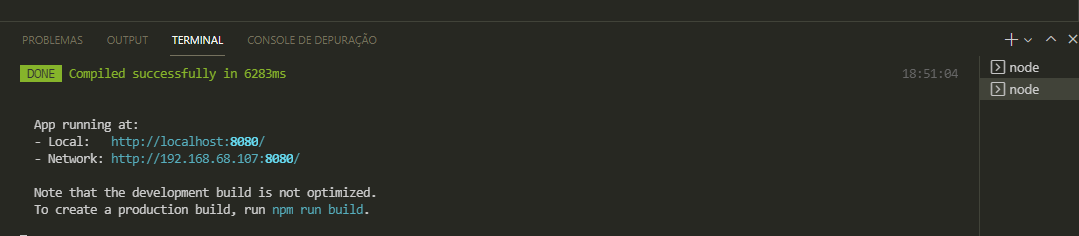
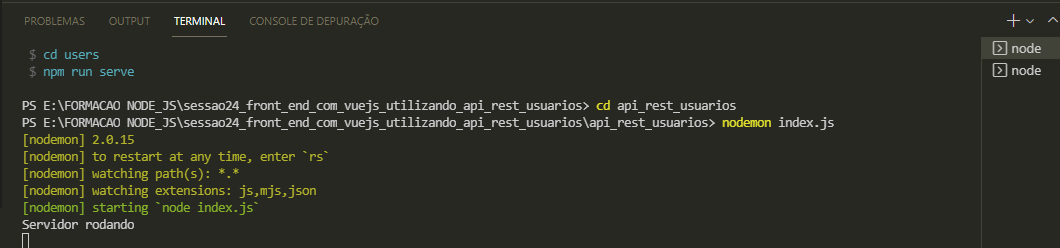
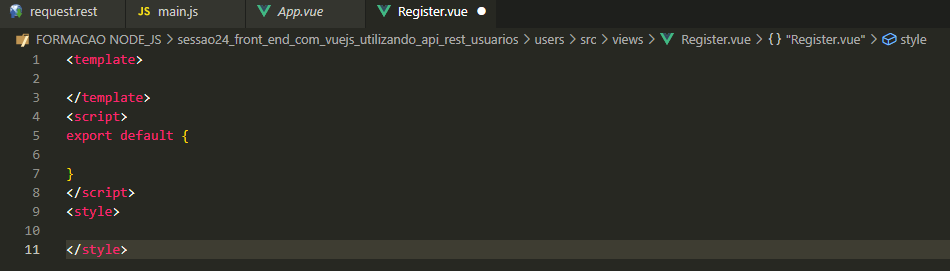
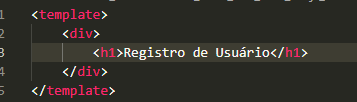
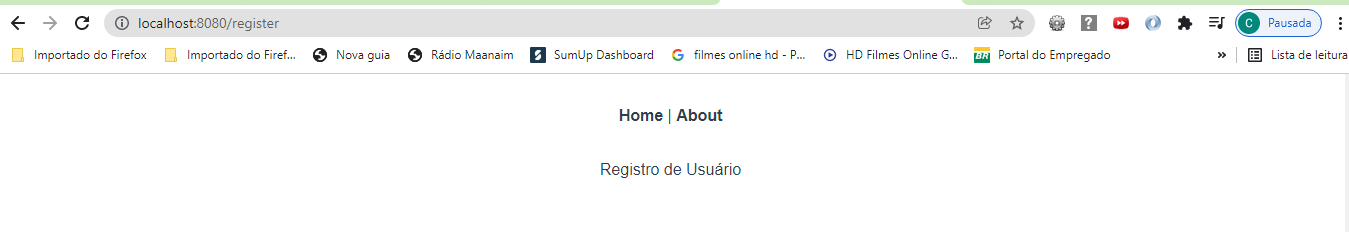
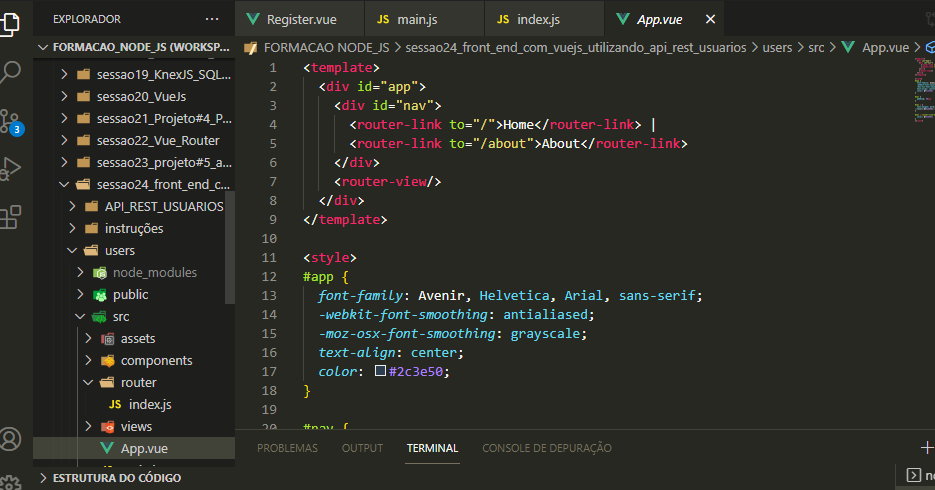
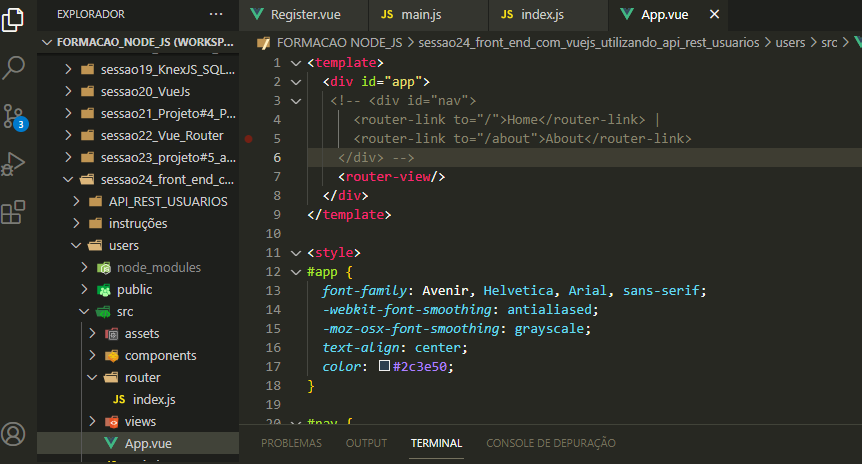
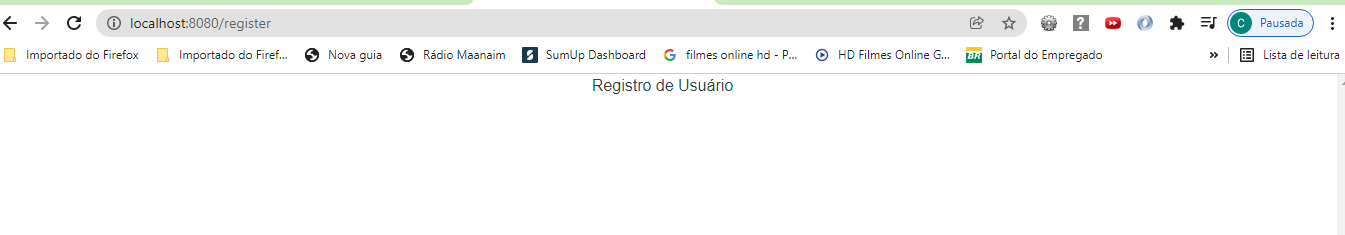
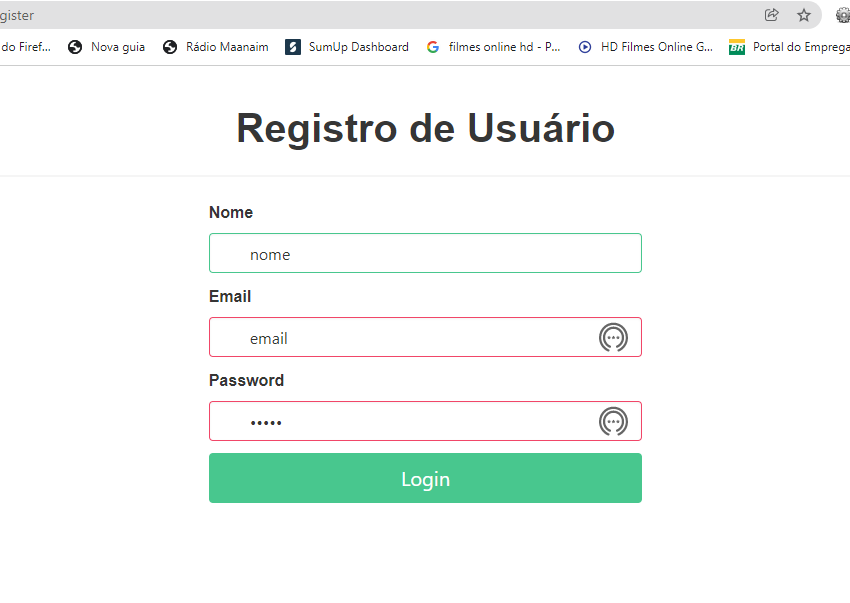
**SISTEMA DE GESTÃO DE USUÁRIOS COM FRONTEND VUEJS**

1. Copie a pasta API\_RES\_USUARIOS do projeto anterior para dentro da pasta deste novo projeto
2. Vamos primeiramente criar um projeto Vue com o router chamado usuer digite:
3. Vue create users.
4. Selecione a opção ‘Manually select features’ para selecionar o que será adicionado à aplicação vue,
5. Navegue até Router e digite a barra de espaço para seleciona-lo.
6. Tecle enter
7. Tecle enter
8. A próxima tela pergunta se vc quer instalar o history mode. O history mode cria rotas mais amigáveis, porém quando for fazer o deploy será necessário alterar configurações do apache dependendo do servidor.
9. Mas vamos utilizar o history mode assim mesmo.
10. Digite Y
11. Pressione Enter na primeira opção,
12. Pressione Enter na primeira opção
13. Pressione Enter na primeira opção
14. Digite N e tecle Enter
15. Mantenha o servidor rodando na pasta da api.
16. Abra outro terminal e instale o Bulma como framework css na pasta users.
17. Digite npm install bulma
18. Abra o arquivo main.js na pasta src e importe o bulma
19. 
20. Na pasta users instale também o axios para podermos realizar as consultas na nossa api.
21. Npm install axios
22. Vamos agorar rodar o projeto vue dentro da pasta user, digite npm run serve.
23. teremos a API rodando dentro de um terminal e o projeto vue rodando em outro.
24. Terminal com o servidor do vue rodando
25. 
26. Terminal com a API rodando
27. 

**Criando view Register**

1. Na pasta views crie um arquivo chamado Register.vue que será a página de cadastro.
2. Vamos começando definindo a configuração básica de uma página vue
3. 
4. Dentro do template crie uma div
5. Cire um h1 com a mensagem Registro de usuário
6. 
7. Vamos abrir o arquivo index.js na pasta router
8. Vamos importar o componente Register.vue e criar uma nova rota. Com o path ‘/register’ o name ‘Register’ e componente Register.
9. Agora para testar basta ir na página e digitar a url register.
10. 
11. Como podemos ver está aparencendo o link Home e About na página,
12. Abra o arquivo App.vue e podemos ver que eles estão fora do <router-view> eles estão dentro de um <router-link> por isso estão aparencendo como um menu.
13. 
14. Vamos comentar então os <router-link>.
15. 
16. A página ficará da seguinte forma.
17. 
18. Vamos desenvolver um pequeno formulário no componente Register.vue
19. Vamos na página do bulma para para copiar os códigos dos inputs que vamos utilizar.
20. <template>
21. <div class="content is-medium">
22. <h1 style="padding-top:40px">Registro de Usuário</h1>
23. <hr>
24. <div class="columns is-centered">
25. <div class="column is-one-third">
26. <div class="field">
27. <label class="label" style="text-align:left">Nome</label>
28. <div class="control has-icons-left has-icons-right">
29. <input class="input is-success" type="text" name="name" placeholder="Nome do usuário" value="">
30. <span class="icon is-small is-left">
31. <i class="fas fa-user"></i>
32. </span>
33. <span class="icon is-small is-right">
34. <i class="fas fa-check"></i>
35. </span>
36. </div>
37. </div>
38. <div class="field">
39. <label class="label" style="text-align:left">Email</label>
40. <div class="control has-icons-left has-icons-right">
41. <input class="input is-danger" type="email" name="email" placeholder="Email do usuário" value="">
42. <span class="icon is-small is-left">
43. <i class="fas fa-envelope"></i>
44. </span>
45. <span class="icon is-small is-right">
46. <i class="fas fa-exclamation-triangle"></i>
47. </span>
48. </div>
49. </div>
50. <div class="field">
51. <label class="label" style="text-align:left">Password</label>
52. <div class="control has-icons-left has-icons-right">
53. <input class="input is-danger" type="password" name="password" placeholder="\*\*\*\*\*\*" value="">
54. <span class="icon is-small is-left">
55. <i class="fas fa-lock"></i>
56. </span>
57. <span class="icon is-small is-right">
58. <i class="fas fa-exclamation-triangle"></i>
59. </span>
60. </div>
61. </div>
62. <div class="field">
63. <p class="control">
64. <button class="button is-success is-medium is-fullwidth">
65. Login
66. </button>
67. </p>
68. </div>
69. </div>
70. </div>
71. </div>
72. </template>
73. Vamos criar um data() que irá retornar um json com os campos do formulário.
74. <script>
75. export default {
76. data(){
77. return{
78. name:'',
79. password:'',
80. email: ''
81. }
82. }
83. }
84. Agora vamos fazer um databynd nos campos do formulário com o v-model e nome do campo.
85. <input class="input is-success" type="text" placeholder="Nome do usuário" v-model="name">
86. <input class="input is-danger" type="email" placeholder="Email do usuário" v-model="email">
87. <input class="input is-danger" type="password" placeholder="\*\*\*\*\*\*" v-model="password">
88. Vamos criar um método chamado register() dentro do json de methods, que no inicio irá fazer um console.log das variáveis
89. methods:{
90. register(){
91. console.log(this.name);
92. console.log(this.email);
93. console.log(this.password);
94. }
95. }
96. Agora vamos chamar o envento @click no botão passando o método register().
97. <button class="button is-success is-medium is-fullwidth" @click="register">
98. Login
99. </button>
100. A página ficará assim:
101. 

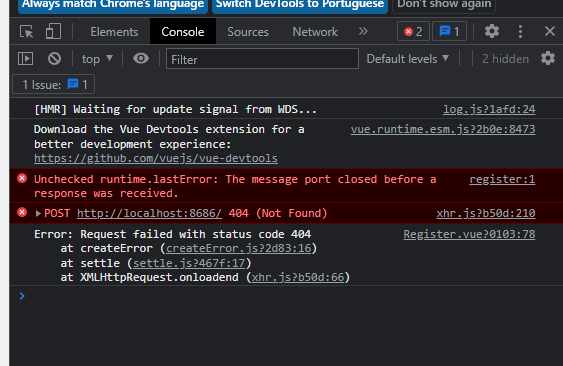
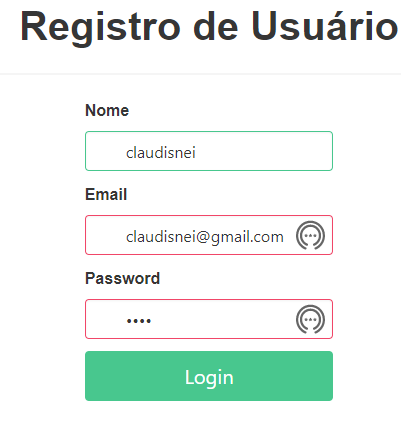
**Requisição da formulário de Registro para a API**

1. Faremos a requisição utilizando o axios.
2. Vamos importar o axios dentro do script do componente Register.vue
3. <script>
4. import axios from 'axios';
5. no método register vamos utilizar o axios que fará uma requisição do tipo get para uma rota da API. <http://localhost:8686/> e nos dará uma resposta ou um erro.
6. methods:{
7. register(){
8. axios.get('http://localhost:8686/').then((*res*)*=>*{
9. console.log(*res*);
10. }).catch(*err=>*{
11. console.log(*err*);
12. });
13. }
14. }
15. Ocorreu um erro devido ao cors, pois é uma política de segurança para troca de informações entre servidores. O projeto vue está rodando em uma porta diferente da API.

**Instalando a biblioteca cors na API**

1. Para resolver este erro temos que ir até a pasta onde está rodando a API, parar o servidor e instalar a biblioteca chamada cors. – npm install cors -- save
2. Essa biblioteca funciona tipo um middleware do express
3. Vamos importar essa biblioteca para o arquivo index.js do projeto API\_REST\_USERS
4. *var* cors = require('cors');
5. vamos solicitar que o app utilize o cors.
6. app.use(cors());
7. rode novamente o servidor e faça o teste na view register.

**Enviando dados para o servidor**

1. Vamos alterar a rota no método register para enviar os dados dos inputs.
2. methods:{
3. register(){
4. axios.post('http://localhost:8686/user',{
5. name: this.name,
6. password: this.password,
7. email: this.email;
8. }).then((*res*)*=>*{
9. console.log(*res*);
10. }).catch(*err=>*{
11. console.log(*err*);
12. });
13. }
14. }
15. Ao clicarmos no botão salvar aparece um erro que não informa nada
16. 
17. Vamos alterar a mensagem de erro para melhorar a resposta ao usuário.
18. }).catch(*err=>*{
19. *var* errMens = *err*.response.data.err;
20. console.log(errMens);
21. });
22. Desta forma o erro será mais amigável.
23. 
24. Agora vamos criar um usuário para testar
25. 
26. O retorno no console foi.
27. 