над нужным слогом строки песни осуществляется пробелами и/или табуляцией.

Главным преимуществом этого формата является его доступность и универсальность. Для создания записей песен не требуется ничего, кроме базового текстового редактора. Помимо этого, при использовании

одинаковых и/или моноширинных шрифтов, полученная запись может быть просмотрена на любом устройстве, которое способно отображать текстовые файлы.

Вместе с тем такой подход обладает рядом недостатков:

1. Трудоёмкость расстановки аккордов При позиционировании аккордов пробелами для установки одного аккорда потребуется не только ввести аккорд, но и вручную выставить требуемое количество пробелов;

2. Сложности при изменении вёрстки В случае изменения шрифта может также потребоваться перерасстановка аккордов из-за изменения ширины символов шрифта. Эта проблема может быта решена использование моноширинного шрифта. Однако, в свою очередь, моноширинные шрифты могут не удовлетворить всех потребностей дизайнера при вёрстке;

3. Сложности при унификации вёрстки В сети интернет распространено большое количество различных сайтов с записями песен с буквенно-цифровой нотацией аккордов. При копировании в общий файл они могут иметь различающуюся вёрстку. Для использования в дальнейшем потребуется её унификация. При этом, если в источнике использовался не моноширинный шрифт, то потребуется перерасстановка аккордов;

4. Сложности при корректуре Аккорды и текст никак не связаны друг с другом иначе, кроме как визуально. В случае необходимости корректуры текста и аккордов может потребоваться перерасставлять все аккорды.

3.2. Формат ChordPro

ChordPro - это открытый текстовый формат, используемый для вёрстки песен с буквенно-цифровой нотацией аккордов в программе ChordPro[4]. Спецификация формата доступна на сайте формата и программы ChordPro[3].

3.3. Прочие форматы

Также при исследовании встречались записи песен, сделанные при помощи языка разметки HTML[2]. В практике автору также встречались записи сделанные с помощью системы TEX(в том числе в рамках данной работы). Основным недостатком подобных подходов является порог вхождения.

4. Требования

«ChordWriter» представляет собой редактор песен с буквенно-цифровой нотацией аккордов с возможностью вывода редактируемой, а также сохранённых ранее песен в виде автоматически свёрстанного песенного сборника в формате PDF.

4.1. Постановка проблемы

Основной проблемой, которую предполагается решать продукту является высокая трудоёмкость создания и вёрстки сборников песен с аккордами, записанными в буквенно-цифровой нотации.

Был проведён анализ процесса составления сборника, в котором были выделены следующие этапы:

1. Сбор материалов:
2. Ввод песен, для каждой состоящий из:
    - ввода текста песни;
    - расстановки аккордов:
3. Вёрстка и оформление сборника, приведение введённых песен к единому стандарту оформления.

В данной схеме «ChordWriter» покрывает 2 и 3 пункты, фактически устраняя процесс приведения к единому стандарту оформления.

Основным итогом работы пользователя с программой должны служить готовые свёрстанные сборники.

4.2. Портрет пользователя

При рассмотрении сферы применения продукта были выделены две основные категории потенциальных пользователей, которые были обозначены следующим образом: **Авторы песен и подборов** и **Исполнители песен и организаторы мероприятий**.

Авторы песен и подборов

К данной категории относятся пользователи, создающие записи песен. Авторы песен создают записи песен собственного сочинения, авторы подборов создают записи для песен, не имеющих таковых.

В рамках работы под подбором понимается запись песни с аккордами осуществлённая независимо от автора песни.

Основным продуктом деятельности этой категории пользователей являются как записи песен, так и свёрстанные сборники (например, тематический сборник песен одного автора).

Исполнители песен и организаторы мероприятий

К данной категории относятся пользователи, создающие только сборники песен. Под организаторами мероприятий автор работы понимает организаторов мероприятий, включающих в себя совместное исполнение песен группой людей. Для этой категории пользователей важна предварительная подготовка сборников песен, так как она позволяет удостовериться в том, что все участники знакомы или имеют возможность ознакомиться с текстом, который будет для всех одинаков, без временных затрат на самостоятельный поиск текстов.

Для исполнителей песен сборники полезны с точки зрения сохранения исполняемых ими песен в качестве подсказки или для возможного предоставления другим людям для совмсестного исполнения.

Ключевым отличием между этими двумя категориями является тот факт, что авторы песен и подборов создают запись песни с нуля, вводя (или копируя) текст и самостоятельно расставляя аккорды. Напротив, исполнители песен и организаторы мероприятий чаще при вёрстке используют готовые записи песен, ограничиваясь корректурой текста песен и перерасстановкой уже введённых аккордов.

4.3. Итоговые требования

Таким образом были сформулированы следующие требования к продукту:

- Наличие возможности создания, сохранения и загрузки песен с буквенно-цифровой нотацией аккордов;
- Наличие возможности экспорта введённых и сохранённых песен в пригодные к печати форматы;
- Наличие возможности импорта в редактор песен, записанных в текстовом формате;
- Максимальное облегчение корректуры импортируемых и записываемых песен.

5. Архитектура

5.1. Главный сценарий использования продукта

В результате формирования требований был выработан следующий главный сценарий использования продукта:

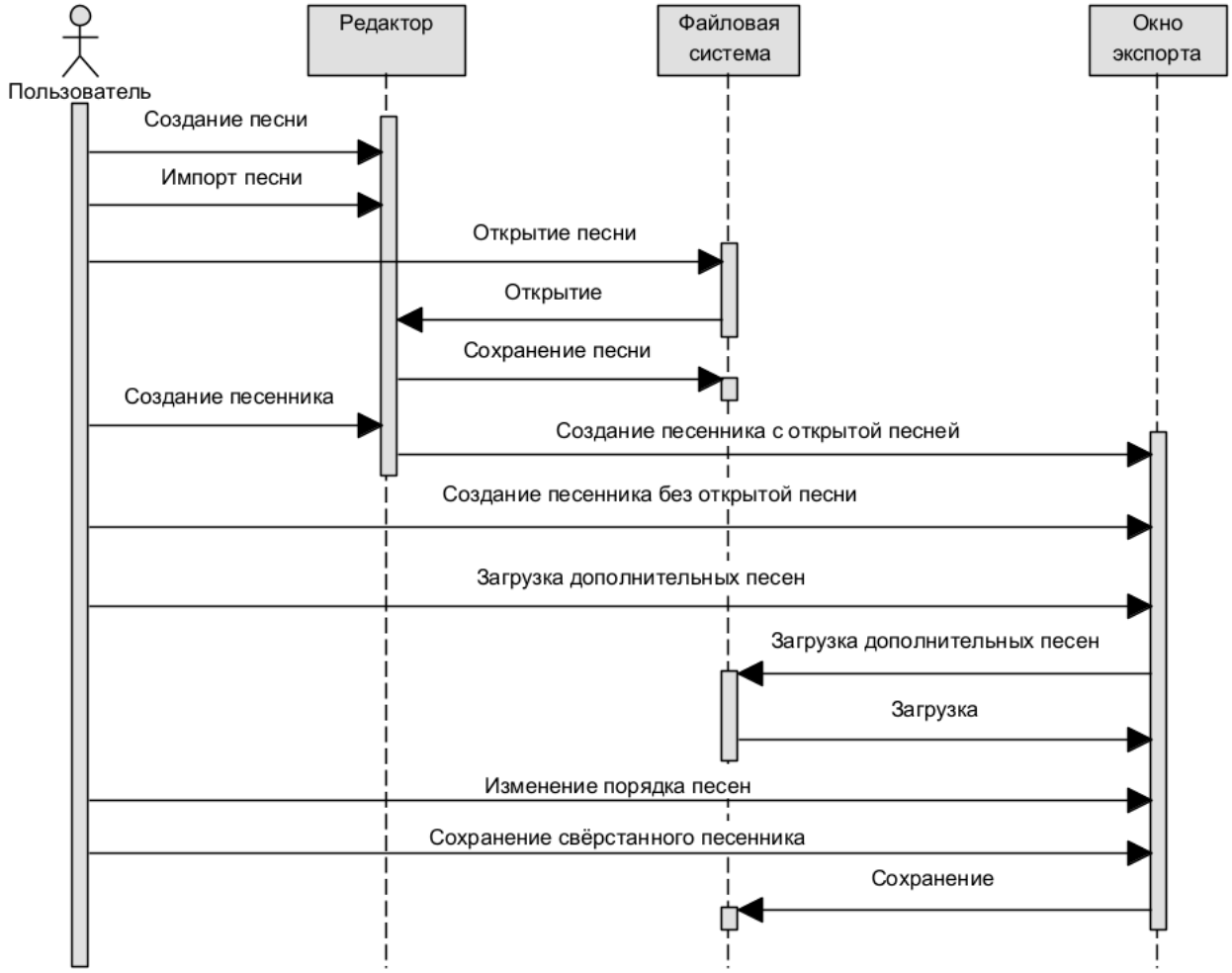

Рис. 1: Главный сценарий использования продукта

Рассмотрим данный сценарий подробнее.

После запуска системы пользователь может выполнить одно из следующих действий:

1. **Создание песни**

   Открывается интерфейс редактора готовый к вводу новой песни;

2. **Импорт песни**
   Открывается интерфейс импорта песни, при помощи которого можно открыть в редакторе песню, сохранённую в формате, отличном от формата «ChordWriter»;

3. **Открытие песни**
   Открывается интерфейс работы с файловой системой, позволяющий открыть для редактирования ранее сохранённую в формате «ChordWriter» песню;

4. **Создание песенника**
   Открывается интерфейс создания песенника.

В свою очередь при открытом интерфейсе редактора пользователь может дополнительно сделать следующее:

1. **Сохранение песни**
   Открывается интерфейс работы с файловой системой, позволяющий сохранить открытую в данный момент в редакторе песню;

2. **Создание песенника**
   Действие аналогично созданию песенника в начале работы, но дополнительно в списке для создания песенника будет присутствовать песня, открытая в редакторе.

Наконец, при взаимодействии с окном экспорта, пользователю доступны следующие действия:

1. **Загрузка дополнительных песен**
   Открывается интерфейс работы с файловой системой, позволяющий добавить в список песни, ранее сохранённые в формате «ChordWriter»;

2. **Изменение порядка песен**
   В рамках интерфейса создания песенника пользователь может изменить порядок выбранных для вёрстки сборника песен;

3. **Сохранение свёрстанного песенника**
   Открывается интерфейс работы с файловой системой, позволяющий сохранить автоматически свёрстанный песенник в формате PDF.

5.2. Стек технологий

В качестве базового стека технологий было принято решение использовать связку **TypeScript** + **Electron** + **React**.

TypeScript, React

TypeScript - язык программирования, разработанный компанией Microsoft. Представляет собой расширение языка JavaScript, в первую очередь добавляющее статическую систему типов. Использование статической системы типов упрощает процесс разработки, снижая объём отладки за счёт устранения ошибок, связанных с несоответствием типов [13].

React - популярная библиотека JavaScript для разработки веб-интерфейсов [11].

Electron

Electron - фреймворк для разработки кроссплатформенных нативных приложений с использований веб-технологий. На момент написания работы Electron является самым популярным фреймворком для таких задач. Использует в работе Node.js[9] для взаимодействия с ОС и движок Chromium для отрисовки интерфейса приложений [1].

React Suite

В качестве основы для разработки визуальной состовляющей интерфейса была использована библиотека компонентов React Suite [8]. Этот выбор обусловлен исключительно субъективными эстетическими предпочтениями автора.

Использование веб-технологий для разработки продукта удобно в первую очередь потенциальной переиспользуемостью компонентов си-

стемы. При необходимости приложение или его элементы с невысокими трудозатратами можно развернуть уже как полноценное браузерное веб-приложение или как составную часть такого приложения. Использование React позволяет переиспользовать отдельные компоненты и решения также и для мобильной разработки с помощью React Native.

5.3. Внутреннее представление песен

Одним из первых этапов в разработке стал выбор модели хранения редактируемых песен. При описании этого этапа хотелось бы отдельно остановиться на анализе свойств редактируемых сущностей и выборе самой модели.

Анализ свойств редактируемых сущностей

В первую очередь был произведён анализ песен с целью выделить свойства, которые можно будет редактировать и определить модель, в которой песни будут храниться.

В результате анализа были выделены следующие составляющие:

- Название;
- Автор;
- Куплеты, припевы и переходные части, содержащие текст;
- Инструментальные части, не содержащие текста;
- Различные примечания.

Первые две составляющие представляют собой обыкновенную строку без переносов, поэтому отдельно на них останавливаться не будем. Рассмотрим подробнее текстовые и инструментальные части.

Текстовые части

Текстовая часть состоит из строчек текста песни, которые могут дополняться или не дополняться аккордами. Помимо этого отдельные фраг-

менты из нескольких строк в песне могут повторяться, что может быть сокращённо записано указанием числа повторов.

Аккорды в каждой строке могут быть расположены над конкретным слогом, а также перед началом строки, если аккорд начинает играться до вступления голоса или после её окончания, если несколько аккородов проигрываются подряд перед началом следующей строки.

Инструментальные части

Под инструментальной частью автором понимается отрывок песни, который играется без сопровождения голосом. Такая часть может состоять из последовательностей аккордов или же более сложной музыкальной линии, которая обычно записывается при помощи гитарных табулатур.

Итоговая модель

На диаграмме на рис. 4 представлено разработанное по итогу анализа

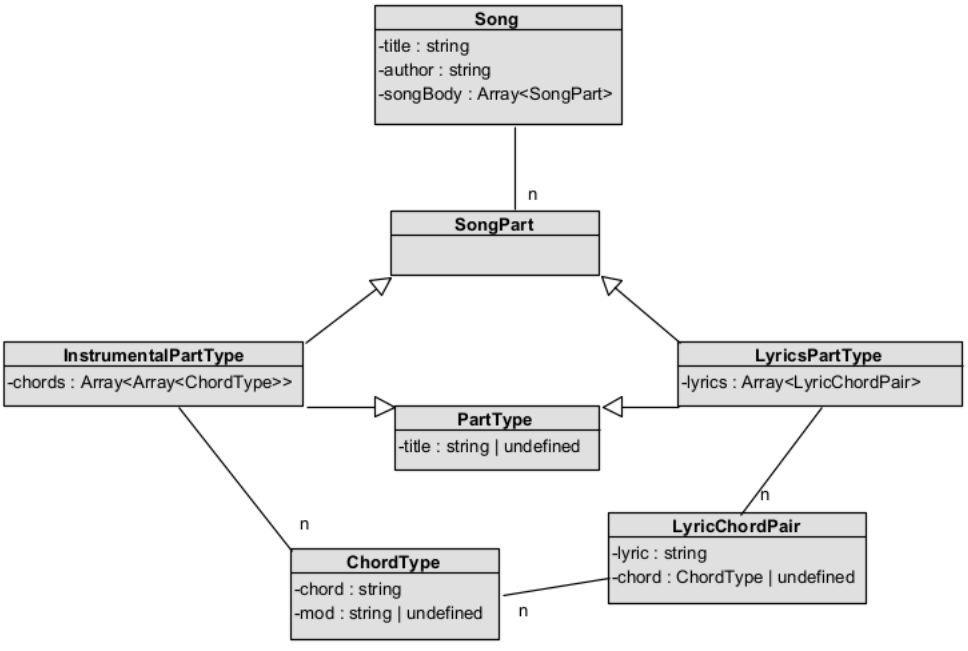

Рис. 2: UML-диаграмма модели данных

внутреннее представление песни:

- Базовый тип `Song` содержит поля `title` и `author` для записи на-

звания и автора песни, а также поле `songBody` в котором содержатся имеющиеся в песне части;

- Тип `SongPart` - тип, который может быть либо `InstrumentalPartType`, либо `LyricsPartType`;
- Тип `PartType` - общий тип, от которого наследуются `InstrumentalPartType` и `LyricsPartType`, содержит в себе общее для них обоих свойство title - опциональное поле строкового типа, в котором содержится название части;
- Тип `InstrumentalPartType` для записи инструментальных частей. Содержит поле `chords` для записи последовательностей аккордов;
- Тип `LyricsPartType` для записи текстовых частей. Содержит в себе поле `lyrics` для записи подстрок строки с привязанными к ним аккордами;
- Тип `LyricChordPair` для записи пар из подстроки и аккорда;
- Тип `ChordType` для записи аккордов. У каждого аккорда может быть его основное обозначение записанное в поле `chord`, представляющее собой основную ноту аккорда, а также некоторый опциональный модификатор, записанный в поле `mod`, который пишется подстрочным шрифтом справа от аккорда.

5.4. Диаграмма компонент

Базовая архитектура приложения во многом была определена используемым стеком. С помощью фреймворка Electron нативные приложения создаются с использованием веб-технологий: языка JavaScript для программирования и комбинации HTML+CSS для вёрстки.

Каждое окно приложения Electron отображается с использованием браузерного движка Chromium. Взаимодействие с ОС обеспечивается через отдельно запущенный процесс, который также занимается созданием окон приложения. В ходе работы приложения может открываться

вплоть до двух окон приложения – **основное окно** и **окно предпросмотра PDF**, а также диалоги взаимодействия с файловой системой.

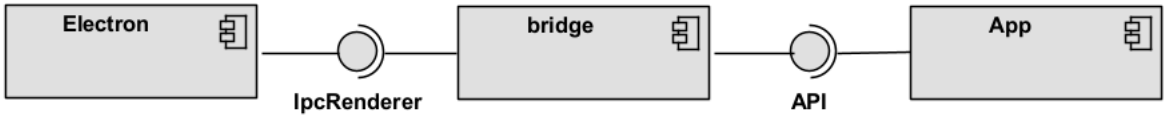

Рис. 3: Диаграмма компонент

Всё взаимодействие с основным процессом Electron обеспечивается через `bridge`, предоставляющий веб-содержимому окон приложения ограниченный API доступа к основному процессу.

Архитектура веб-составляющей строилась на стандартных современных принципах проектирования React - композиции функциональных компонентов.

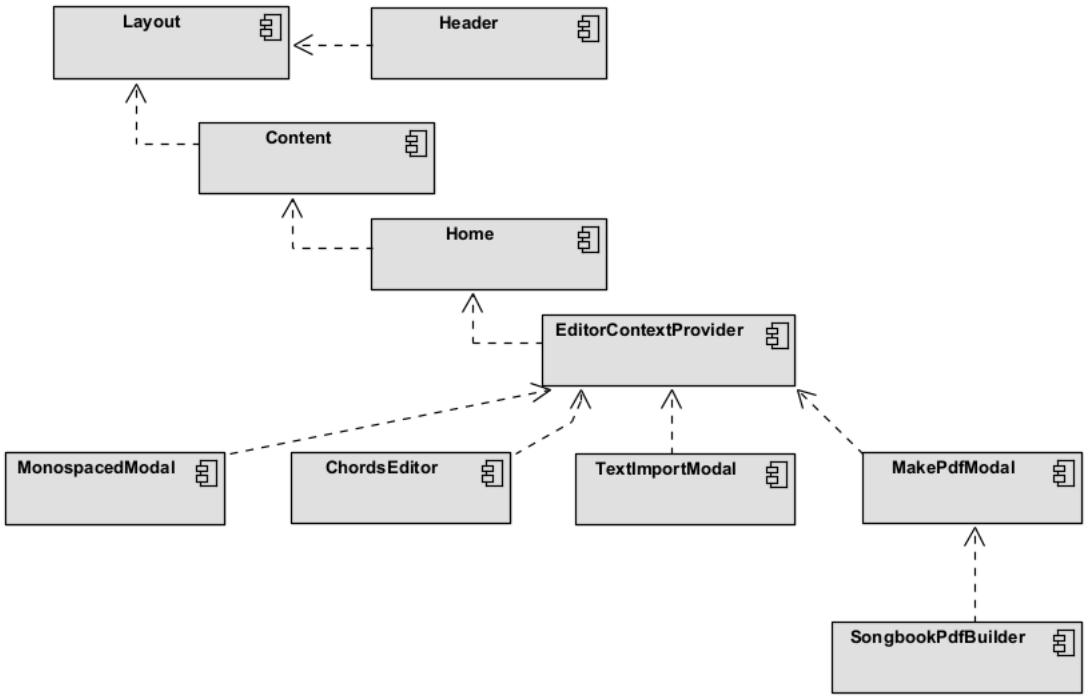

Рис. 4: Диаграмма компонент веб-составляющей

Точкой входа основного окна приложения служит компонент `App` с вложенным компонентом `Layout`, отвечающим за выведение основных элементов окна. В компоненте `Layout` содержатся базовые компоненты:

- `Header`, обеспечивающий отображение заголовка и элементов управления окном;

- `Content`, который отвечает за отображение содержимого окна.

В компоненте `Content` содержится компонент `Home` в свою очередь содержащий в себе все компоненты основного окна редактора. Этот компонент содержит в себе следующие:

- `ChordsEditor`
  Интерфейс редактора песен. Данный компонент отвечает за отображение редактируемой песни и элементов управления. Структура поддерева компонентов-потомков строится на основе модели хранения песен;

- `TextImportModal`
  Модальное окно импорта песен из текстового формата;

- `MonospacedModal`
  Модальное окно вывода введённой песни в моноширинный текстовый формат;

- `MakePdfModal`
  Модальное окно создания песенного сборника. Содержит в себе также компоненты отрисовки окна вёрстки и предпросмотра PDF.

Отдельно обратим внимание, что модальные окна в данном случае являются частью веб-составляющей приложения и управляются целиком внутри главного окна без взаимодействия с Electron.

Все перечисленные компоненты, вложенные в `Home` дополнительно обёрнуты в компонент `EditorContextProvider` для обеспечения доступа к внутреннему состоянию редактора.

При сохранении песенного сборника, он выводится в отдельном окне для вёрстки и вывода предпросмотра. Это отдельное окно приложения, создаваемое при помощи средств Electron. Вместе с тем, его веб-составляющая управляется из дерева компонентов React основного ок-

на с помощью компонента `SongbookPdfBuilder` (см. подробнее в разделе 6.3).

5.5. Интерфейс

Основным свойством разрабатываемого редактора изначально была заложена визуальность или же WYSIWYG (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et). Исходя из этого и принималось большинство дальнейших решений.

Окно редактора состоит из элементов управления окна, меню и окна редактора с отображаемой редактируемой песней (рис 5).

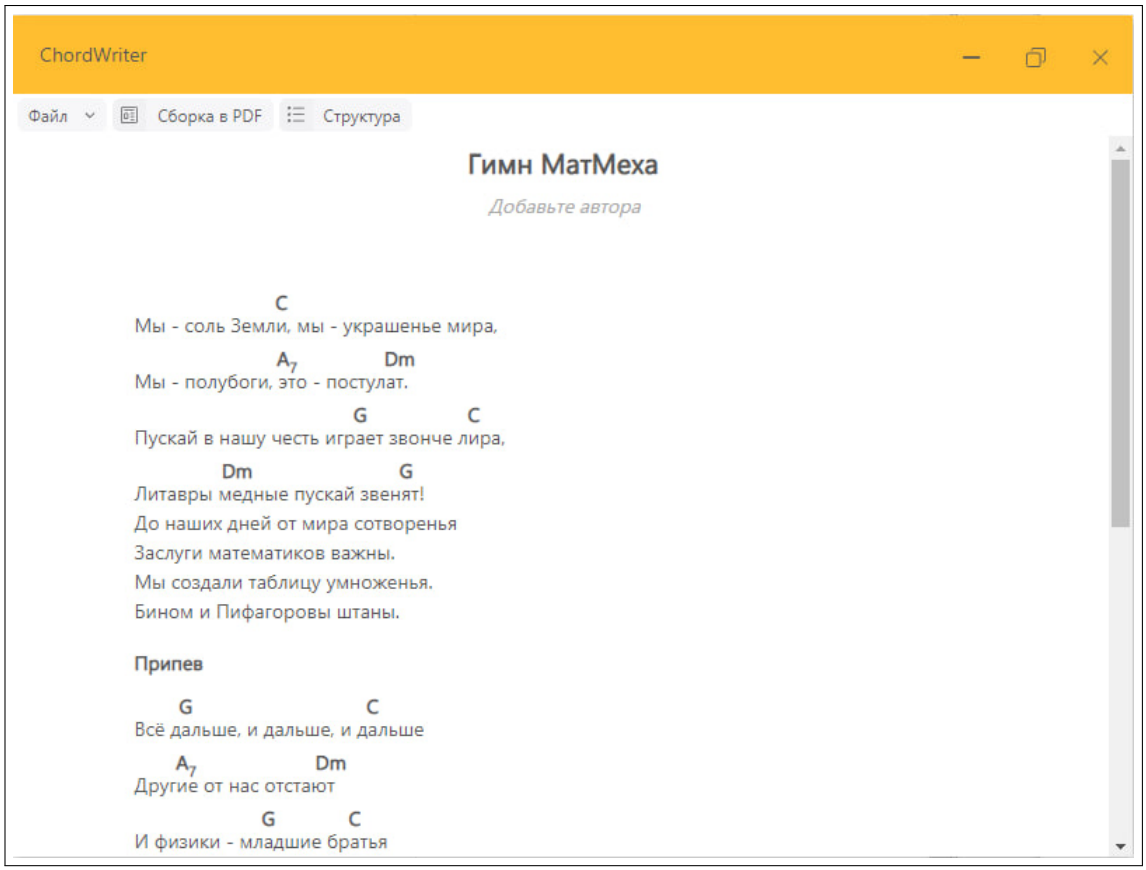

Рис. 5: Интерфейс приложения

Редактирование структуры песни

Как ранее было упомянуто в подразделе 5.3, во внутреннем представлении песня делится на отдельные части различных типов. Отображение

структуры песни и элементов её редактирования переключается нажатием на кнопку ”Структура” в верхней панели (рис. 6).

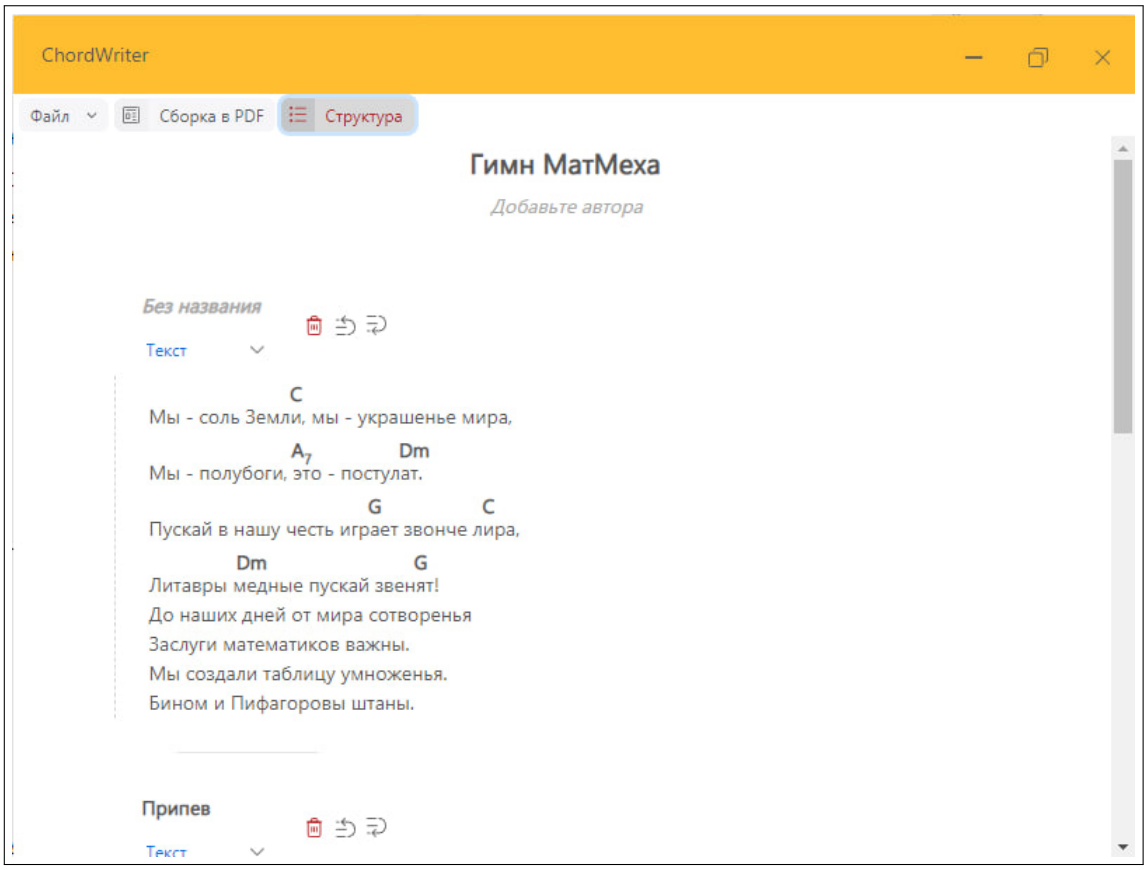

Рис. 6: Интерфейс приложения с отображёнными элементами редактирования структуры

Редактирование текста песни и аккордов

Так же из соображений наглядности было решено сделать элементы управления отдельными строками видимыми только при наведении на строку (рис. 7). При наведении появляется возможность перейти в режим редактирования строки (рис. 8), удалить строку, добавить строку выше или ниже.

Рис. 7: Элементы управления строкой

Добавление аккордов осуществляется при нажатии на букву/символ, над которым пользователь хочет добавить аккорд. (рис. 9)

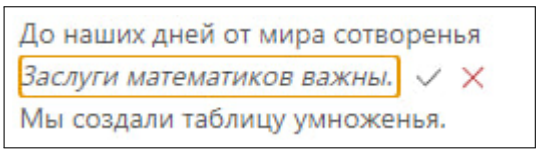

Рис. 8: Строка в режиме редактирования текста

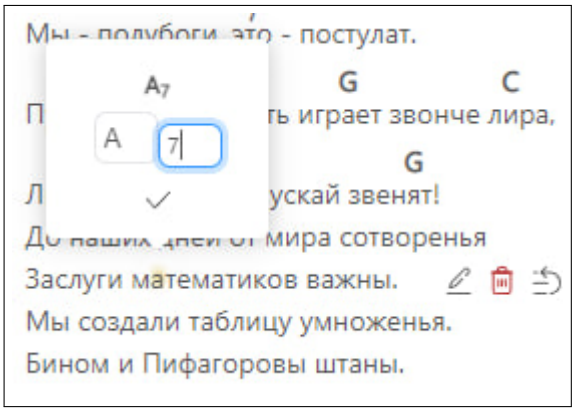

Рис. 9: Элемент добавления/редактирования аккорда

Модальные окна импорта из текстового формата, создания песенника (сборки в PDF) вызываются из меню или с начального экрана приложения.

Импорт песен из текстового формата осуществляется через соответствующее модальное окно (рис. 10) с помощью простой вставки.

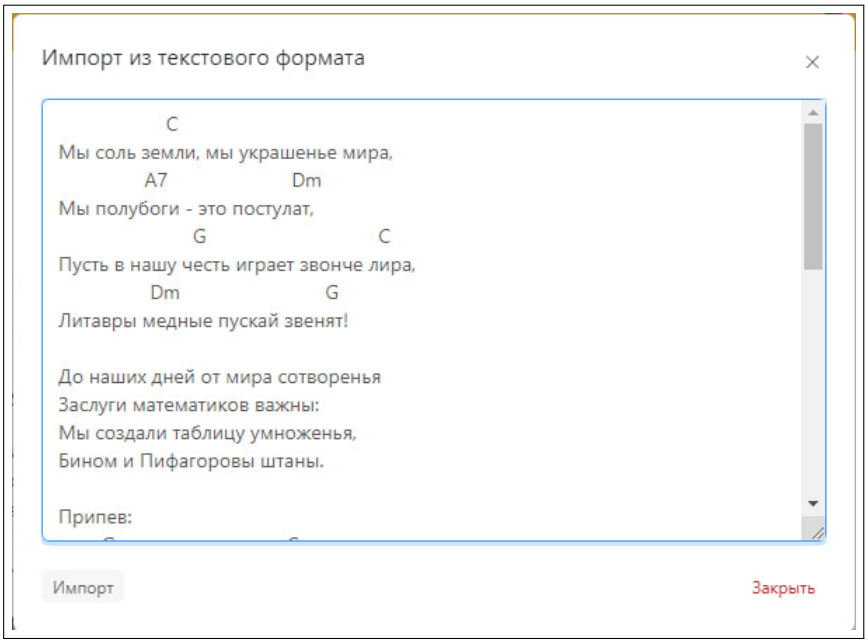

Рис. 10: Модальное окно импорта из текстового формата

Создание песен производится через соответствующее модальное ок-

но (рис. 11). Оно допускает добавление/удаление песен и изменение порядка, при подтверждении вывода вызывается окно предпросмотра (рис. 12).

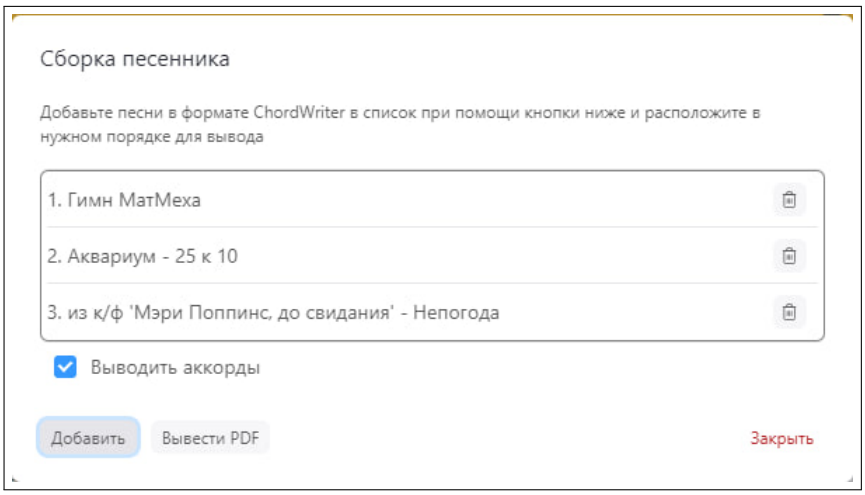

Рис. 11: Модальное окно сборки в PDF

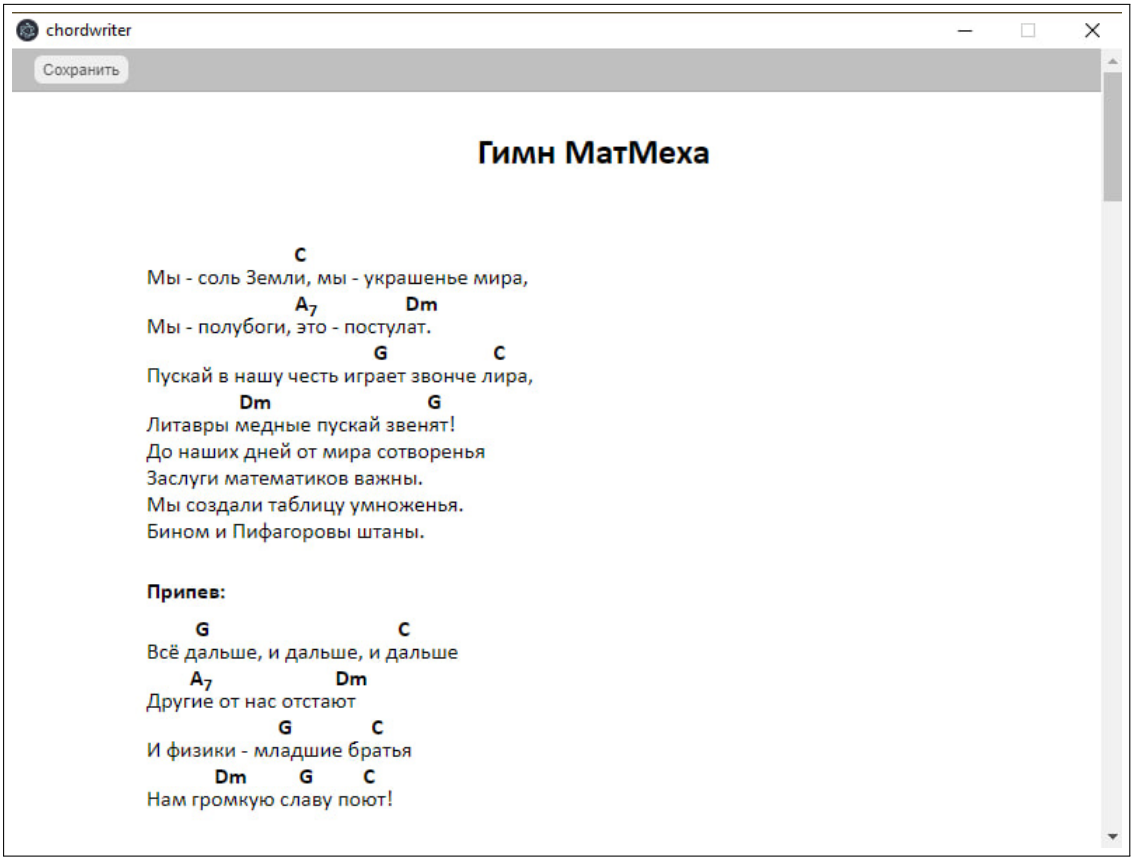

Рис. 12: Окно предпросмотра свёрстанного песенника

6. Особенности реализации

6.1. Сохранение изменений во внутреннем представлении

Отдельным вопросом стала организация сохранения изменений. Изначально был выбран максимально прямолинейный подход с хранением на уровне рендера компонента типа `Song` объекта типа `Song`, с передачей вниз по дереву компонентов анонимных функций, меняющих соответствующий компоненту узел изначального объекта. Очень быстро стало очевидным, что такой подход обладает рядом недостатков: помимо общеизвестного 'Prop drilling', полученную систему было очень сложно поддерживать и править. Было принято решение перейти к выделенному управлению состоянием.

Для обозначенной задачи было рассмотрено три различных варианта решения:

- **Redux**

  Популярная библиотека для управления состоянием в веб-приложениях JavaScript с широким функционалом[12]. Обычно использование Redux рекомендуется при наличии в приложении сложного внутреннего состояния с сильной связностью. Широта функционала Redux компенсируется относительной сложностью его настройки. Было принято решение, что использование этой библиотеки не оправдано.

- **Formik**

  Дополнительная библиотека для React для упрощения работы с формами[2]. Библиотека позволяет для каждой формы задать контекст, в котором будет храниться текущее состояние формы. Если подключить все поля в форме к Formik, то библиотека возьмёт всю работу по обновлению состояния на себя. Управление состоянием в Formik строится с помощью инструментария React.

- **Собственная реализация**

  Реализация управления состоянием на основе хуков React `useContext`[16] и `useReducer`[17].

В конечном итоге было остановиться на собственной реализации. Сделано это было во избежание разрастания дерева зависимостей проекта.

Реализация управления состоянием редактора
Для управления состоянием была использована комбинация хуков React Был заведён комплект команд: `setValue`, `addArrayValue` и `removeArrayValue`, принимающих путь, по которому необходимо внести изменение и, при необходимости, значение. При использовании такого подхода по дереву компонентов теперь необходимо спускать только путь к задаваемому компоненту.

Также использование такого подхода впоследствии позволило без особых издержек внедрить функцию отмены последнего действия.

6.2. Сохранение и загрузка сохранённых песен

Сохранение и загрузка файлов с компьютера пользователя в приложение, как и прочие взаимодействия с ОС, в целях безопасности, осуществляется через фоновый процесс Electron. Для взаимодействия с ним из окон используются "мосты". В процессе Electron есть модуль `ipcMain`, в котором можно задавать обработчики событий. С помощью "моста" задаётся доступный окну приложения API, через который можно либо отправить сигнал основному процессу, либо вызвать в нём метод, чтобы получить какой-либо ответ. API из "моста" обращается именно к тем событиям, обработчики которых заданы в `ipcMain` [14].

Для сохранения песен было принято решение просто выгружать объект типа `Song` в `.json`-файл. Выбор в пользу такого решения вместо использования, например, формата ChordPro, упомянутого в подразделе 3.2, был обусловлен большей простотой обработки.

При загрузке песни из файла осуществляется проверка корректности. Сначала встроенными средствами JavaScript проверяется корректность формата JSON, затем уже внутри программы проверяется корректность всех полей согласно модели.

6.3. Реализация экспорта в PDF

Для вывода сборников и песен было решено выбрать формат PDF. Этот выбор был обусловлен:

- широкой распространённостью формата;
- простотой реализации вёрстки в сравнении с редактируемыми текстовыми форматами;
- возможностью воспользоваться для вывода встроенными средствами Node.js и языка JavaScript, в отличие от таких форматов как, например, TEX.

Для решения этой задачи было выявлено два пути - формирование документа из блоков с использованием сторонних библиотек и сохранение в PDF страниц, свёрстанных с помощью HTML.

Реализация первого подхода потребовала бы подключение сторонних библиотек для формирования документа или самостоятельной разработки. Этот подход был отвергнут как избыточный и излишне трудоёмкий.

Для реализации второго подхода фреймворк Electron предлагает встроенные средства, которыми и было решено воспользоваться.

Для вывода HTML-страницы с свёрстанной песней было решено использовать отдельное окно, в котором был бы отображена подготовленная HTML-страница с заданными размерами и только необходимой для вывода информацией. Такой подход был выбран, так как отображение непосредственно страницы редактора потребовало бы временного сокрытия управляющих элементов, что неоправданно трудоёмко.

Для создания подобного окна были использованы средства языка JavaScript, фреймворка Electron и библиотеки React, уже используемых в проекте.

После вызова функции «Сборка в PDF» из меню приложения открывается модальное окно, позволяющее добавить в список песни, сохранённые в формате редактора в памяти компьютера пользователя. Если в окне редактора уже присутствует набранная песня, то она также добавляется в список. В списке пользователь, помимо добавления, имеет возможность задать порядок песен в полученном сборнике, удалить ранее добавленные в список песни.

После нажатия кнопки «Вывести PDF» программа открывает дополнительное окно с свёрстанным при помощи HTML в виде, без заметных изменений масштабируемом при выводе к размеру листа «A4».

Для открытия окна используется встроенная в JavaScript команда `window.open()`. Эта команда перехватывается основным процессом Electron и создаёт новое окно приложения. Ссылка на открытое окно, полученная после выполнения команды, используется для создания портала React.

Порталом в библиотеке React называется функция служащая для рендеринга потомков текущего компонента React. Портал позволяет работать с содержимым нового окна так, словно это компонент в структуре DOM текущего окна[19].

При нажатии на кнопку «Сохранить», кнопка скрывается, а в созданном окне вызывается встроенная функция вывода в PDF окон Electron. После создания PDF-документа, вызывается диалог сохранения файла.

6.4. Реализация импорта из текстового формата

В качестве приоритетного импортируемого формата, было решено использовать наиболее распространённый текстовый формат.

Текстовый формат записи нигде не стандартизирован, однако обладает некоторыми общепринятыми свойствами, такими как:

- Последовательное расположение строк, содержащих аккорды и

содержащих текст. При этом аккорды в строке позиционируются над строкой с текстом так, чтобы визуально располагаться над слогом, при пропевании которого начинает исполняться соответствующий аккорд;

- Разделение частей пустыми строками

Данные свойства не являются обязательными, однако широко распространены.

Было принято решение реализовать максимально базовый текстовый анализатор, который сможет распознавать аккорды и расставлять их в соответствии с их положением в тексте.

Анализатор не накладывает существенных ограничений на форматирование вводимого текста и не производит его валидации, в связи с отсутствием стандартизированного формата.

Реализация импорта была целиком выполнена с помощью встроенных средств React и JavaScript. В ходе разработки был частично переиспользован код, используемый для экспорта в текстовый формат.

6.5. Доработки редактора песен

Для дальнейшего упрощения ввода и редактирования песен был осуществлён ряд доработок редактора для упрощения редактирования и ввода новых песен:

- Добавлена возможность экспорта в моноширинный формат;
- Добавлено автоматическое выделение новых частей при наличии пустых строк в вставке и при сохранении пустых строк;
- Добавлена возможность перестановки аккордов перетаскиванием.

Остановимся подробнее на последнем пункте.

Эта функциональность востребована, так как позволяет существенно упростить корректуру импортированных песен. На момент начала

работы для перестановки аккорда в строке следовало его удалить, после чего создать на новом месте. В свою очередь корректура расстановки аккордов - достаточно частый кейс при редактировании сборников песен, так как зачастую авторы подборов не уделяют достаточного внимания расстановке аккордов. В подборе число аккордов может составлять до нескольких десятков и более. Требовавшееся ранее пересоздание каждого аккорда в таком случае оказывается слишком трудоёмким.

Для реализации drag-and-drop было решено использовать стороннюю библиотеку. Обзор существующих библиотек показал, что наиболее популярными являются следующие[15]:

- react-beautiful-dnd[4]
- react-dnd[10]
- react-draggable[5]

Из указанных библиотек в итоге была использована **react-dnd**. Данная библиотека позволяет добавить drag-and-drop к существующим компонентам React в качестве расширения их функциональности.

Для добавления drag-and-drop были введены дополнительные компоненты для перетаскиваемых аккордов и для букв, на которые может быть установлен аккорд. Соответствующая функциональность добавляется при помощи средств **react-dnd**.

6.6. Настройка CI/CD

В ходе работы разработка велась при помощи GitHub. Для предоставления пользователям возможности использовать актуальную версию приложения и для упрощения и повышения стабильности разработки была произведена настройка CI/CD (Непрерывная интеграция/-Непрерывное развёртывание).

Интеграция и развёртывание были настроены и производятся с помощью GitHub Actions[6]. GitHub Actions – мощный инструмент для

настройки и выполнения текущих регулярных задач в ходе разработки, в том числе таких интеграция и развёртывание, использующий для настроек язык YAML.

Автоматическое развёртывание было преднастроено для создания совместимых с ОС Windows 10 и выше с 64-разрядной архитектурой бинарных файлов. Для упаковки установочных файлов использовался комплект инструментария Electron Forge[3].

7. Апробация

В ходе разработки приложения трижды производилась апробация текущей версии.

Апробация производилась следующим образом:

- Ручное тестирование функциональности путём полноценного использования продукта;
- Ручное тестирование с привлечением посторонних лиц из числа знакомых автора в следующем порядке:
  - Участнику тестирования передаётся собранный дистрибутив приложения, а также предоставляется доступ к заранее набранным автором песням;
  - Участник может задавать вопросы по порядку использования продукта;
  - После тестирования у участника собирается обратная связь на предмет предложений по доработкам и обнаруженных дефектов.

При разборе полученной обратной связи составлялся приоритезированный список исправлений и доработок, при необходимости корректировались требования. Среди прочих, в результате анализа обратной связи были произведены следующие доработки и приняты следующие решения:

- Добавлена перестановка аккордов перетаскиванием;
- Улучшена интуитивность редактора за счёт изменения используемых иконок, изменения поведения элемента редактирования строки, создания новой строки при сохранении текущей нажатием клавиши **Enter** и т.п.;
- Добавлен импорт из текстового формата;

- Принято решение со временем отходить от использования режима ”Структура” в связи с неинтуитивностью его работы.

Заключение

Исходный код проекта доступен в открытом GitHub-репозитории [7].

В результате выполнения выпускной квалификационной работы были достигнуты следующие результаты:

1. Сформулирована решаемая продуктом проблема и составлены портреты пользователей. На основе этого сфомированы требования к функциональности продукта «ChordWriter»;
2. Разработана архитектура приложения;
3. Реализовано сохранение сборников и отдельных песен в формате PDF;
4. Реализована функциональность импорта песен с аккордами в редактор;
5. Произведена апробация со сбором обратной связи;
6. На основе собранных предложений произведены доработки продукта;
7. Произведена повторная апробация.

Список литературы

[1] Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron. — URL: https://www.electronjs.org (дата обращения: 2022.11.20).

[2] Formik: Build forms in React, without the tears. — URL: https://formik.org (дата обращения: 2023.26.05).

[3] Getting Started - Electron Forge. — URL: https://www.electronforge.io (дата обращения: 2023.26.05).

[4] GitHub - atlassian/react-beautiful-dnd: Beautiful and accessible drag and drop for lists with React. — URL: https://github.com/atlassian/react-beautiful-dnd (дата обращения: 2023.26.05).

[5] GitHub - react-grid-layout/react-draggable: React draggable component. — URL: https://github.com/react-grid-layout/react-draggable (дата обращения: 2023.26.05).

[6] GitHub Actions Documentation - GitHub Docs. — URL: https://docs.github.com/en/actions (дата обращения: 2023.26.05).

[7] GitHub репозиторий проекта. — URL: https://github.com/GF2595/ChordWriter (дата обращения: 2022-11-20).

[8] Home - React Suite. — URL: https://rsuitejs.com (дата обращения: 2022.11.20).

[9] Node.js. — URL: https://nodejs.org/en (дата обращения: 2023.26.05).

[10] React DnD. — URL: https://react-dnd.github.io/react-dnd/about (дата обращения: 2023.26.05).

[11] React – JavaScript-библиотека для создания пользовательских интерфейсов. — URL: https://ru.reactjs.org (дата обращения: 2022.11.20).

[12] Redux - A predictable state container for JavaScript apps. | Redux. — URL: https://redux.js.org (дата обращения: 2023.26.05).

[13] TypeScript: JavaScript With Syntax For Types. — URL: https://www.typescriptlang.org (дата обращения: 2022.11.20).

[14] ipcMain | Electron. — URL: https://www.electronjs.org/docs/latest/api/ipc-main (дата обращения: 2022-11-21).

[15] react-beautiful-dnd vs react-dnd vs react-drag-and-drop vs react-draggable vs react-dragula | npm trends. — URL: https://npmtrends.com/react-beautiful-dnd-vs-react-dnd-vs-react-drag-and-drop-vs-reac

[16] useContext – React. — URL: https://react.dev/reference/react/useContext (дата обращения: 2023.26.05).

[17] useReducer – React. — URL: https://react.dev/reference/react/useReducer (дата обращения: 2023.26.05).

[18] Агафошин Пётр Спиридонович. Школа игры на шестиструнной гитаре // Школа игры на шестиструнной гитаре / Ed. by Е.Д. Ларичев.

[19] Порталы - React. — URL: https://ru.reactjs.org/docs/portals.html (дата обращения: 2023.03.27).

[20] Слова, аккорды и история Гимна МатМеха. — URL: http://dm47.com/GimnHistory.html (дата обращения: 2022.11.20).

[21] Урок #5. Музыкальная теория. Буквенно цифровые обозначения аккордов (БЦО) (Синтаксис и правила чтения). — URL: http://www.akkords.ru/lessons/bco.php (дата обращения: 2022.11.20).