Técnicas de Construção de Programas

Trabalho Prático – Fase 3

Semestre 2020/2
Professor Marcelo Soares Pimenta
00302072 - Victória Duarte

REQUISITOS

REQUISITOS FUNCIONAIS	
Descrição	
O usuário deve poder adicionar seu texto a ser musicado tanto digitando o texto pelo teclado, quanto adicionando um arquivo de texto	
O usuário deve poder salvar a música gerada em um .MID/.MIDI	
O valor de volume inicial será aproximadamente 40% do volume do sistema	
O valor de oitava inicial será 1	
O texto a ser convertido em música não deve ter mais de 1000 caracteres	
Caractere	**Ação**
Letra A	Nota Lá
Letra B	Nota Si
Letra C	Nota Dó
Letra D	Nota Ré
Letra E	Nota Mi
Letra F	Nota Fá
Letra G	Nota Sol
Letras a, b, c, d, e, f, g	Se caractere anterior era NOTA (A a G), repete nota; Caso contrário, Silêncio ou pausa
Outras vogais (i/I, o/O, u/U)	Trocar instrumento para o instrumento General MIDI #7 (Harpsichord)

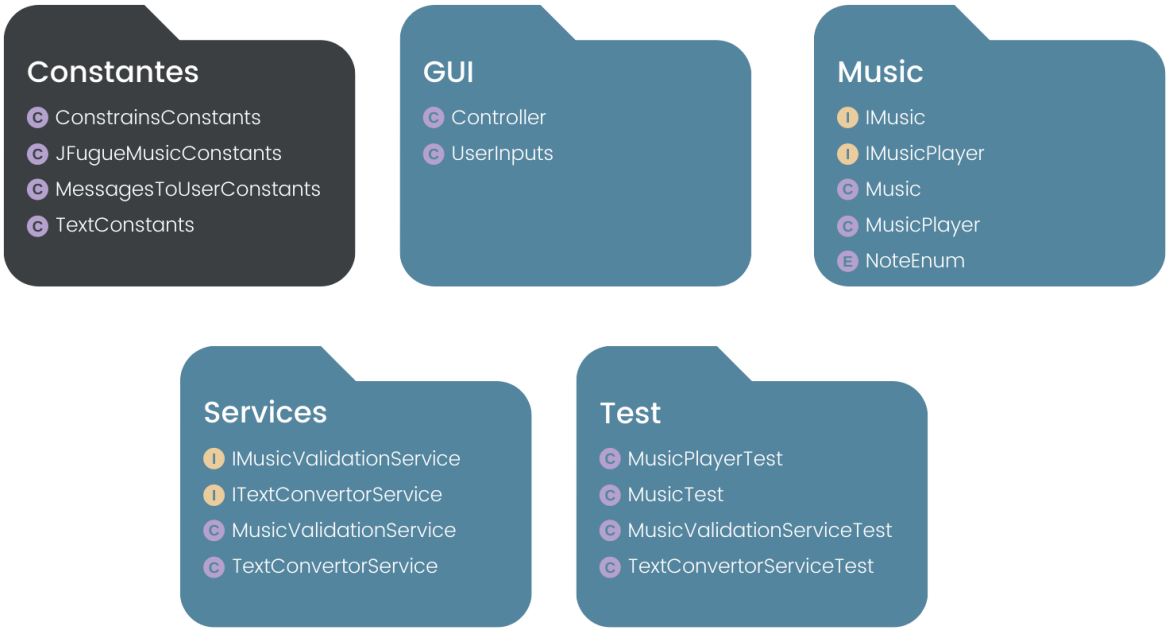
Outras consoantes	Se caractere anterior era NOTA (A a G), repete nota; Caso contrário, Silêncio ou pausa
Caractere espaço	Aumenta volume para o DOBRO do volume; Se não puder aumentar, volta ao volume default (de início)
Caractere NL	Trocar instrumento para o instrumento General MIDI #15 (Tubular Bells)
Caractere !	Trocar instrumento para o instrumento General MIDI #114 (Agogo)
Caractere ? e .	Aumenta UMA oitava; Se não puder, aumentar, volta à oitava default (de início)
Caractere ;	Trocar instrumento para o instrumento General MIDI #76 (Pan Flute)
Caractere ,	Trocar instrumento para o instrumento General MIDI #20 (Church Organ)
Dígito	Trocar instrumento para o instrumento General MIDI cujo número é igual ao valor do instrumento ATUAL + valor do dígito
ELSE (outros caracteres)	Se caractere anterior era NOTA (A a G), repete nota; Caso contrário, Silêncio ou pausa

REQUISITOS NÃO FUNCIONAIS	
Descrição	**Tipo de requisito**
Se, durante a execução do programa, ocorrer um erro, o usuário deve ser comunicado de forma clara sobre o erro	Confiabilidade / Facilidade de Uso
O software deve funcionar em máquinas que utilizam Java na versão 8 ou superior	Portabilidade
O software deve ser fácil e intuitivo de usar	Facilidade de Uso
Caso o usuário não insira um nome de arquivo, será usado um nome padrão no download	Facilidade de Uso
O software deve ser um executável	Implementação
O software deve ser orientado a objetos	Implementação
O software deve funcionar em máquinas que utilizam a versão 8 ou superior do Java	Portabilidade

OPERAÇÕES DO USUÁRIO

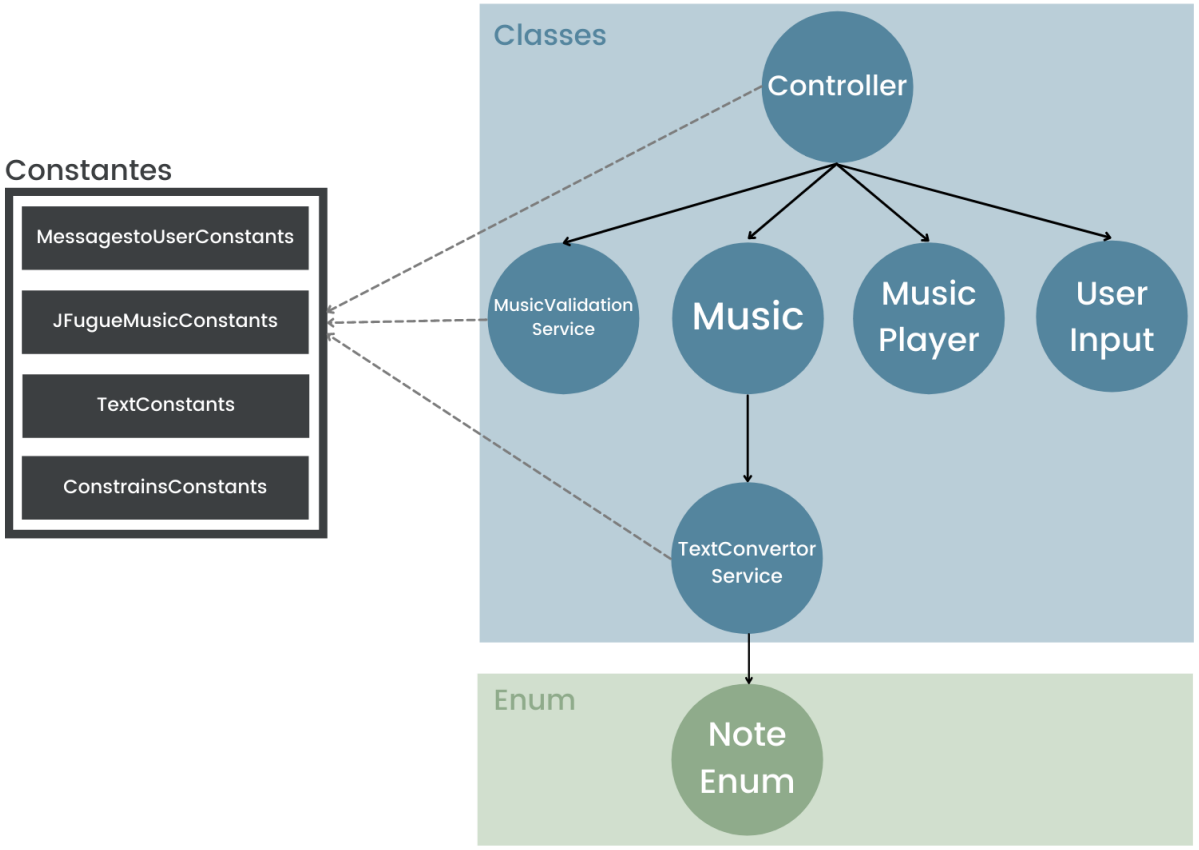
Operação	Alteração
Digitar no campo texto	O texto aparecerá no campo
Escolher instrumento	Ao clicar na aba de seleção (em um primeiro momento vazia), esta abrirá mostrando os instrumentos disponíveis. Quando o instrumento for selecionado, a aba encolhe e o nome do instrumento aparece na tag.
Importar arquivo de texto	O usuário poderá escolher um arquivo de texto de seu computador
Gerar música	Ao clicar, a aba com o play da música aparece na tela
Iniciar música	A música é iniciada
Baixar música	A música é salva no computador

PACOTES



esquema demonstrando a organização dos pacotes do programa

CLASSES



esquema demonstrando o acesso de todas as classes

NotesEnum							
Nota	Dó	Ré	Mi	Fá	Sol	Lá	Si
Valor	0	2	4	5	7	9	11
Métodos							
int getValue() → retorna o valor de uma nota							

Music
Atributos
int initialVolume → volume inicial da música
String music → string em formato musicável pelo JFugue
Métodos
createMusicFromText (String rawTex, int initialBpm, int initialInstrument) → constrói a string musicável

MusicPlayer
Métodos
playMusic (String musicString) → toca a música
boolean downloadMusic (String musicString, String filename) → salva a música retornando se o processo obteve sucesso ou não
Runnable createThread(String musicString) → cria uma thread para a música ser executada

MusicValidationService
Atributos
String errorMessage
Métodos
boolean validateText() → verifica se o texto de input do usuário é válido
boolean validateInstrument() → verifica se o usuário escolheu algum instrumento
int parseBPM() → converte o BPM em um valor numérico, se possível

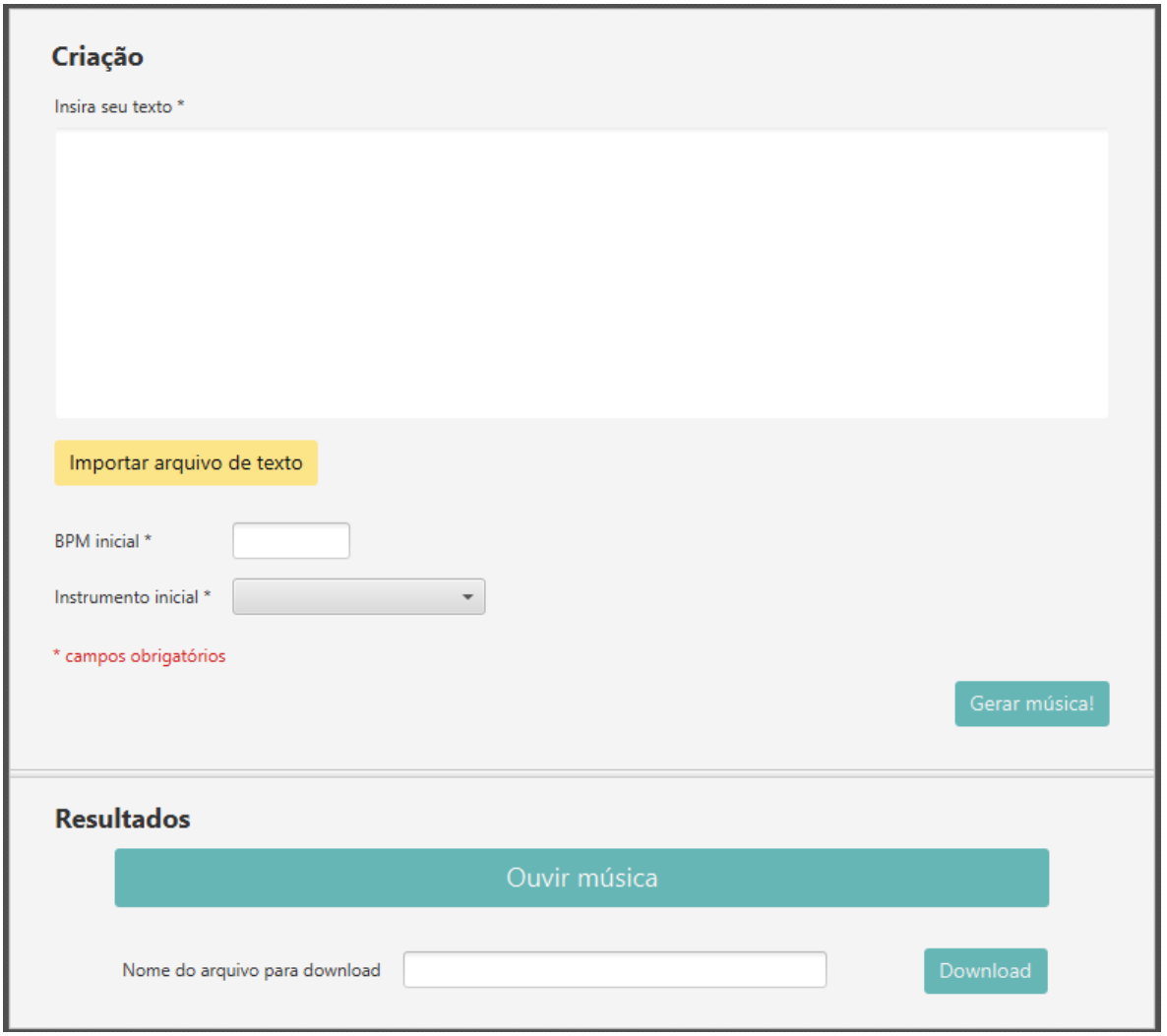
TextConvertorService
Atributos
int currentOctave
HashMap<String, Integer> instrumentHashMap
Métodos
String convert(String raw_text, int initialVolume, int initialBpm, int initialInstrument) → converte o texto fornecido pelo usuário em uma string musicável
boolean verifyNotesExistence(String rawText) → verifica se há caracteres a serem convertidos em notas
String cleansString(String text) → altera alguns caracteres para facilitar a criação da música
String convertArrayToString(ArrayList<String> manipulationArray) → converte um array de strings em uma única string
ArrayList<String> setNotesOnOctaves(ArrayList<String> text) → troca os caracteres que geram notas pelos caracteres de nota do JFugue
ArrayList<String> setInstruments(ArrayList<String> text, int initialInstrument) → troca os caracteres que geram alteração do instrumento pelos caracteres que alteram instrumentos do JFugue
ArrayList<String> setVolume(char[] text) → troca os caracteres que geram alteração no volume pelos caracteres que alteram volume no JFugue

Controller
Atributos
TextField fileInput
TextField bpmInput
TextArea textInput
ChoiceBox choiceBox
Music music
MusicPlayer player
HashMap<String, Integer> instrumentHashMap
Métodos
initialize() → faz a inicialização da interface
OnGenerateMusicButtonClicked() → cria uma música quando o botão de Gerar Música! é pressionado
OnPlayButtonClicked() → toca a música quando o botão de play é clicado
OnDownloadButtonClicked() → faz o download da música quando o botão de Download é clicado
String getFileInput() → pega a informação fornecida pelo usuário no campo de Arquivo
String getBPMInput() → pega a informação fornecida pelo usuário no campo de BPM
String getTextInput() → pega a informação fornecida pelo usuário no campo de Texto
String onSelectInstrument() → pega a informação fornecida pelo usuário na choice box de instrumentos
createErrorAlert(String message) → cria um alerta de erro
createSuccessAlert(String message) → cria um alerta de sucesso
boolean validateUserInputs(MusicValidationService musicValidationService, UserInputs userInputs) → valida os inputs do usuário, verificando se estão corretos
chooseInputFile() → abre uma janela de seleção de arquivos para o usuário
openFile(File file) → abre o arquivo e insere o texto na caixa de texto principal

UserInputs
Atributos
String textInput
String initialInstrument

int initialBPM
Métodos
String getTextInput() → pega o conteúdo do input de texto
String getInitialInstrument() → pega o valor do input de instrumento inicial
int getInitialBPM() → pega o valor do BPM inicial

INTERFACE DE USUÁRIO



interface com o usuário do programa. O programa contém uma única tela principal com todas as funções que o usuário pode fazer.

A interface criada busca ser o mais simples e intuitiva possível. Todas as funcionalidades disponíveis foram dispostas em um único local.

A *GUI* é dividida em duas áreas: a área de criação da música e a área de execução e *download* da música.

Espaço de criação

Nessa área, o usuário insere as informações necessárias para a criação da música. Todos os campos desta área são obrigatórios, marcados por um asterisco (*).

Para inserir o texto que será transformado em música, o usuário também tem a opção de selecionar um arquivo de texto, clicando no botão “Importar arquivo de texto”. Selecionado o arquivo, o texto contido nele é colocado no campo de texto principal, onde é possível modificá-lo.

Espaço de execução de download

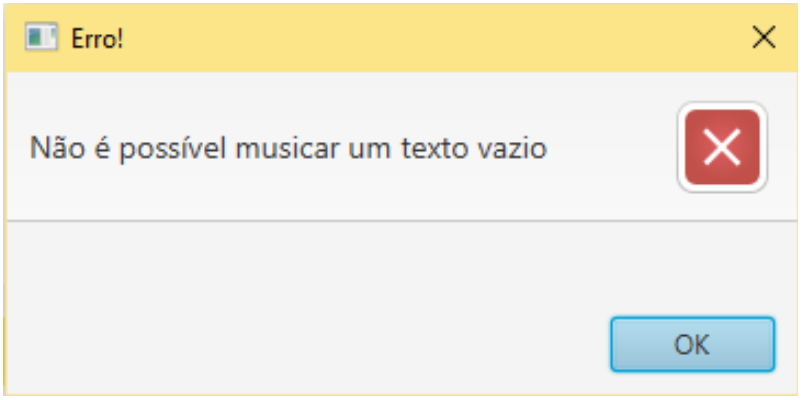
O usuário pode ouvir e fazer o download da música gerada.

O campo para inserir o nome do arquivo de *download* é opcional. Caso o usuário deixe esse campo em branco e, depois de gerar a música, deseje fazer o *download* do resultado, um nome *default* é usado.

Avisos ao usuário

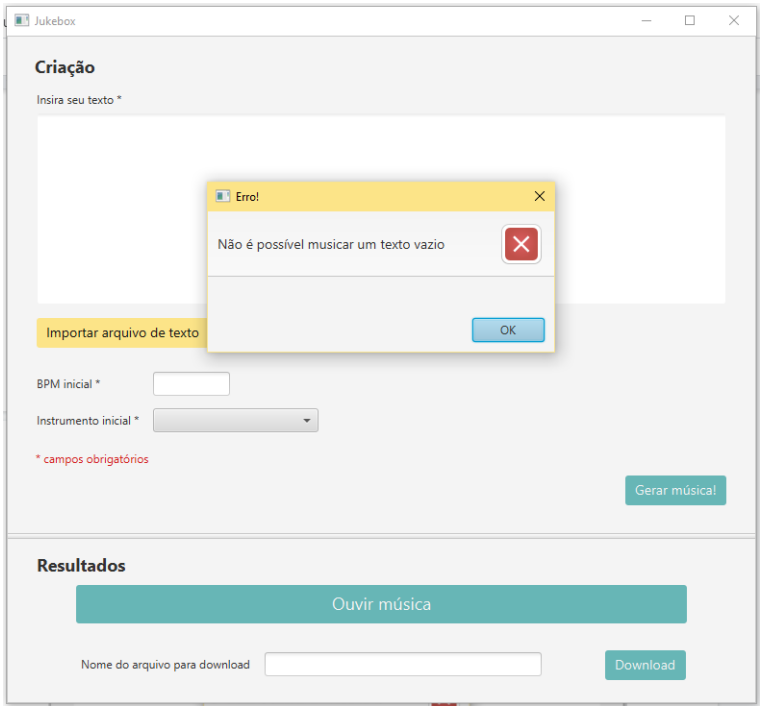
São emitidos alertas ao usuário em casos de dados faltantes ou incorretos, ações que ainda não possam ser realizadas ou sobre resultado na criação da música ou *download*.

Todos os avisos seguem um mesmo formato, mudando apenas a mensagem que informa o acontecimento em questão.

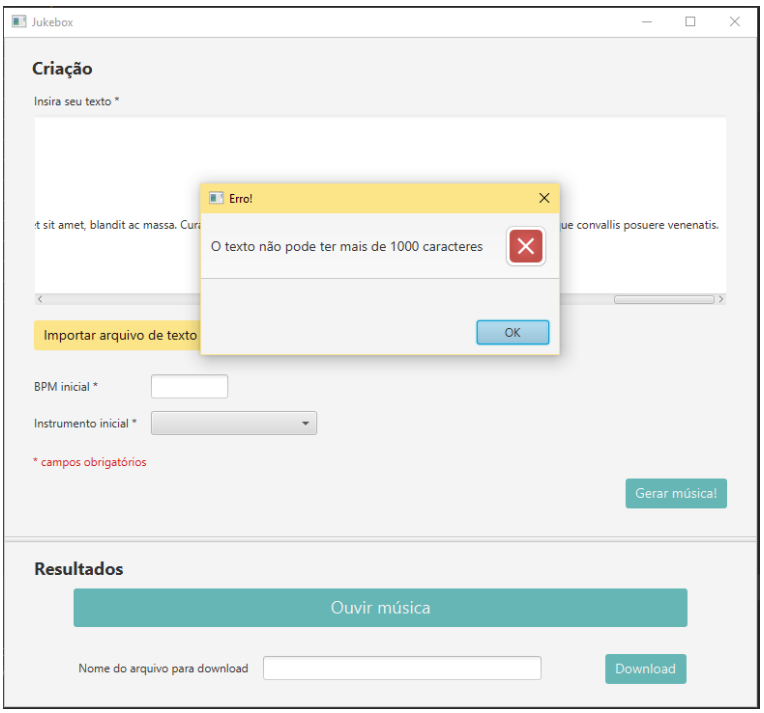


Alerta padrão com um aviso ao usuário. O título indica se houve um erro ou um sucesso na ação tentada e a mensagem especifica o resultado.

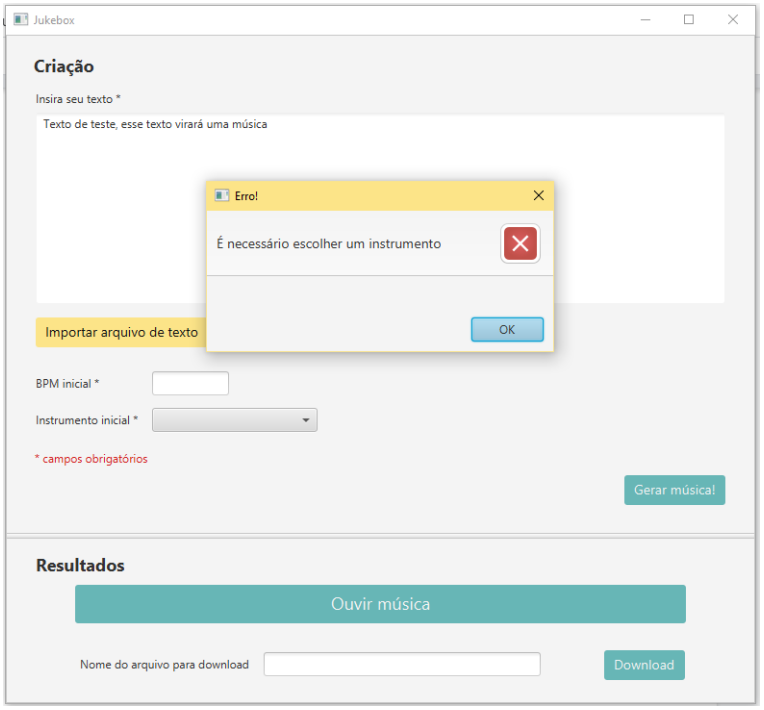
Alertas possíveis



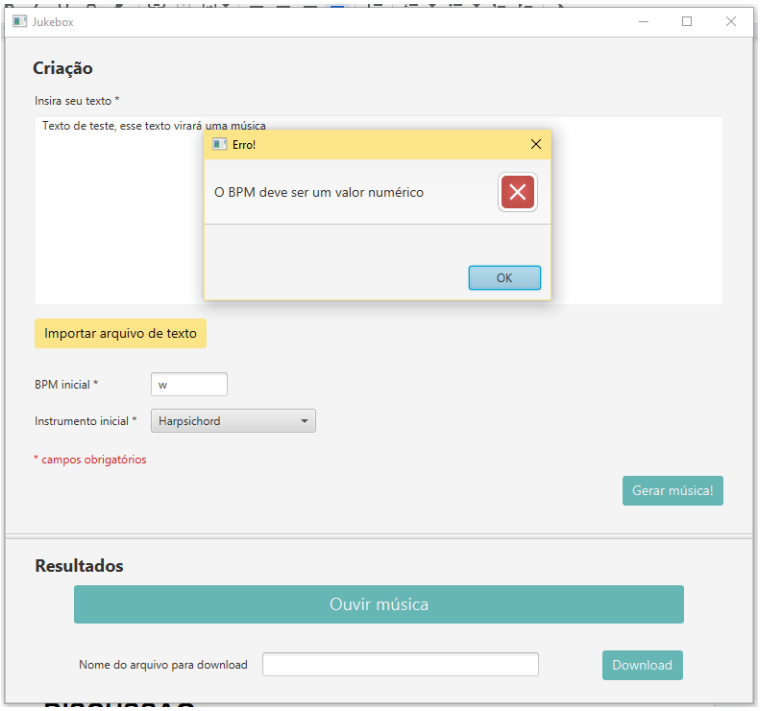
Mensagem de erro ao encontrar o campo de texto principal vazio



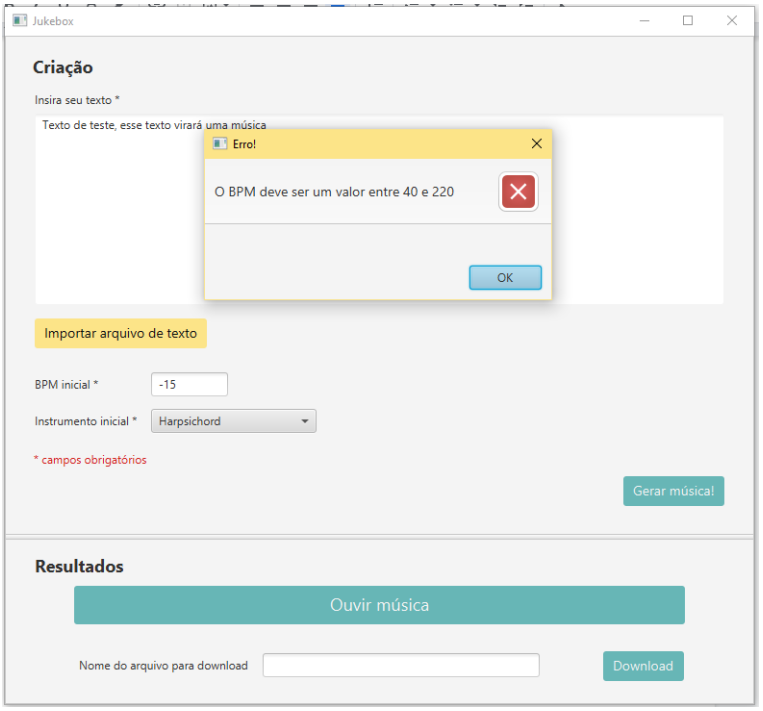
Mensagem de erro ao encontrar o campo de texto que passe do limite de 1000 caracteres



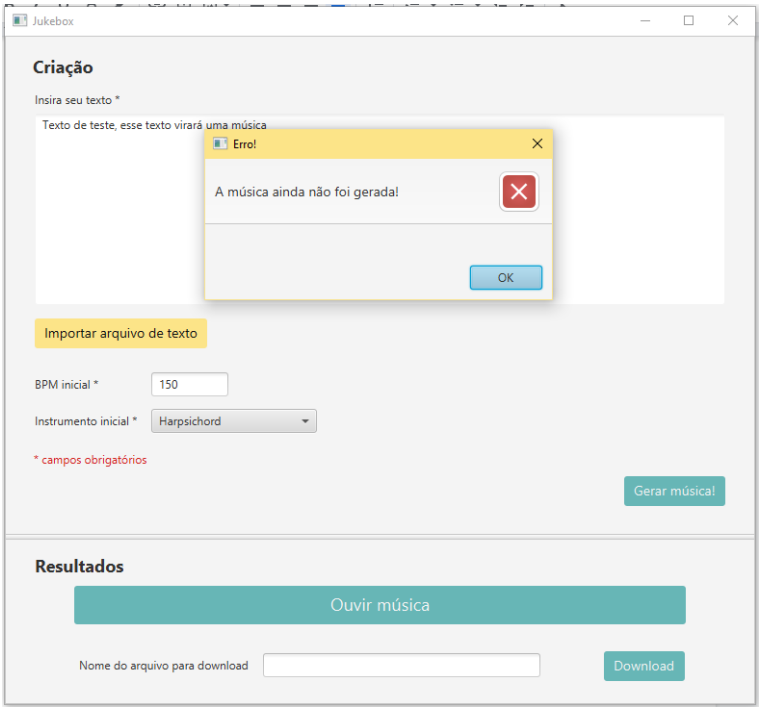
Mensagem de erro ao encontrar o campo de instrumento inicial vazio (sem nenhum instrumento selecionado)



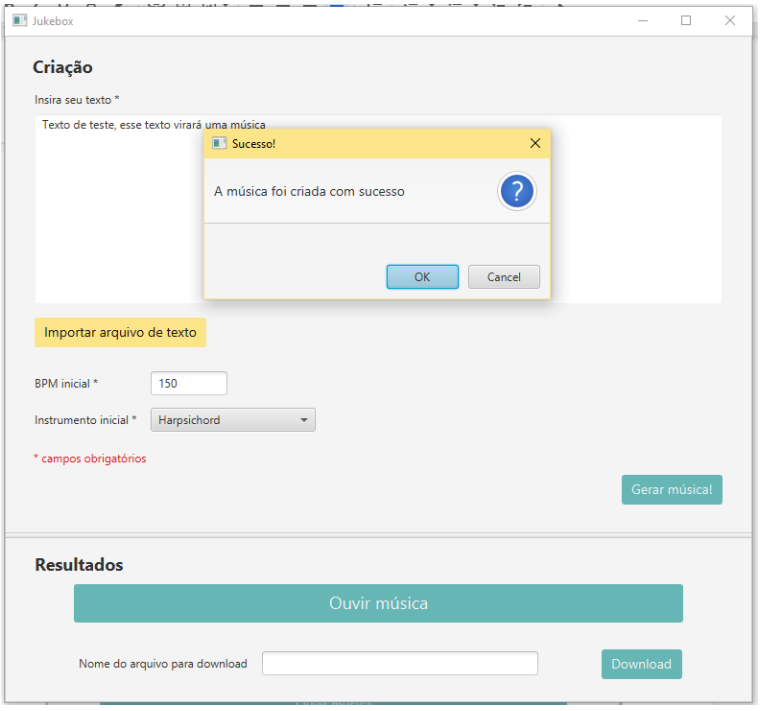
Mensagem de erro ao encontrar o campo de BPM inicial vazio ou com um valor não numérico



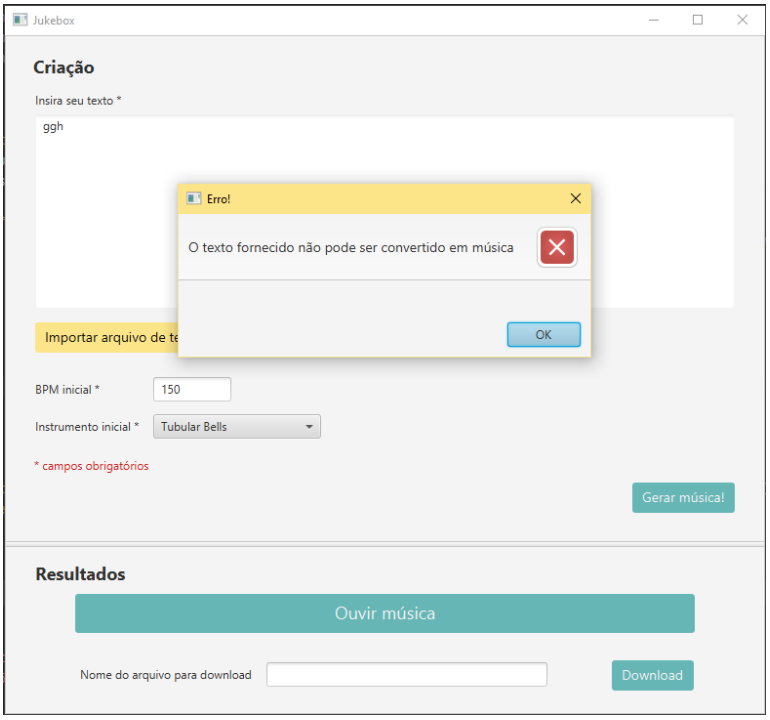
Mensagem de erro ao encontrar o campo de BPM inicial com um valor numérico menor que o valor mínimo ou maior que o valor máximo possível



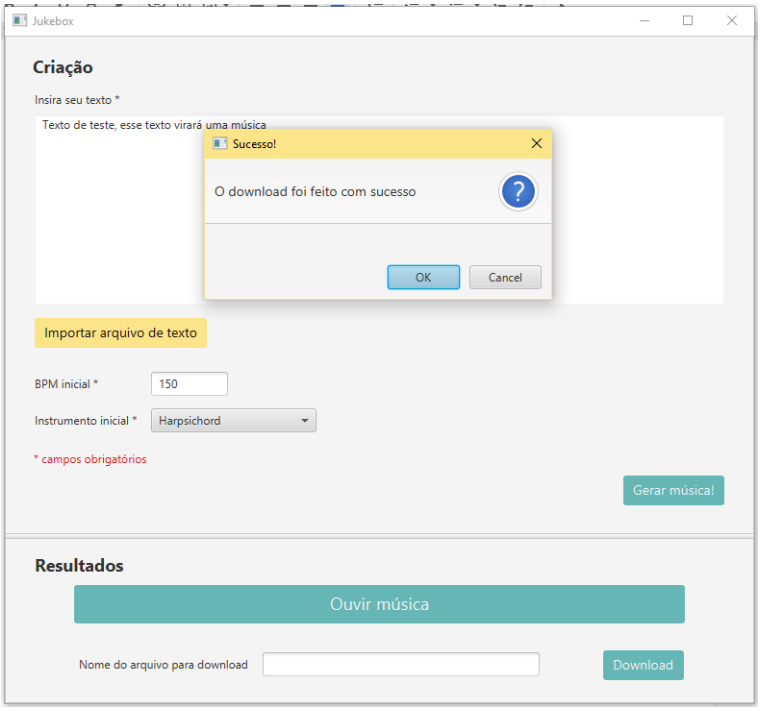
Mensagem de erro ao tentar ouvir ou fazer download de uma música não gerada



Mensagem de sucesso ao gerar a música corretamente



Mensagem de erro quando não é possível converter o texto em um texto musicável



Mensagem de sucesso ao fazer o download da música corretamente

DISCUSSÃO

A mudança nos requisitos não acarretou em mudanças em muitos lugares. Na verdade, houve apenas mudanças em duas classes e dois enums: *Controller*, *TextConverterService*, *NoteEnum* e *InstrumentEnum*.

A classe *Controller* precisou apenas de um método para lidar com o botão de importar um arquivo texto e mais um método auxiliar que lidasse com a abertura e *display* de conteúdo do arquivo.

No caso dos *Enums*, no *NoteEnum* foi removido o método *getRandomNote()* enquanto o *InstrumentEnum* foi totalmente excluído. Em ambos os casos, a deleção veio da falta de necessidade de conter as estruturas.

A classe *TextConverterService*, no entanto, sofreu grandes mudanças. Embora os nomes dos métodos antigos tenham se mantido, eles foram totalmente reformulados. A refatoração visou adequar o código às novas regras de negócio e melhorar sua organização.

Além das mudanças nas classes e *Enums*, houve alterações na interface de usuário. Foi adicionado um botão que permita ao usuário importar um arquivo texto; o *input* de arquivo para *download* foi transferido de local, tornando mais clara sua intenção; o *mini player* foi trocado por um botão “Ouvir música”.