**CRIANDO A PÁGINA INICIAL**

Esse é o template da Home.vue, que é a página inicial:

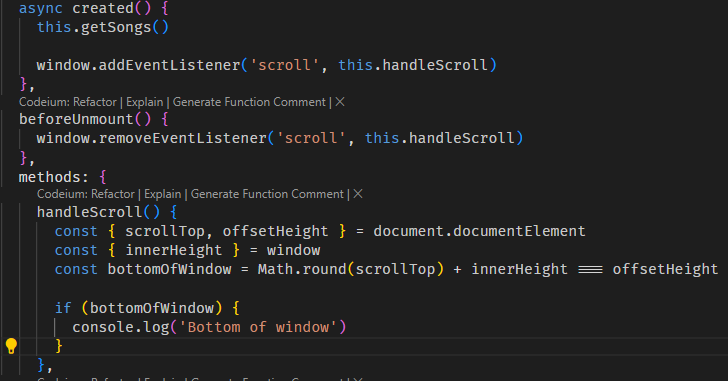
****

Para popular esse template, a lógica está descrita abaixo:

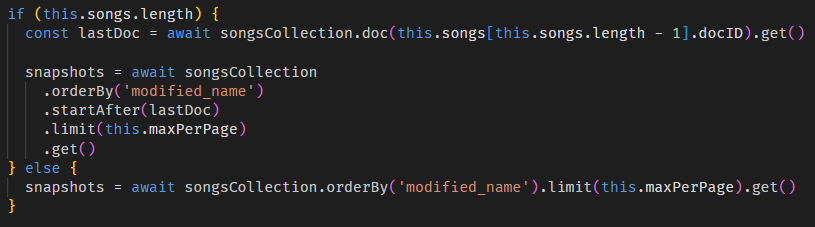


Perceba que é simples: é feito uma consulta no firebase de todas as collections de songs e armazenado dentro de snapshots. Essa songsCollection, está definida em includes/firebase.js, e está lá descrita como db.collections(‘songs’), então essa variável snapshots contém todos os documents da collection. O forEach vai popular o array songs definindo uma chave docID e pegando essa chave na document, e o resto irá ser puxado tudo como está.

**VERIFICANDO A POSIÇÃO DO SCROLL**

****

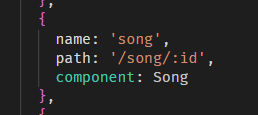
**SCROLL INFINITO**

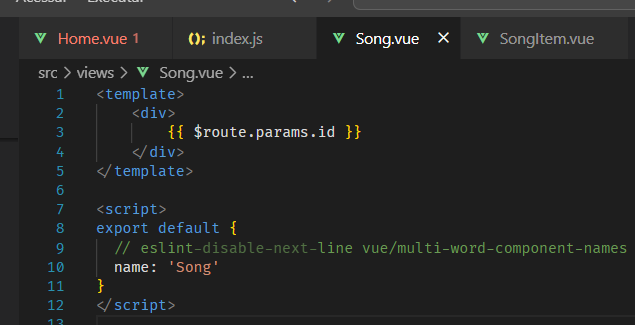
****

O maxPerPage vai permitir estabelecer uma quantidade máxima de resultados na página.

**PARÂMETROS PATH**

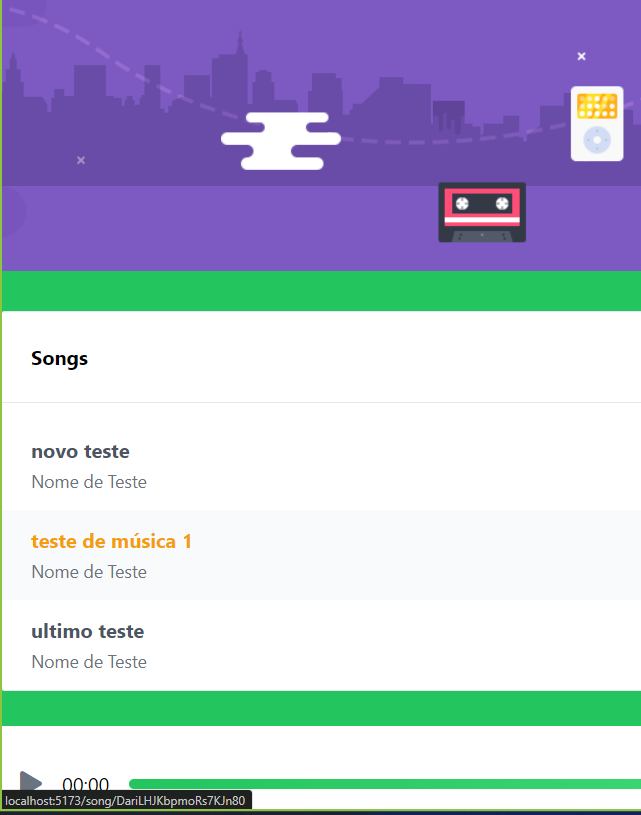
Criada a rota das músicas:



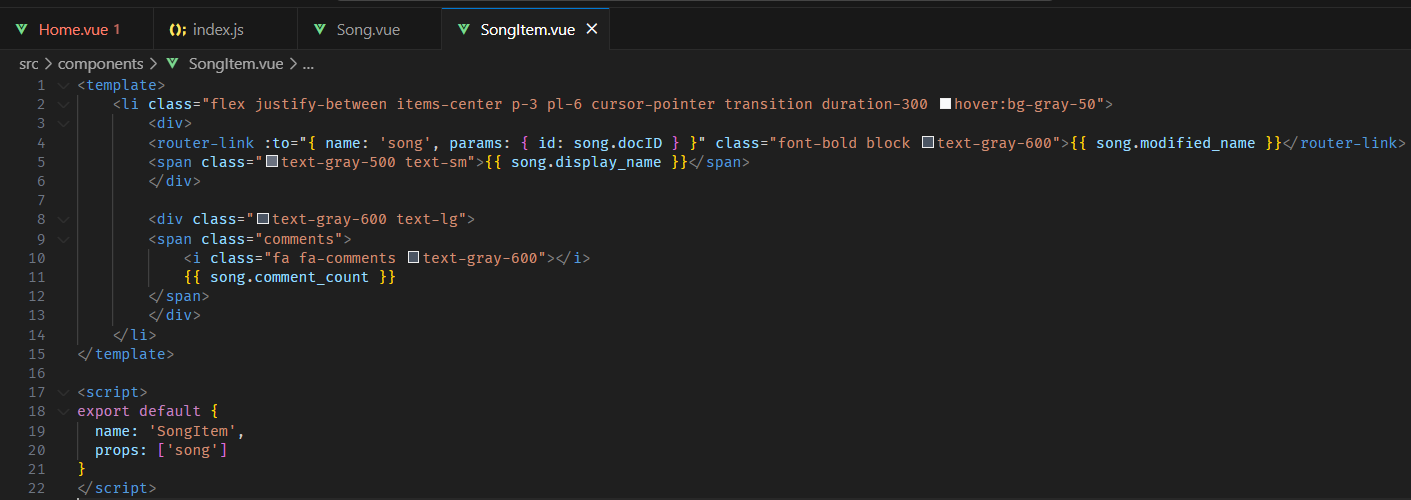


**CRIANDO UM TEMPLATE DE MÚSICA**

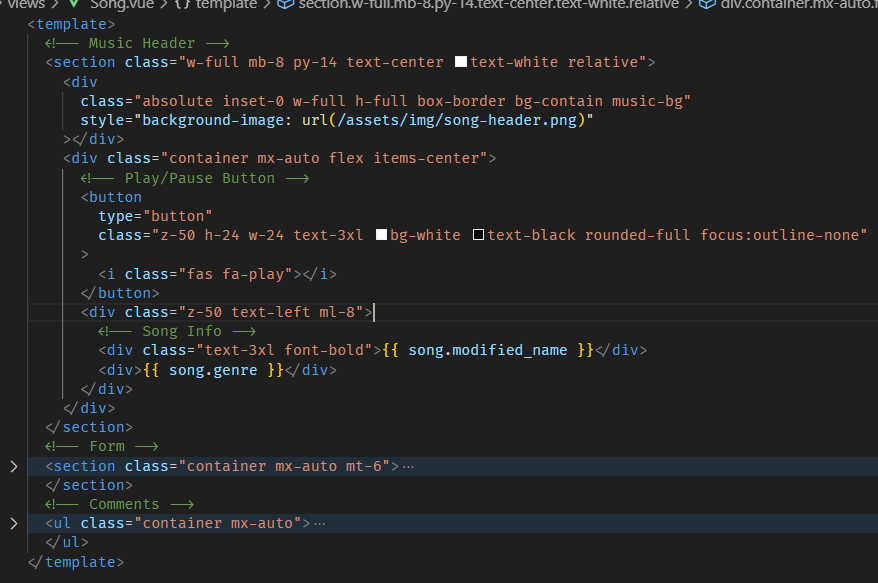
Uma necessidade urgente é fazer com que as músicas da home, ao serem clicadas, redirecionem para sua página individual da música:



Para isso, basta fazer a seguinte modificação no componente de SongItem.vue:



O que ocorre é que foi transformada a tag a em router-link, com o bind :to colocando o name da rota e passando o parâmetro dinâmico para música. A página de song.vue é para a qual é roteada ao clicar no nome da música:

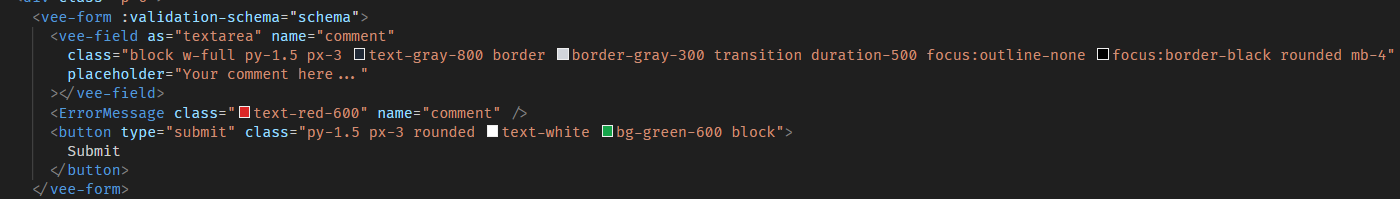




**VALIDANDO COMENTÁRIOS**

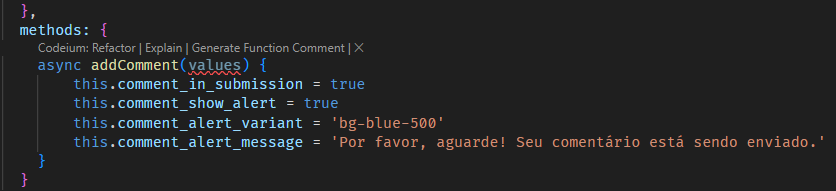
Passos para validar o comentário:

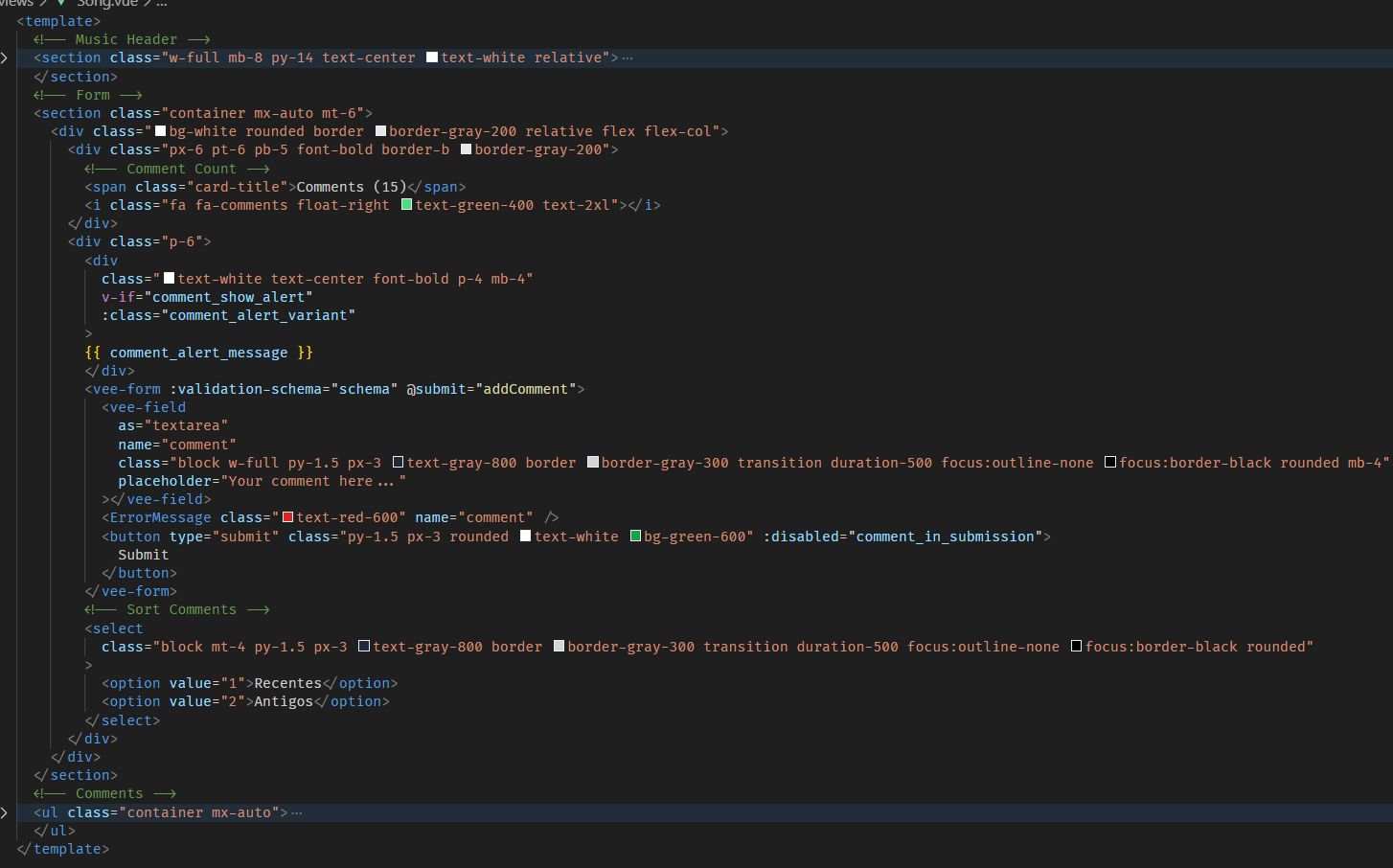
1. Validar o comentário
   1. Required
   2. Mínimo de 3 caracteres
2. Enviar comentário
3. Limpar input
4. Obter comentários
5. Obter último comentário em submissão
6. Classificar Comentários





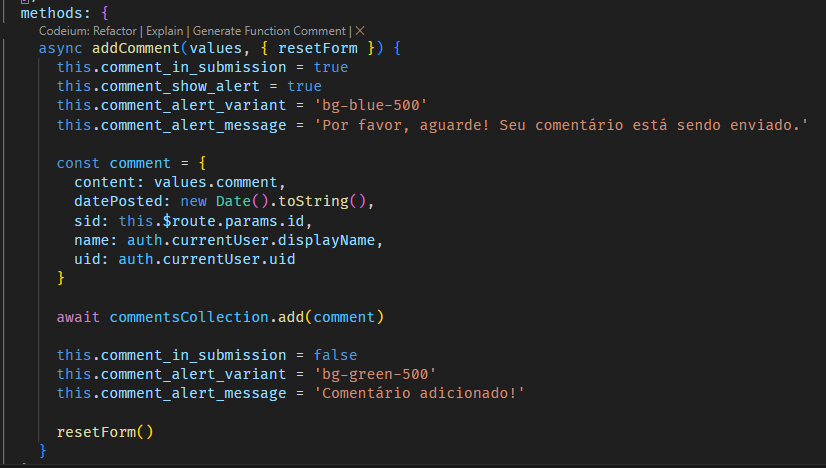
**PREPARANDO O FORMULÁRIO**

****

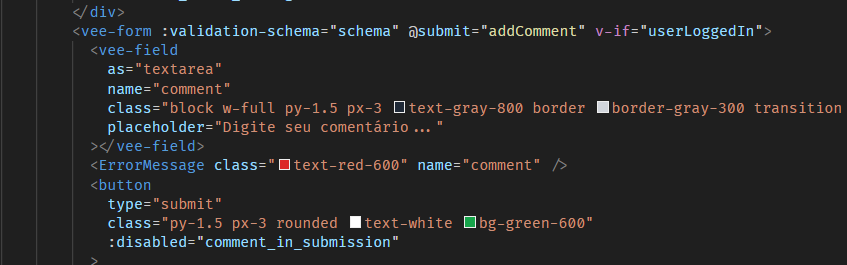
****

**FINALIZANDO O FORMULÁRIO DE COMENTÁRIO**

Desenvolvido o alerta para os comentários já adicionados, ficando na cor verde, alterada a mensagem e também limpado o input.

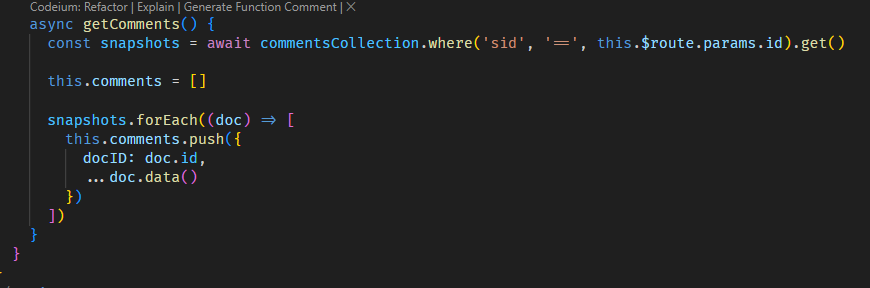


Por fim, para poder inserir comentários, é necessário estar logado. Para isso, adicionamos o auth e também colocamos uma diretiva v-if para usuário logado:



**EXIBINDO OS COMENTÁRIOS**

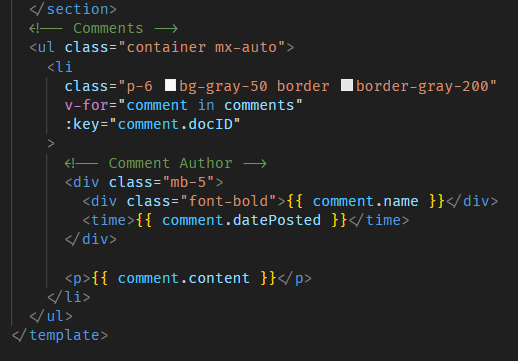
Há duas situações em que se deseja ver os comentários: ao carregar o componente e após fazer um comentário**.** A função async a seguir irá popular o array de comments:

****

Tenha atenção, pois essa função de getComments é chamada no created:



Ali é exibido o array de comments:

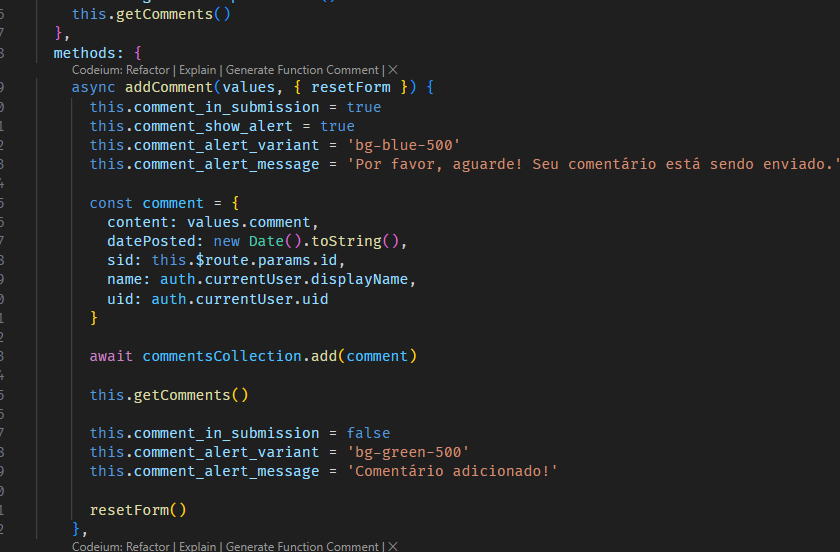


**ATUALIZANDO A LISTA DE COMENTÁRIOS**

Importante fazer uma distinção entre slice e sort:

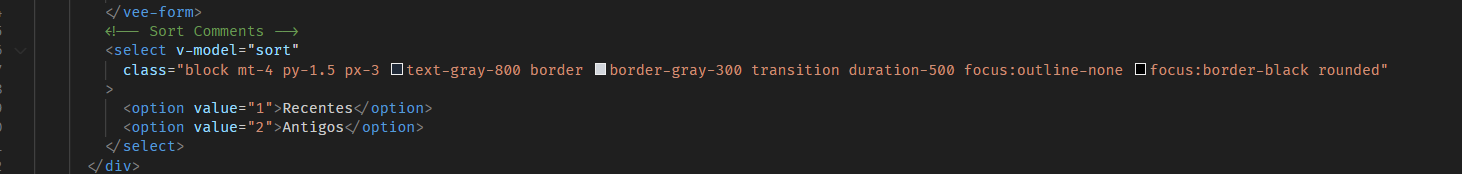
O método slice vai retornar um novo array. enquanto que o sort irá agir sobre o array original. Além disso, o sort permite passar uma função para decidir como será a classificação desse array, em que o valor 1 será do mais recente para o mais antigo e o 2 será do mais antigo para o mais recente.

Atualizando os comentários logo em seguida ao envio do comentário:

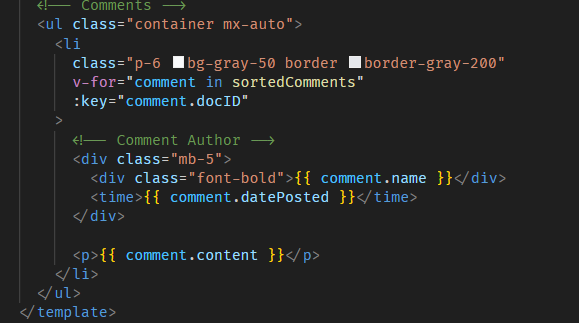


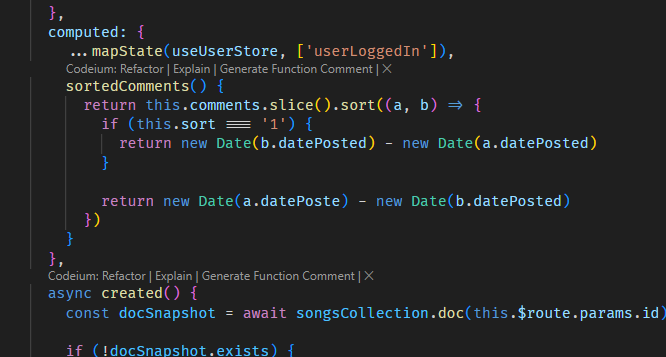
Mecanismo de filtragem dos mais recentes para os mais antigos, e vice-versa:

Cada uma dessas opções está atrelada a um valor:



Esse valor irá ordenar os comentários:





**PARÂMETROS DE CONSULTA**

[**https://example.com/this/is/a/path?foo=bar&order=1**](https://example.com/this/is/a/path?foo=bar&order=1)

Parâmetros de consulta são tudo o que se encontra após o path, partindo do ponto de interrogação.

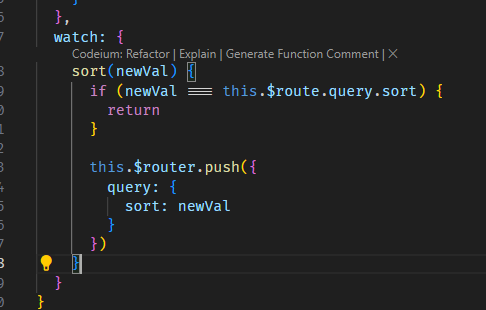
**DETECTANDO PARÂMETROS DE CONSULTA**

Query Parameters - Devem ser utilizados para filtros de dados.

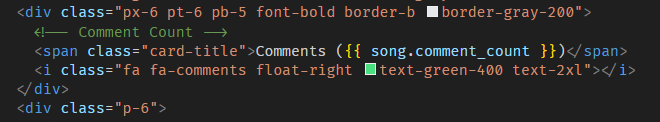
Ex: /song?id=1

Path Parameters - Devem ser utilizados para retornar simples ou múltiplos recursos.

Ex: /song/1



**ATUALIZANDO A QUANTIDADE DE COMENTÁRIOS**

****

****

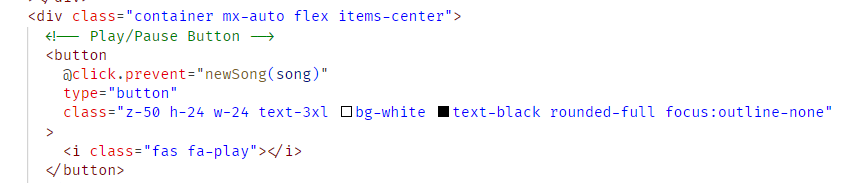
**ARMAZENANDO A MÚSICA EM ESTADO**

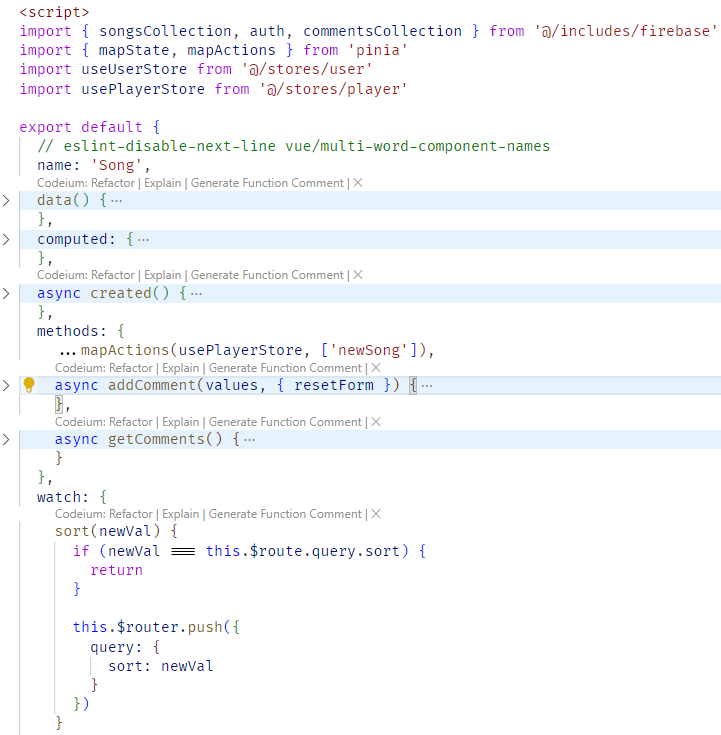
O processo:

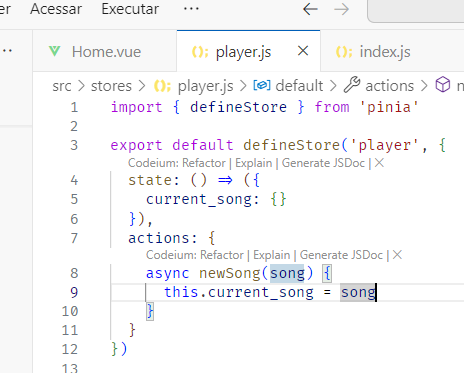
1. Armazenar a música em estado quando o botão do play é clicado
2. Reproduzir música
3. Alternar reproduzir/pausar
4. Exibir informações da música no player
   1. Duração / Tempo atual
   2. Nome da música
   3. Artista
5. Manter rastreado o progresso atual da música
6. Permitir o usuário arrastar o tempo para o momento desejado na música
7. Pausar a música após concluir a reprodução

Primeiro, precisamos de uma biblioteca de áudio que contenha já o reprodutor e os componentes de som necessários: npm install howler

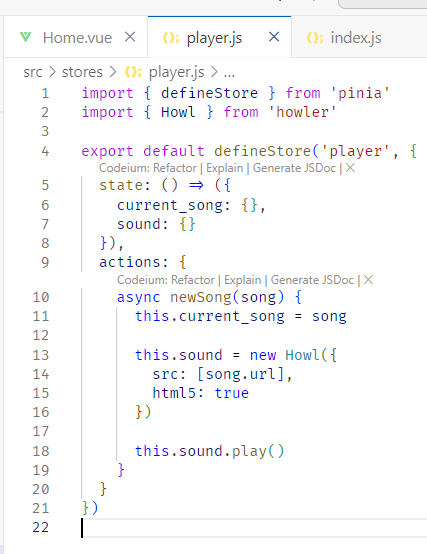
Em seguida, adicionamos um evento listener no botão de play:





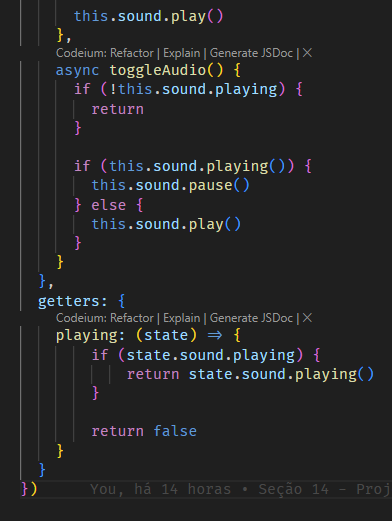


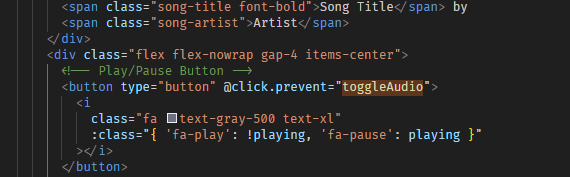
**REPRODUZINDO ÁUDIO**

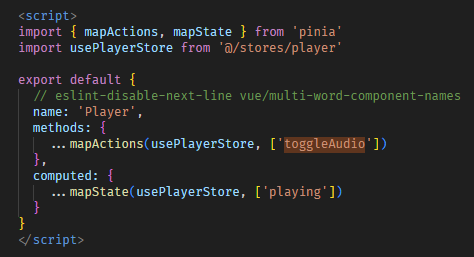


Instanciei o Howler e adicionei as propriedades de src e html5, em seguida acessei a propriedade play(). Isso já faz com que a música seja reproduzida!

**ALTERNANDO ÁUDIO**

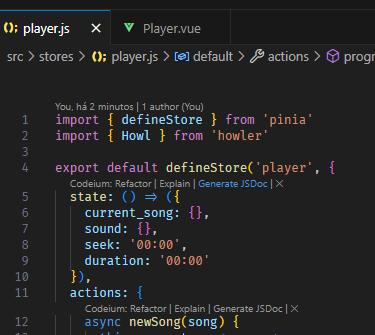
****

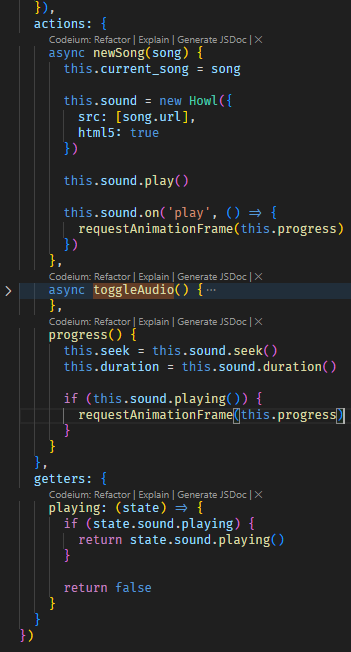




**DURAÇÃO E POSIÇÃO ATUAL**

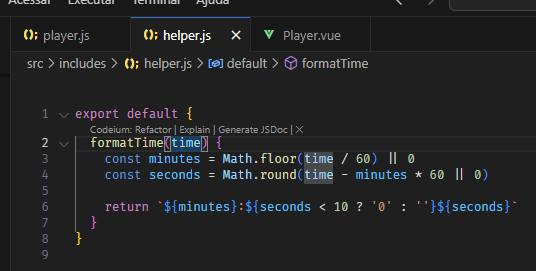
Criei duas propriedades novas no state, uma referente ao tempo atual de execução e outra ao tempo total. Colocamos essas funções dentro do map de playing, e substituímos elas nos templates.



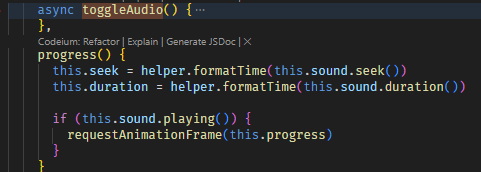


**FORMATANDO O TEMPO**

Criado um novo arquivo para colocar funções de utilidade, como formatar o tempo, por exemplo.

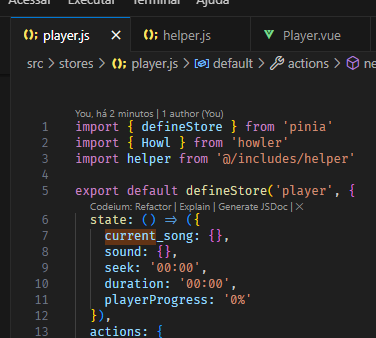


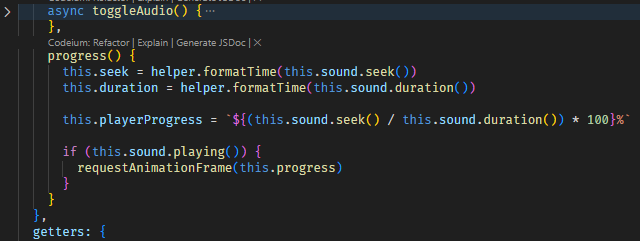
Essa função retorna o tempo no formato desejado. Para aplicar essa formatação, basta aplicar ela:



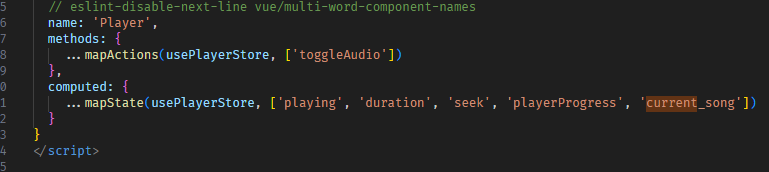
**TORNANDO FUNCIONAL A BARRA DE PROGRESSO**

Criada uma nova propriedade para a barra de progresso:





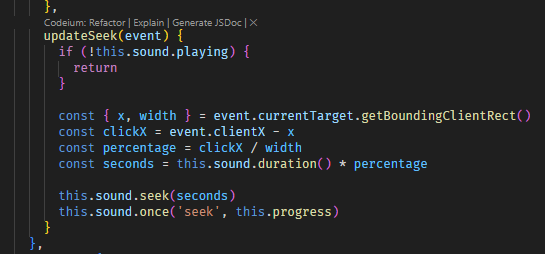
A propriedade playerProgress recebe os dados de seek e divide por duration e deixa pronto para exibição.

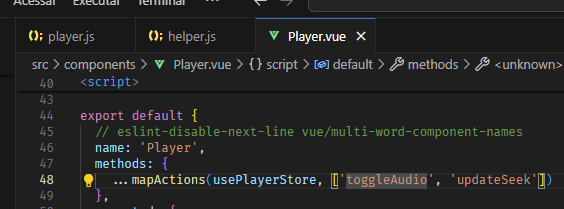


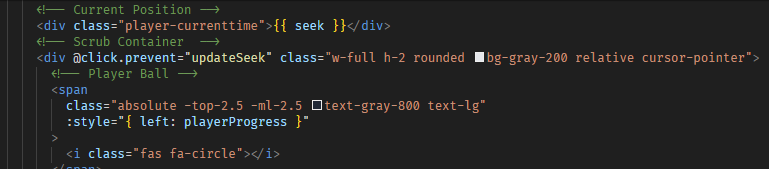
Importamos essa propriedade na computed.

**ESCOLHENDO A POSIÇÃO DO ÁUDIO**

Criada uma função para atualizar o tempo e também a reprodução da música, updateSeek:







**CRIANDO LINKS COM FRAGMENTOS HASH**

