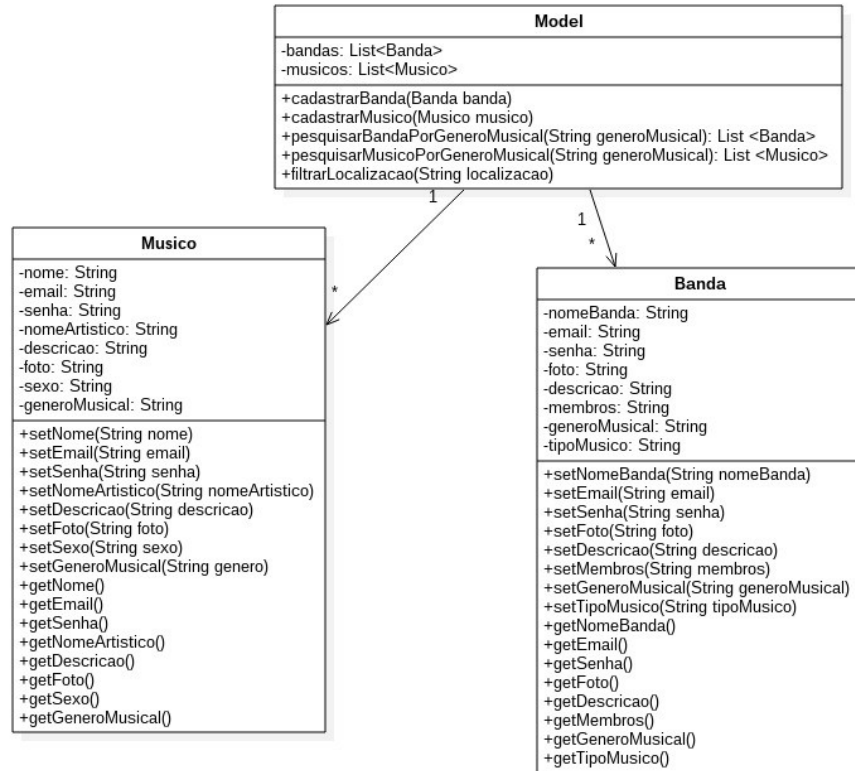


Atividades	Match Band
Requisitos	<p>Requisitos Funcionais:</p> <ul style="list-style-type: none"> - Usuario se Cadastra como Músico com Nome, e-mail, senha, nome artistico, foto, descrição, sexo, genero musical, contato - Usuario Cadastra Banda com Nome da Banda, e-mail, senha, foto, descrição da banda, membros, genero musical, Tipo do musico que esta procurando - Banda Pesquisa Músico com Filtro por genero musical - Musico Pesquisa Banda com Filtro por genero musical - Banda Conversa com Músico por chat - Musico Conversar com Banda por chat - Musico edita usuário com nome, senha, e-mail, foto, descrição, sexo, genero musical, contato, nome artistico - Banda Edita usuário com edição de nome, senha, e-mail, foto, descrição, sexo, genero musical, contato - Deletar Senha - Deletar cadastro <p>Requisitos Não Funcionais:</p> <ul style="list-style-type: none"> - Separação de Interesses (Separation of Concerns - SoC): separar corretamente lógica, interação com o usuário e comportamentos na aplicação. - Utilizar a Arquitetura de Software MVC (Model-View-Controller) - Portabilidade: desenvolver para diferentes plataformas. A Arquitetura MVC, através de sua separação de interesse, permitirá a criação de diferentes Views para o mesmo Model. <ul style="list-style-type: none"> - Plataforma Web <ul style="list-style-type: none"> - Desktop - Notebook - Tablet - Celular - Usabilidade: oferecer uma experiência de uso simples para as pessoas considerando os padrões de interação das plataformas que elas estão usando. - Usabilidade para plataforma Web http://www.w3.org/WAI/WCAG20/quickref/ - Desempenho: melhorar a performance da aplicação, buscando uma melhor plataforma para o usuário.

	<p>- Segurança: garantir que apenas os usuários habilitados acessarão a aplicação e que os dados serão transmitidos de forma segura.</p>
Projeto	<p>Match between system and the real world: O sistema deve falar a língua do usuário, com palavras, frases e imagens familiares, fazendo com que a informação apareça de forma natural e lógica. Isso vai acertar em cheio com nosso projeto, pois temos que ser bem claros em palavras e imagens para que o usuário possa utilizar de forma agradável e sem complicação.</p> <p>Error prevention: Utilizar de forma correta mensagens de erro, cores e de forma suave, mostrar erros que o usuário pode ter cometido e/ou o sistema.</p> <p>Flexibility and Efficiency of use: Eficiência na velocidade de uso e flexibilidade no tratamento de diferentes usuários.</p> <p>Aesthetic and Minimalist design: Informações simples e de rápido entendimento, com design leve para o usuário não ficar com a vista pesada.</p>



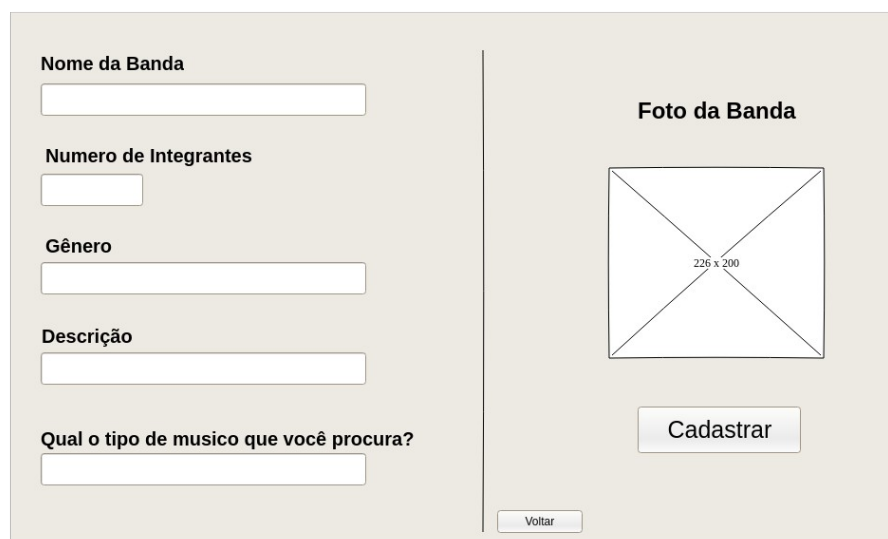
- Projeto de Interação com o Usuário

8 -Estética e Design Minimalista

The image shows two forms for user interaction. The left form is for login, with fields for 'E-Mail' and 'Senha' (Password), and an 'Entrar' (Enter) button. The right form is for registration, with fields for 'E-mail', 'Nome' (Name), 'Senha', and 'Confirmar Senha' (Confirm Password), a checkbox for 'Eu aceito os Termos de uso regidos pelo Match Band' (I accept the terms of use governed by Match Band), and a 'Cadastrar' (Register) button.

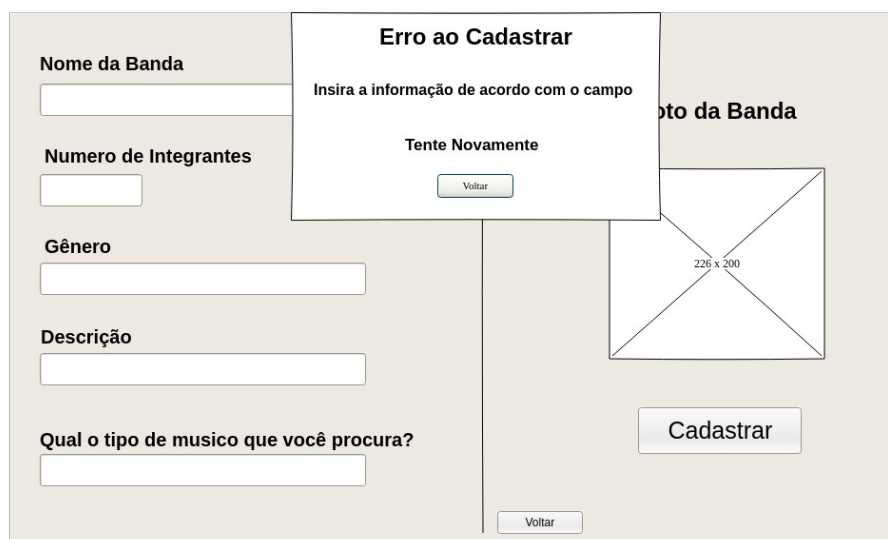


2. Conexão entre o sistema e o mundo real

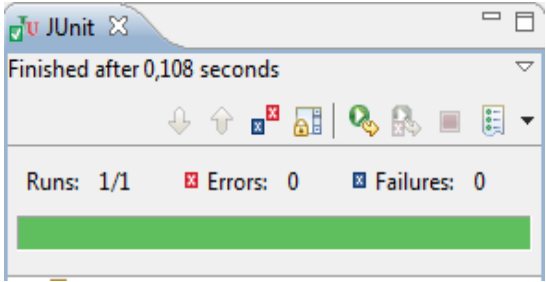


A registration form for a band. It is divided into two columns. The left column contains five input fields: 'Nome da Banda', 'Numero de Integrantes', 'Gênero', 'Descrição', and 'Qual o tipo de musico que você procura?'. The right column features a placeholder for a band photo, labeled 'Foto da Banda', with a 226 x 200 pixel dimension. Below the photo placeholder is a 'Cadastrar' button. At the bottom right of the form is a 'Voltar' button.

3 - Prevenção de Erros



The same registration form as above, but with an error message displayed in a modal window. The modal window has the title 'Erro ao Cadastrar' and contains the text 'Insira a informação de acordo com o campo' and 'Tente Novamente'. There is a 'Voltar' button inside the modal. The background form is partially obscured by the modal.

Implementação	https://github.com/giulianobertoti/eXtremeProgramming
Testes	<p>https://github.com/giulianobertoti/eXtremeProgramming/tree/master/Tablatures/src</p>  <pre>@Test public void test() { MusicList list = new MusicList(); list.addMusic(new Music("Note:Capo in the 3rd -- [G] How many times do you come back?")) list.addMusic(new Music("Note:Capo in the 2nd -- [Em] Today I'm a soldier")) assertEquals(list.getMusics().size(), 2); //testing ADD Music music = list.searchMusic(new Specification("Oasis", "Wonderwall")) assertEquals(music.getChords(), "Note:Capo in the 2nd -- [Em] Today I'm a soldier") }</pre> <p>1º Teste: Adicionei 2 objetos Music no atributo musics da</p>

	<p>classe MusicList e depois testei se de fato haviam 2.</p> <p>2º Teste: Fiz uma busca passando um objeto Specification e conferi se o resultado era correto.</p>
Manutenção	<p>1ª Testes Automatizados: em toda atualização, principalmente naquelas onde não fui eu que implementei a lógica (foi outro membro da equipe), eu tenho testes automatizados para checar se eu quebrei ou não o código.</p> <p>2ª Encapsulamento: para acrescentar um novo requisito funcional do cliente, por exemplo comparação por ano da música, eu precisei alterar apenas a classe Specification, pois todas as comparações da busca por músicas estão lá (ou seja, meu projeto está bem encapsulado). A classe MusicList, que é a classe principal de lógica, não precisou ser alterada.</p> <pre> public class Specification { private String band; private String musicName; private int year; public Specification(String band, String musicName, int year){ this.band = band; this.musicName = musicName; this.year = year; } public boolean matches(Specification spec){ if(!band.equals(spec.band)) return false; if(!musicName.equals(spec.musicName)) return false; if(year!=spec.year) return false; return true; } } </pre>
	FIM