



**FACULDADE DE INFORMÁTICA DE PRESIDENTE
PRUDENTE**

SISTEMAS DE INFORMAÇÃO

**Andressa Felisberto Ferreira Diniz
Caio Lomas de Oliveira
Leonardo Ancieto das Neves
Livia Gomes de Souza**

Site - 35º INFOESTE

Presidente Prudente - SP

2023

Sumário**Primeiro bimestre**

- Observações 4
- Seção inicial 4 a 7
- Sobre o evento 7 a 8
- Programação do evento 9 a 11
- Inscrição12 a 14
- Contato e portfólio14 a 17

Segundo bimestre

- Pastas e organização.....18
- Classes.....18 a 20
- Formulário.....20 a 22
- Chamando o HTML.....22 a 24
- O projeto.....25

35º INFOESTE

De 23 a 27 de outubro

- > 35º Ciclo de Cursos e Palestras
- > 17º Festa Linux
- > 17º Maratona de Programação
- > 9º FIPPETEC - Ciclo de Cursos e Palestras para ETECs

Observações

- Leia somente a partir da página onde começa o segundo bimestre, o documento é a junção do primeiro com o segundo bimestre, portanto este começo já foi visto anteriormente.
- O site foi feito utilizando HTML e CSS com modelo de layout FlexBox, além da linguagem Python com banco de dados e o framework cherrypy.
- Há muitas linhas de código, principalmente no CSS então trarei filtrado partes mais importantes e interessantes do HTML e da estilização.
- O tema (identidade visual) escolhido para o site foi “Neon”
- Todos os arquivos utilizados para fazer o site estão na pasta “35º Infoeste” e também o documento.
- No site há 4 text files, “index.html” – que é o html onde contém toda estrutura do site. “general.css” – é a estilização padrão do site. “ queries.css “ – contém as animações e classes que auxiliam na responsividade do site para telas reduzidas e mobile. “ style.css “ – É o css principal, com todas as classes, fontes, imagens, luzes etc. do site.

Sessão inicial

No cabeçalho da página, temos algumas imagens e textos com links levando a outros sites, ou mais abaixo na mesma.

Logo Unoeste e FIPP



Ambos estão ancorados com href e seus links.

Menu de itens da página



A direita estão os itens do menu, eles estão ancorados com as sections abaixo no site.

Para ser fiel ao tema neon, foram adicionados caixas com cores e brilho de fundo ao passar o mouse por cima.

É importante salientar que houve um cuidado especial com a responsividade desta parte, pois sem ela, ao minimizar a página a barra onde ficam os itens se desmancharia, mas isso foi corrigido com a classe "sidebar".



HTML

```
<header>
  <div class="hero maxWidthHeight center">
    <div class="sidebar">
      <div class="logo-uno-box center">
        <a href="https://www.unoeste.br" target="_blank">
      <a href="page.html" class="menu-item">Sobre</a>
      <a href="#inscricao" class="menu-item">Inscrição</a>
      <a href="#programacao" class="menu-item">Programação</a>
      <a href="" class="menu-item">Edições anteriores</a>
      <a href="#contato" class="menu-item">Contato</a>
    </div>
  </div>
```

CSS

```
.sidebar{
  grid-column: 1;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: start;
  width: 5rem;
  height: 100%;
  background-color: rgba(34, 34, 34, 0.5);
  color: #fff;
  overflow: hidden;
  border-radius: 0 1rem 1rem 0;
  transition: width .75s ease;
}
```



Finalizando a section “hero”, há o nome do evento em destaque, e os eventos que haverão na Infoeste em lista, com suas respectivas âncoras.

HTML

```

<div class="text">
  <div class="title">
    <p class="number">
      <h1 class="title">
    </div>
  <div class="data">
    <p class="data">
  </div>
  <div class="program">
    <ul class="list-program">
      <li class="list-program-item">
        <a href="#">
      </li>
      <li class="list-program-item">
      <li class="list-program-item">
      <li class="list-program-item">
    </ul>
  </div>
</div>

```

CSS

```

.title-hero:hover{
  color: #fff;
  text-shadow: 0 0 5px #83ff89,
    0 0 25px #83ff89,
    0 0 50px #83ff89,
    0 0 100px #83ff89;
  transition: 0.5s;
}

.data-hero-box{
  font-family: 'Barlow Condensed', sans-serif;
  justify-self: end;
  font-size: 2.4rem;
  margin-top: -3rem;
  margin-right: 1rem;
  color: rgba(150, 240, 147, 0.6);
}

.list-program {
  list-style: none;
}

.list-program li{
  padding: 0.8rem 0;
  font-family: 'Barlow Condensed', sans-serif;
  font-size: 2.2rem;
  margin-left: 2rem;
}

```

Sobre o evento

Para acessar a página sobre o evento deve-se clicar no item “Sobre” no cabeçalho da sessão inicial, isso deverá carregar uma nova página sob a mesma, um novo arquivo html “page”.

Evento



A Faculdade de Informática de Presidente Prudente (FIPP), da Universidade do Oeste Paulista (UNOESTE), realiza anualmente, a Semana de Computação e Informática da FIPP/Unoeste - INFOESTE.

A INFOESTE 2023, que em sua 35ª edição, espera um público aproximado de 600 participantes, é promovida e organizada pela FIPP com apoio institucional da Sociedade Brasileira de Computação (SBC), UNOESTE e Associação das Empresas de Software do Oeste Paulista POLOIN, e a colaboração de Docentes, Discentes, Funcionários, Empresa Júnior de Informática da Unoeste (UNINFO JR), Incubadora Tecnológica de Presidente Prudente (INTEPP), Diretório Acadêmico e Atlético da FIPP, Empresas Privadas/Públicas, IES Públicas/Privadas e Profissionais.

Aqui temos as informações gerais sobre a Infoeste, bem como todos seus eventos, palestras e cursos, porém sem detalhes aprofundados.

HTML

Esta section é bem simples, apenas textos e imagens.

```
</div>
<div class="sidetext">
<p>A Faculdade de Informática de Presidente Prudente (FIPP), da Universidade do Oeste Paulista (UNOESTE), realiza anualmente, a Semana de Computação e Informática da FIPP/Unoeste - INFOESTE.</p>

<p>A INFOESTE 2023, que em sua 35ª edição, espera um público aproximado de 600 participantes, é promovida e organizada pela FIPP com apoio institucional da Sociedade Brasileira de Computação (SBC), UNOESTE e Associação das Empresas de Software do Oeste Paulista (POLOIN), e a colaboração de Docentes, Discentes, Funcionários, Empresa Júnior de Informática da Unoeste (UNINFO JR), Incubadora Tecnológica de Presidente Prudente (INTEPP), Diretório Acadêmico e Atlético da FIPP, Empresas Privadas/Públicas, IES Públicas/Privadas e Profissionais.</p>

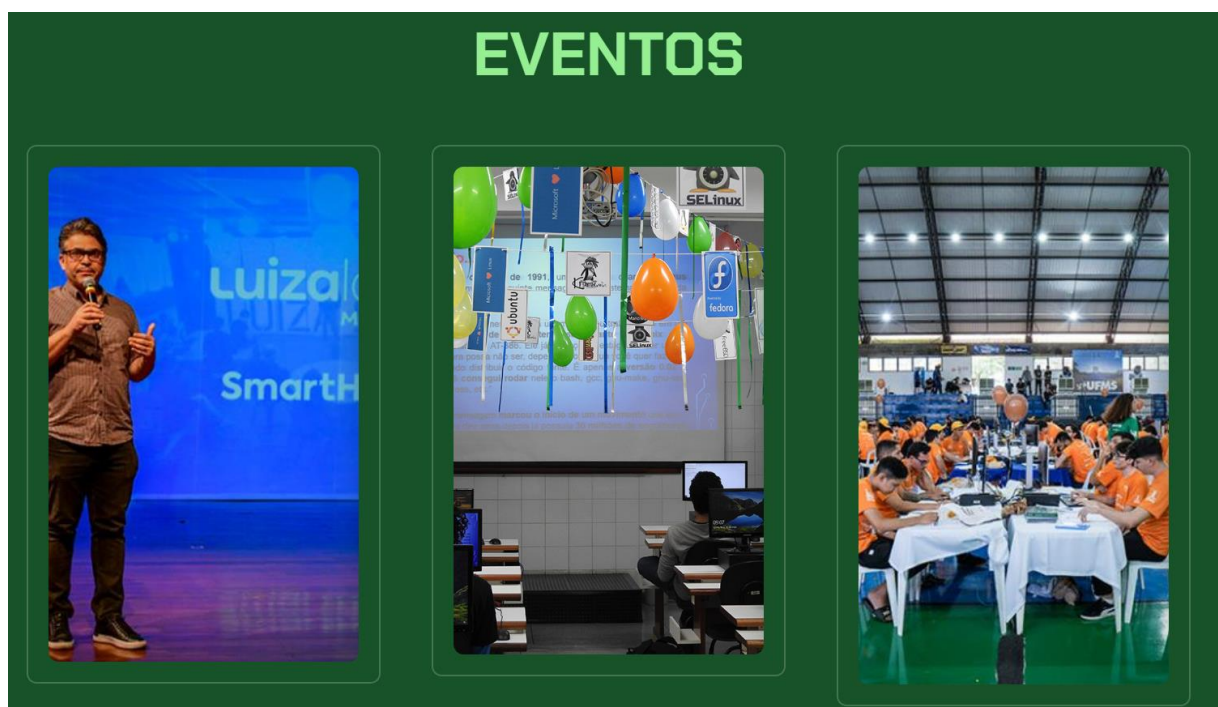
</div>
</div>
<div class="row-reversed">
<div class="sidetext">
A INFOESTE 2023 contará com os seguintes eventos integrados:

35ª Ciclo de Cursos e Palestras (Experience)
17ª Festa Linux;
17ª Maratona de Programação;
09ª FIPPETEC - Ciclo de Cursos e Palestras para as ETECs;
<p>O evento, que é aberto a toda a comunidade regional e de outros centros, procura congrega estudantes de graduação e pós-graduação, professores, pesquisadores, profissionais e demais interessados em informática, objetivando a difusão da computação e informática e do seu bom uso, trazendo para o debate aberto temas importantes que indicam as tendências tecnológicas e do mercado atual, além da discussão de práticas pedagógicas no ensino superior, buscando atingir com mais ênfase os alunos dos cursos da FIPP/Unoeste.</p>

</div>
</div>
```

Programação do evento

Essa section mostra com mais detalhes como serão os eventos que ocorrerão na Infoeste, com imagens ilustrando e seus respectivos textos explicativos.



O diferencial desta section é que a organização dela foi dividida em “cards”, onde cada um é uma descrição que representa o evento ao qual se coloca o mouse em cima.

EVENTOS





OUT
27

17ª Festa Linux

A Festa Linux tem como principais objetivos desmistificar a percepção de que o Linux é uma plataforma difícil de usar e complicada de instalar. Além disso, busca demonstrar a existência de alternativas de software livre de alta qualidade e confiabilidade, bem como oferecer apoio para aqueles que desejam instalar o Linux, mas não sabem por onde começar.

LEIA MAIS



Há muitas classes css presentes nos cards, como a estilização das suas cores, brilho, caixa de texto onde fica a data, caixa de texto com a explicação e até mesmo o “ Leia mais “.

HTML

```
</header>
<section class="programacao-box-cards" id="programacao">
  <div class="program-container flex-column center">
    <h1 class="title-box-program">Eventos</h1>
    <div class="cards-container">
      <div class="card">
        <div class="program-card card1">
          <div class="content-box-card">
            <div class="content-descr-card">
              <h1 class="card-title">35º Ciclo de Cursos e Palestras</h1>
              <p class="card-content">
                Durante o transcorrer do evento, uma ampla variedade de cursos e palestras são cuidadosamente organizados e disponibilizados aos participantes. Essa programação abrangente e diversificada é projetada com o intuito de enriquecer a experiência dos presentes, oferecendo oportunidades únicas de aprendizado e aquisição de conhecimento.
              </p>
              <div class="link-card"><a href="#" class="see-more">Leia mais</a></div>
            </div>
            <div class="date-box-card">
              <span class="month">Out</span>
              <span class="date">27</span>
            </div>
          </div>
        </div>
      </div>
      <div class="card">
        <div class="program-card card2">
          <div class="content-box-card">
            <div class="content-descr-card">
              <h1 class="card-title">17ª Festa Linux</h1>
              <p class="card-content">A Festa Linux tem como principais objetivos desmistificar a percepção de que o Linux é uma plataforma difícil de usar e complicada de instalar. Além disso, busca demonstrar a existência de alternativas de software livre de alta qualidade e confiabilidade, bem como oferecer apoio para aqueles que desejam instalar o Linux, mas não sabem por onde começar.
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

CSS

```
# style.css > ...
292
293 .card-title {
294   color: #000b03;
295   font-size: 2.5rem;
296   font-weight: 700;
297 }
298
299 .card-content {
300   padding-top: 1.5rem;
301   font-size: 1.8rem;
302   color: #000b03;
303 }
304
305 .link-card {
306   display: flex;
307 }
308
309 .see-more:link,
310 .see-more:visited {
311   display: block;
312   font-weight: 700;
313   font-size: 1.6rem;
314   letter-spacing: 0.1rem;
315   text-transform: uppercase;
316   color: #fff;
317   margin-top: 1.5rem;
318   text-decoration: none;
319
320   padding: 0.2rem 0.4rem;
321
322   background-color: #003710;
323 }
324
325 .see-more:hover,
326 .see-more:active {
327   box-shadow: 0 0 5px #83ff89,
328             0 0 25px #83ff89,
```

Inscrição

Na penúltima section está a realização da inscrição.

Com a reutilização de algumas classes css para containers e estilização das letras, o diferencial dessa parte é que foi feita como formulário <form> e listada.

FAÇA JÁ SUA INSCRIÇÃO!

- > Nome:
- > E-mail:
- > Telefone:
- > Data de nascimento:
- > Endereço completo:
- > Curso:
- > Endereço completo:
- Carregue sua foto
 - > Nenhum arquivo escolhido
- > Quer receber informações sobre a Infoeste?
 - ☐ Sim
 - ☐ Não

Cada item da inscrição tem seu tipo inserido a partir de um input e a frase a ser colocada é posta com um placeholder.

HTML

```

</li>
<li class="program-item inscr ">
  <a class="program-link font-size__enrollment"><label for="tnome">Endereço completo:</label>
    <input type="text" name="tnome" size="30rem" maxlength="55" placeholder="Digite seu endereço"
      required="required" /></a>
</li>
<li class="program-item inscr ">
  <a class="program-link font-size__enrollment"><label for="tnome">Curso:</label>
    <input type="text" name="tnome" size="30rem" maxlength="55" placeholder="Digite o curso"
      required="required" /></a>
</li>
<li class="program-item inscr ">
  <a class="program-link font-size__enrollment"><label for="tnome">Endereço completo:</label>
    <input type="text" name="tnome" size="30rem" maxlength="55" placeholder="Digite seu endereço"
      required="required" /></a>
</li>

<li class="program-item inscr ">
  <a class="program-link font-size__enrollment"><label for="tarq">Carregue sua
    foto</label><br /><br />
    <input type="file" name="tarq" />
  <br /></a>
</li>
<li class="program-item inscr ">
  <a class="program-link font-size__enrollment"><label for="tnome">Quer receber informações sobre
    a Infoeste?</label>
  <br /></a>
</li>
<li class="program-link font-size__enrollment checkbox">
  <input type="radio" name="op" value="1" class="check">Sim
  <input type="radio" name="op" value="2" class="check" />Não<br />
</li>
<br />
<div class="botoes">
  <input class="botao" type="button" name="bmensagem" value="OK"
    onclick="alert('Gravado com sucesso!')" />
  <input class="botao" type="reset" name="blimpar" value="LIMPAR" />

```

Já no css as únicas classes adicionadas foram as para o botão de submit e cores pro input e as letras do placeholder.

CSS

```
.botao {  
  border: 0;  
  background-color: rgba(0, 16, 5, 0.425);  
  border-radius: 1rem;  
  color: #fff;  
  cursor: pointer;  
  font-size: 1.6rem;  
}  
  
.botao:hover {  
  box-shadow: 0 0 5px #83ff89;  
  border: 1px solid #fff;  
}  
  
input {  
  background-color: rgba(0, 16, 5, 0.425);  
  border-radius: 1rem;  
  padding: 1rem 2rem;  
}  
  
input::placeholder {  
  color: #fff;  
}  
  
.program-item.inscr.center {  
  display: flex;
```

Contato e portfólio

Finalizamos a página com a section “contato”, que mostra não só a localização, telefone e Email da FIPP, mas também os portfólios dos integrantes do grupo.

Informações da localização

CONTATE-NOS

UNOESTE - Universidade do Oeste Paulista
FIPP - Faculdade de Informática de Presidente Prudente
 Secretaria da Faculdade de Informática / R. José Bongiovani, 700 - Cidade Universitária
 Bloco H - 1º andar
 Telefone: (18) 3229 - 1060
 Email: infoeste@fipp.unoeste.br



Aba de portfólios

Mapa interativo.



Lívia Gomes



Andressa Diniz



Caio Lomas



Leonardo Neves

Algo interessante a se destacar é a utilização da tag “iFrame” que disponibiliza um mapa interativo com a localização escolhida, neste caso, o Bloco H no Campus 1.

HTML Localização


```

<section class="contato-box flex-column" id="contato">

  <div class="title-contato">
    <h1>Contate-nos</h1>
  </div>
  <div class="dados-fipp">
    <div class="content-dados flex-column">
      <h3>UNOESTE - Universidade do Oeste Paulista</h3>
      <h4>FIPP - Faculdade de Informática de Presidente Prudente</h4>
      <p>Secretaria da Faculdade de Informática / R. José Bongiovani, 700 - Cidade Universitária</p>
      <p>Bloco H - 1º andar</p>
      <p>Telefone: (18) 3229 - 1060</p>
      <p>Email: infoeste@fipp.unoeste.br</p>
    </div>
    <div class="loc-fipp">
      <div>Iframe com link do Google Maps</div>
      <iframe
        src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d3695.821827114494!2d-51.402715!3d-22.132768!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x9493f45b18f82b4f%3A0xc61c929c186b7d81!2sFIPP!5e0!3m2!1spt-BR!2sbr!4v1694959282996!5m2!1spt-BR!2sbr"
        class="loc-fipp-iframe" style="border:0;" allowfullscreen="" loading="lazy"
        referrerpolicy="no-referrer-when-downgrade"></iframe>
    </div>
  </div>
</div>

```

HTML Portfólio

(href para outros arquivos html)

```
<div class="box-cards-profile">
  <div class="card-profile flex-column">
    <div class="img-box-card">
    </div>
    <div class="content-card-profile">
      <div class="name-profile">
        <h2>Livia Gomes</h2>
      </div>
      <div class="details-hidden flex-column">
        <p>Estudante FIPP</p>
        <div><a href="./PortfólioLivia/index.html" class="see-more">Leia Mais</a></div>
      </div>
    </div>
  </div>
  <div class="card-profile flex-column">
    <div class="img-box-card addressa">
    </div>
    <div class="content-card-profile">
      <div class="name-profile">
        <h2>Andressa Diniz</h2>
      </div>
      <div class="details-hidden flex-column">
        <p>Estudante FIPP</p>
        <div><a href="./Portfolio_Andressa/html.html" class="see-more">Leia Mais</a></div>
      </div>
    </div>
  </div>
  <div class="card-profile flex-column">
    <div class="img-box-card caio">
    </div>
    <div class="content-card-profile">
      <div class="name-profile">
        <h2>Caio Lomas</h2>
      </div>
      <div class="details-hidden flex-column">
        <p>Estudante FIPP</p>
      </div>
    </div>
  </div>
</div>
```

CSS box informações de loc.

```

475 .dados-fipp {
476   display: flex;
477   align-items: center;
478   justify-content: center;
479   gap: 4rem;
480   width: 100%;
481 }
482
483 .content-dados {
484   gap: 1rem;
485   background-color: #00160515;
486   padding: 1rem 2rem;
487   border-radius: 1rem;
488   min-height: 25rem;
489 }
490

```

CSS box p/ portfólios.

```

517 .card-profile {
518   position: relative;
519   width: 22rem;
520   height: 12rem;
521   background-color: #fff;
522   border-radius: 2rem;
523   box-shadow: 0 3.5rem 8rem #00000015;
524   transition: 0.5s;
525
526   align-self: end;
527 }
528
529 .card-profile:hover {
530   height: 20rem;
531 }
532
533 .img-box-card {
534   position: absolute;
535   left: 50%;
536   top: -50%;
537   transform: translateX(-50%);
538   width: 12rem;
539   height: 12rem;
540   border-radius: 1rem;
541   box-shadow: 0 3.5rem 8rem #00000035;
542
543   background-image: url("../PortfólioLivia/imgs/eu-verde2.jpg");
544   background-size: cover;
545   background-position: center;

```

Cherrypy

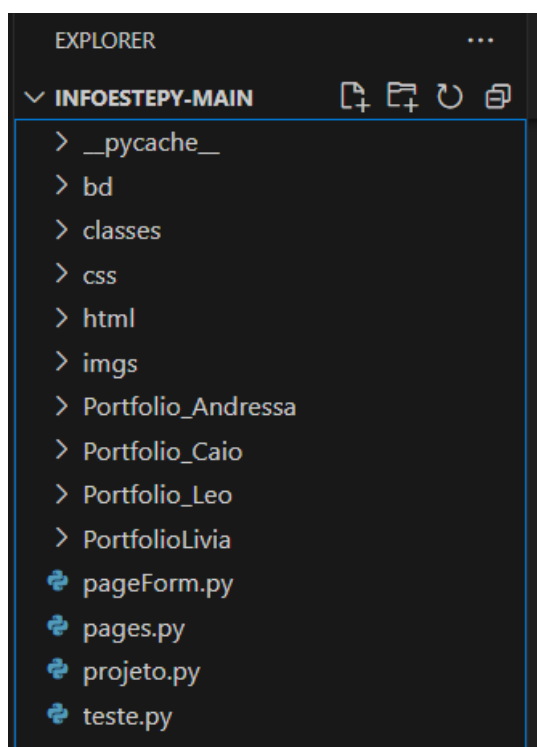
Este projeto aproveita a robustez e a flexibilidade do CherryPy, um framework web em Python, para facilitar a conexão com o banco de dados. Através da estrutura simplificada e eficiente fornecida pelo CherryPy, conseguimos desenvolver uma aplicação de gerenciamento de alunos de forma mais intuitiva e eficaz.

O CherryPy oferece suporte a paradigmas de programação orientada a objetos, o que torna a interação com o banco de dados suave e elegante.

A integração simples e direta com o CherryPy nos permitiu criar operações de gravação, leitura e alteração de dados dos alunos de maneira eficiente, resultando em uma experiência de desenvolvimento mais agradável. Essa escolha estratégica do CherryPy demonstra o compromisso com a eficiência e a facilidade de manutenção em todo o ciclo de vida do projeto.

Pastas e organização

Todo código está separado em 13 pastas e arquivos diferentes, sendo eles:



__pycache__ - Pasta criada automaticamente com alguns arquivos de cache do python.

bd – Scripts de criação do banco, modelo do bd etc.

classes – Módulos e classes com suas respectivas funções.

css – Parte de estilização do frontend.

html – Parte do texto html do frontend.

imgs - Todas as imagens usadas no site, exceto a dos portfólios.

portfólios – Contém tanto o html como as imagens de cada participante do grupo.

pageForm.py – Scripts relacionados ao formulário do site.

pages.py – Toda concatenação e exposição do html do site com o cherrypy.

projeto.py – Arquivo de execução do projeto, contém o import de todas as classes, configura e executa efetivamente o cherrypy. (o teste.py é só um script para testar o form.)

Classes > banco.py

Esse código é uma abstração para interagir com o banco de dados SQLite, dentro dele é criado a classe banco que será chamada posteriormente em outros arquivos para fazer a comunicação do banco de dados com o site.

```
banco.py x
classes > banco.py > Banco

4 class Banco():
5
6     def __init__(self):#construtor
7         self.__conexao = None
8         self.__cursor = None
9
10    def __abrirConexao(self):
11        self.__conexao = sqlite3.connect("bd\\Aluno.db")
12        self.__conexao.row_factory = sqlite3.Row #isso serve para que você possa acessar os dados pelos nomes dos atributos da tabela e r
13        self.__cursor = self.__conexao.cursor()
14
15    def __fecharConexao(self):
16        self.__cursor.close()
17        self.__conexao.close()
18
19    def executarInsertUpdateDelete(self, sql):
20        # Quando não recebeu o comando de SQL para ser executado
21        linhasAfetadas = -10 #variável de controle de erro...
22        if len(sql) > 0:
23            self.__abrirConexao()
24            self.__cursor.execute(sql) #executar no banco
25            linhasAfetadas = self.__cursor.rowcount #número de linhas afetadas pelo comando SQL
26            self.__conexao.commit()#efetuar o sql
27            self.__fecharConexao()
28        return linhasAfetadas
29
30    def executarSelect(self, sql):
31        dados = ''
32        if len(sql) > 0:
33            self.__abrirConexao()
34
35            self.__cursor.execute(sql)
36            dados = self.__cursor.fetchall() #colocar os dados que vieram do BD no dados (retorna o resultado do select) - Lista
37
38            self.__fecharConexao()
39        return dados
```

Após as configurações iniciais de abrir/fechar conexão com o banco, temos duas funções principais, sendo elas:

“def executarInserirUpdateDelete(self,sql):”

“def executarSelect(self,sql):”

O objetivo de cada uma, é, respectivamente -> Executar instruções do SQL como inserir, atualizar ou deletar informações do formulário, o return é o número de linhas afetadas para demonstrar sucesso ou não da operação.

Executar consultas do banco de dados, inicializando uma variável “dados” em lista e depois de executar as instruções do SQL, retorna os dados obtidos.

Classes > aluno.py

Esse código contém a classe aluno que armazena informações privadas como ID, email, telefone e curso. Ele interage com o banco de dados para obter e alterar tais valores através do formulário no html, ou seja, contém logo no início a import da classe banco anteriormente descrita.

```

aluno.py x
classes > aluno.py > ...
1  from classes.banco import Banco
2
3  class Aluno(): # está é o nome da classe de Espécies
4      """
5          Documentação da classe
6          - aqui nós vamos descrever os campos (propriedades) e funções (métodos) para definirmos de acordo com a teoria de Programação O
7      """
8      # Construtor
9      def __init__(self):
10         # propriedades privadas
11         self.__id = 0
12         self.__nome = ''
13         self.__email = ''
14         self.__tel = ''
15         self.__curso = 1
16         self.__banco = Banco() # aqui será criado o objeto que representa o acesso ao Banco de Dados, nós iremos utilizar ela para gravar
17         # propriedades públicas
18
19         # definir os métodos para a nossa classe para colocar os valores nas propriedades
20         def set_id(self, pId): # setar o valor é autoincremento
21             if pId > 0: # validação dos valores para não serem negativos ou zerados, e serem corretamente associados à propriedade
22                 self.__id = pId
23         def set_nome(self, tnome):
24             if len(tnome) > 0:
25                 self.__nome = tnome
26
27         def set_email(self, temail):
28             self.__email = temail
29
30         def set_tel(self, telefone):
31             self.__tel = telefone
32
33         def set_curso(self, selOpcao):
34             self.__curso = selOpcao
35
36         # métodos para obter os valores das propriedades
37         def get_id(self):

```

```

58         ...
59         return self.__banco.executarSelect(sql)
60
61     def gravar(self): # vai pegar os dados do objeto e gravar na tabela do banco
62         sql = ''' INSERT INTO Aluno (aluno_nome,aluno_email,aluno_tel,aluno_curso)
63             values ("%nome", "%email", "%tel", %curso)
64             '''
65         sql = sql.replace('%nome',self.__nome)
66         sql = sql.replace('%email', self.__email)
67         sql = sql.replace('%tel', self.__tel)
68         sql = sql.replace('%curso', self.__curso)
69         return self.__banco.executarInsertUpdateDelete(sql)
70
71     def obterAluno(self, pId=0):
72         if pId != 0:
73             self.__id = pId
74             sql = ''' SELECT aluno_id, aluno_nome, aluno_email, aluno_tel, aluno_curso
75                 FROM Aluno
76                 where aluno_id = #id             '''
77             sql = sql.replace('#id', str(self.__id))
78             return self.__banco.executarSelect(sql)
79
80     def excluir(self):
81         sql = 'delete from Aluno where aluno_id = #id'
82         sql = sql.replace('#id', str(self.__id))
83         return self.__banco.executarInsertUpdateDelete(sql)
84
85     def alterar(self):
86         sql = 'update Aluno set aluno_nome = "%nome", aluno_email = "%email", aluno_tel = "%tel", aluno_curso = "%curso" where aluno_id'
87         sql = sql.replace('%nome',self.__nome)
88         sql = sql.replace('%email',self.__email)
89         sql = sql.replace('%tel',self.__tel)
90         sql = sql.replace('%curso',self.__curso)
91         sql = sql.replace('#id',str(self.__id))
92         return self.__banco.executarInsertUpdateDelete(sql)

```

Aqui temos alguns métodos importantes, como obterAluno, excluir, alterar e gravar.

Nessas funções ocorrem uma comunicação entre a classe banco com a aluno, onde:

A classe Aluno apresenta uma estrutura concebida para interagir com um banco de dados no contexto de um sistema de gerenciamento de alunos. Destacando alguns métodos fundamentais da classe, é possível elucidar as operações que são realizadas em termos científicos.

Método obterAlunos: O método obterAlunos tem por objetivo recuperar informações detalhadas sobre todos os alunos cadastrados no banco de dados. A consulta SQL associada busca selecionar os campos relevantes, como ID do aluno, nome, e-mail, telefone e curso. A execução dessa consulta retorna um conjunto de resultados que reflete a composição da tabela "Aluno". Essa operação é essencial para apresentar uma visão abrangente de todos os alunos registrados no sistema.

Método gravar: O método gravar desempenha um papel crucial na persistência de dados no banco. Ele constrói uma instrução SQL de inserção, incorporando os atributos do objeto Aluno na consulta. Posteriormente, essa instrução é executada, inserindo um novo registro na tabela "Aluno". Esse processo é essencial para a adição de novos alunos ao sistema, assegurando que suas informações sejam armazenadas de maneira consistente.

Método obterAluno: O método obterAluno busca recuperar os detalhes específicos de um aluno identificado pelo seu ID. Semelhante ao método obterAlunos, esta função utiliza uma consulta SQL, porém, desta vez, ela inclui uma cláusula condicional para selecionar apenas o aluno desejado. A resposta dessa consulta retorna os atributos associados ao aluno identificado pelo ID fornecido. Esse método é crucial para a obtenção de informações detalhadas sobre um aluno específico.

Método excluir: O método excluir desencadeia a remoção de um aluno específico do banco de dados. Ao construir e executar uma instrução SQL de exclusão, a classe Aluno realiza a remoção do registro associado ao aluno, garantindo que o banco de dados permaneça atualizado e coerente. Este método é vital para a manutenção da integridade do sistema ao permitir a exclusão de alunos.

Método alterar: O método alterar propicia a modificação dos dados de um aluno no banco de dados. Ao construir uma instrução SQL de atualização, incorporando os atributos atualizados do aluno, a classe Aluno executa a alteração do registro na tabela "Aluno". Esse método é crucial para a atualização dinâmica das informações dos alunos no sistema, refletindo mudanças relevantes.

Formulário:

pageForm.py

Este código usa o CherryPy para carregar o html do formulário (html/headerForm.html) e o executa com `@cherrypy.expose()`.

Em seguida o método `montaFormulario` é criado para efetivamente concatenar e exibir o html do formulário, ele gera um formulário HTML para inserir dados do aluno. O formulário possui campos para nome, e-mail, telefone, curso, etc. Os valores dos campos são preenchidos com os dados passados como parâmetros (`pld`, `tnome`, `temail`, `telefone`, `selOpcao`).

Depois há o método `montaTabela`, que gera uma tabela em HTML que exibe os dados cadastrados no banco de dados. Utiliza a classe "Aluno" para obter os dados dos alunos a partir do banco.

Por fim, após o conteúdo do frontend ser carregado, há a criação das funções em python para gravar, excluir e alterar aluno. Elas são chamadas quando há um submit no formulário.

Trecho onde começa o arquivo python e carrega o html:

```

pageForm.py X
pageForm.py > PaginaForm > gravarAluno
1 import cherrypy
2 from classes.aluno import *
3
4
5 class PaginaForm():
6     header = open("html/headerForm.html", encoding="utf-8").read()
7
8     @cherrypy.expose()
9     def index(self):
10         return self.montaFormulario()
11
12     def montaFormulario(self, pId=0, tnome='', temail='', telefone='', selOpcao=1):
13         str = self.header
14         str += '''
15         <section class="programacao-box-cards inscricao" id="inscricao">
16
17             <head class="program-container center">
18
19                 <div class="program-container flex-column center">
20                     <div class="cards-container-inscricao">
21                         <h1 class="title-box-program">Faça já sua inscrição!</h1>
22                         <div class="card_enrollment">
23                             <div class="program-card_enrollment card1_enrollment">
24                                 <div class="content-box-card_enrollment">
25                                     <div class="content-descr-card_enrollment">
26                                         <p class="card-content_enrollment">
27
28                                             <form name="Cadastro" action="gravarAluno" method="post"><br /><br />
29                                             <input type="hidden" id="txtId" name="txtId" value="%d"/>
30                                             <ul class="list-program">
31                                                 <li class="program-item inscr "><label class="font-size_enrollm
33                                                 for="tnome">Nome:</label>
34                                                 <input type="text" id="tnome" name="tnome" size="30" maxlength="55"
35                                                 placeholder="Digite seu nome" required="required" value="%s" /></a>
36                                                 </li>

```

Trechos onde são definidas as funções de criação, exclusão e alteração do formulário:

```

pageForm.py X
pageForm.py > PaginaForm > gravarAluno
124 # vai ser chamado quando clicar no botão
125 @cherrypy.expose()
126 def gravarAluno(self, txtId, tnome, temail, telefone, selOpcao, bgravar):
127     if len(tnome) > 0:
128         # fazer os procedimentos para gravar
129         objAluno = Aluno()
130         objAluno.set_nome(tnome)
131         objAluno.set_email(temail)
132         objAluno.set_tel(telefone)
133         objAluno.set_curso(selOpcao)
134         # se o txtId = 0, representa que estamos inserindo uma nova espécie
135         retorno = 0
136         if int(txtId) == 0: # nova espécie
137             retorno = objAluno.gravar()
138         else: # vai gravar uma alteração no banco de dados
139             objAluno.set_id(int(txtId))
140             retorno = objAluno.alterar()
141         if retorno > 0:
142             return '''
143             <script>
144                 alert("O aluno %s foi gravado com sucesso!!")
145                 window.location.href = "/rotaAluno"
146             </script>
147             ''' % (tnome)
148         else: # if retorno > 0: - Quer dizer que deu erro na hora de gravar
149             return '''
150             <h2> Erro ao gravar o aluno %s</h2>
151             <a href="/rotaAluno">voltar</a>
152             ''' % (tnome)
153         else: # len(txtDescr) > 0:
154             return '''
155             <h2> O nome do aluno deve ser informado</h2>
156             <a href="/rotaAluno">voltar</a>
157             ...
158
159     @cherrypy.expose()
160     def excluirAluno(self, idAluno):

```

```

pageForm.py X
pageForm.py > ...
150         <h2> Erro ao gravar o aluno %s</h2>
151         <a href="/rotaAluno">voltar</a>
152         ... % (tnome)
153     else: # len(txtDescr) > 0:
154         return '''
155         <h2> O nome do aluno deve ser informado</h2>
156         <a href="/rotaAluno">voltar</a>
157         ...
158
159     @cherry.py.expose()
160     def excluirAluno(self, idAluno):
161         objAluno = Aluno()
162         objAluno.set_id(int(idAluno))
163         if objAluno.excluir() > 0: # informa se conseguiu excluir ou não
164             # para atualizar a página após a exclusão (tipo reload)
165             raise cherry.py.HTTPRedirect('/rotaAluno')
166         else:
167             return '''
168             <h2>Não foi possível excluir o aluno!!</h2>
169             [ <a href="/rotaAluno">Voltar</a>]
170             ...
171
172     @cherry.py.expose()
173     def alterarAluno(self, idAluno):
174         objAluno = Aluno()
175         # buscar no banco a espécie que foi informada no parâmetro
176         dadosAlunoSelec = objAluno.obterAluno(idAluno)
177         # chamar o método para montar o formulário com os dados da espécie selecionada na tabela e carregar nos elementos <input> do form
178         return self.montaFormulario(dadosAlunoSelec[0]['aluno_id'],
179                                     dadosAlunoSelec[0]['aluno_nome'],
180                                     dadosAlunoSelec[0]['aluno_email'],
181                                     dadosAlunoSelec[0]['aluno_tel'],
182                                     dadosAlunoSelec[0]['aluno_curso']
183                                     )
184

```

```

@cherry.py.expose()
def gravarAluno(self, txtId, tnome, temail, telefone, selOpcao, bgravar):
    if len(tnome) > 0:
        #valida email
        if not re.match(r"^[^@]+@[^@]+\.[^@]+", temail):
            return '''
            <script>
            | alert("O formato do e-mail é inválido");
            | window.location.href = "/rotaAluno";
            | </script>
            ...

        #valida tel
        if not re.match(r"\d{10,11}", telefone):
            return '''
            <script>
            | alert("O formato do telefone é inválido. Deve conter 10 ou 11 dígitos");
            | window.location.href = "/rotaAluno";
            | </script>
            ...

        # fazer os procedimentos para gravar
        objAluno = Aluno()
        objAluno.set_nome(tnome)
        objAluno.set_email(temail)
        objAluno.set_tel(telefone)
        objAluno.set_curso(selOpcao)

        retorno = 0
        if int(txtId) == 0:
            retorno = objAluno.gravar()
        else: # vai gravar uma alteração no banco de dados
            objAluno.set_id(int(txtId))
            retorno = objAluno.alterar()
        if retorno > 0:
            return '''
            <script>
            | alert("O aluno %s foi gravado com sucesso!!")
            | window.location.href = "/rotaAluno"
            | </script>
            ... % (tnome)
        else: # if retorno > 0: - Quer dizer que deu erro na hora de gravar
            return '''
            <h2> Erro ao gravar o aluno %s</h2>
            <a href="/rotaAluno">voltar</a>
            ... % (tnome)
    else: # len(txtDescr) > 0:
        return '''
        <h2> O nome do aluno deve ser informado</h2>

```

Nesse código acima, de `gravarAluno()`, possui `Regex`, onde é necessário realizar a importação de sua biblioteca “`re`”, para fazer as seguintes validações em específico:

1. Validação de E-mail:
 - a. Utiliza a expressão regular `^[^@]+@[^@]+\.[^@]+` para validar o formato do e-mail.
 - b. A expressão regular verifica se o e-mail contém pelo menos um caractere antes e depois do símbolo '@' e um ponto '.' depois do símbolo '@'.
 - c. Se o formato do e-mail for inválido, exibe um alerta ao usuário e redireciona para a página `"/rotaAluno"`.

2. Validação de Telefone:
 - a. Utiliza a expressão regular `\d{10,11}` para validar o formato do telefone.
 - b. A expressão regular verifica se o telefone contém entre 10 e 11 dígitos numéricos.
 - c. Se o formato do telefone for inválido, exibe um alerta ao usuário e redireciona para a página `"/rotaAluno"`.

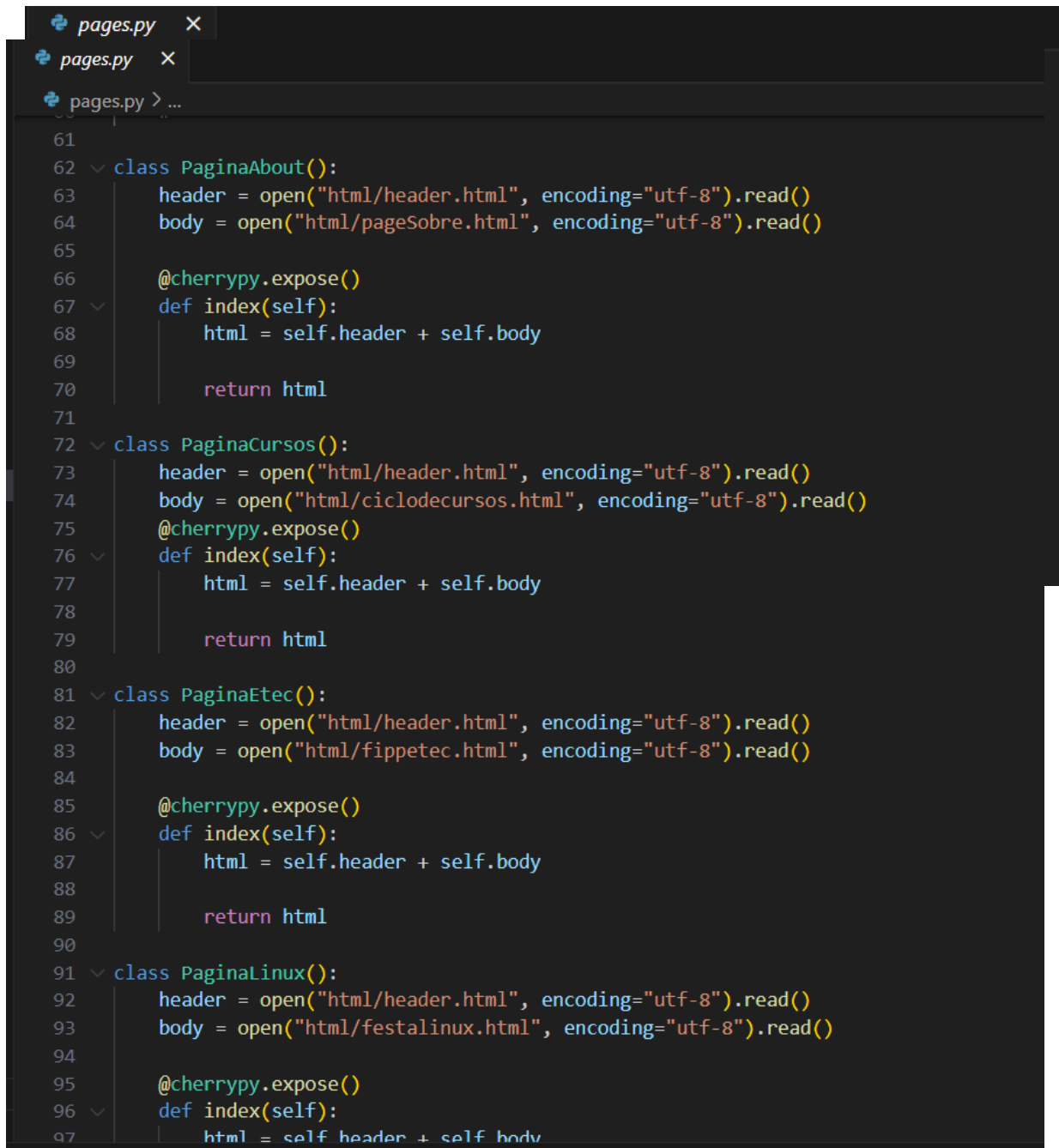
Ainda dentro do `PageForm.py`, no processo de gravação de um aluno, um objeto da classe `Aluno` é instanciado e tem seus atributos configurados com os dados fornecidos. A lógica inclui a verificação para determinar se o aluno é novo, identificado pelo valor de `txtId` igual a 0, ou se trata-se de uma alteração no banco de dados. Com base nessa distinção, são chamados os métodos `gravar()` ou `alterar()` da classe `Aluno`, responsáveis por efetuar a gravação dos dados. Após essa operação, um alerta é exibido ao usuário para informar o resultado da ação, sendo este redirecionado em seguida para a página `"/rotaAluno"`.

No cenário em que o nome do aluno não é fornecido, é realizado um tratamento específico. A aplicação exibe uma mensagem indicando a obrigatoriedade do campo “nome” e fornece um link para retornar à página `"/rotaAluno"`, permitindo que o usuário corrija o erro no preenchimento do formulário.

pages.py

Nesse arquivo python é definida uma classe `index` que carrega alguns html’s “chave” que compõem o site, e em seguida uma função que as expõe ao `cherrypy` é criada, e então logo abaixo acontece o mesmo só que com outros html’s “secundários”.

Criação do index com os html's "chave":



```

61
62 class PaginaAbout():
63     header = open("html/header.html", encoding="utf-8").read()
64     body = open("html/pageSobre.html", encoding="utf-8").read()
65
66     @cherry.py.expose()
67     def index(self):
68         html = self.header + self.body
69
70         return html
71
72 class PaginaCursos():
73     header = open("html/header.html", encoding="utf-8").read()
74     body = open("html/ciclodecursos.html", encoding="utf-8").read()
75
76     @cherry.py.expose()
77     def index(self):
78         html = self.header + self.body
79
80         return html
81
82 class PaginaEtec():
83     header = open("html/header.html", encoding="utf-8").read()
84     body = open("html/fippetec.html", encoding="utf-8").read()
85
86     @cherry.py.expose()
87     def index(self):
88         html = self.header + self.body
89
90         return html
91
92 class PaginaLinux():
93     header = open("html/header.html", encoding="utf-8").read()
94     body = open("html/festalinux.html", encoding="utf-8").read()
95
96     @cherry.py.expose()
97     def index(self):
98         html = self.header + self.body
  
```

Criação das outras funções de chamada com os html's "secundários":

projeto.py

```

projeto.py x
projeto.py > ...
1  import cherrypy
2  import os
3  from pages import *
4  from pageForm import *
5
6  local_dir = os.path.dirname(__file__)
7
8  #run
9  index()
10
11  server_config={
12  'server.socket_host': '127.0.0.1',
13  'server.socket_port': 80
14  }
15  cherrypy.config.update(server_config)
16
17  #Para que o cherrypy possa encontrar os arquivos dentro do diretório da aplicação
18  local_config = {
19      "/": {
20          "tools.staticdir.on": True,
21          "tools.staticdir.dir": local_dir,
22      },
23      "/#": {
24          "request.dispatch": cherrypy.dispatch.MethodDispatcher(),
25      }
26  }
27
28
29
30  #objetos utilizados para rota de navegação
31  root = index() #rota principal
32  root.sobre = PaginaAbout()
33  root.ciclodecursos = PaginaCursos()
34  root.fippetec = PaginaEtec()
35  root.festalinix = PaginaLinux()
36  root.maratonaprogramacao = PaginaMaratona()
37  root.rotaAluno = PaginaForm()

```

Aqui é a “main” do projeto. Ela configura o servidor cherrypy e chama todas as classes e métodos mostrados anteriormente, orquestrando os seguintes elementos:

1. Importação de Módulos: Os módulos essenciais, como cherrypy, os, pages, e pageForm, são importados para possibilitar o desenvolvimento e execução da aplicação web.
2. Definição do Diretório Local: A variável local_dir é configurada para armazenar o diretório local do arquivo em execução, facilitando o acesso a recursos locais e configurações.
3. Inicialização da Aplicação: A função index() é chamada para iniciar a aplicação, preparando o ambiente e definindo rotas iniciais.
4. Configuração do Servidor: Um conjunto de parâmetros, denominado server_config, é definido para configurar o servidor CherryPy. Esses parâmetros

incluem o endereço IP ('127.0.0.1') e a porta (80) utilizados para acessar a aplicação web.

5. Configuração de Arquivos Estáticos: O dicionário `local_config` é criado para especificar a configuração relacionada aos arquivos estáticos (como imagens, folhas de estilo e scripts) presentes no diretório da aplicação. Isso é fundamental para que o CherryPy possa servir esses arquivos corretamente.

6. Criação de Objetos de Rota: São criados objetos representando diferentes rotas da aplicação, como a página inicial, a página "Sobre", ciclos de cursos, informações sobre a Etec, eventos relacionados a Linux, maratona de programação e a rota para manipulação de dados de alunos.

7. Inicialização do Servidor CherryPy: A função `cherrypy.quickstart()` é utilizada para iniciar o servidor CherryPy, com a raiz da aplicação configurada como root e as configurações locais especificadas por `local_config`. Isso inicia o servidor web e torna a aplicação acessível através do endereço definido no `server_config`.