Pontifácia Universidade do Paraná

Leonardo Lino Guedes

Banco de Dados Faculdade

Curitiba

2024

Leonardo Lino Guedes

Banco de dados Faculdade

Implementando Interface gráfica

Ciência da Computação

Trabalho desenvolvido com o intuito de manusear um banco de dados através de uma interface gráfica.

Professor Rodrigo

Curitiba, Paraná

2024

Resumo

Ao utilizar o SQLAlchemy para construção de classes para um banco de dados foi necessário implementar um que fosse conectado pela engrenagem fornecida pela própia biblioteca para fazer a criação dessas tabelas. Posteriormente, a criação de uma interface que interagisse com esses cortes de código e o banco de dados foi o próximo passo, para isso ultilizei o QTpython e suas bibliotecas para criar uma janela com abas que fizessem chamadas das operações CRUDs mais tarde definidas, de forma que fosse possível controlar o que acontecia no banco de dados a partir dela, para isso foi necessário fazer as conexões com os inputs existentes na interface e os botões com suas ações particulares. Grande parte do projeto foi mantido e grande parte foi alterado a fins da nova implementalçao da interface gráfica, para o funcionamento congruente das novas linhas de código da interface montada.

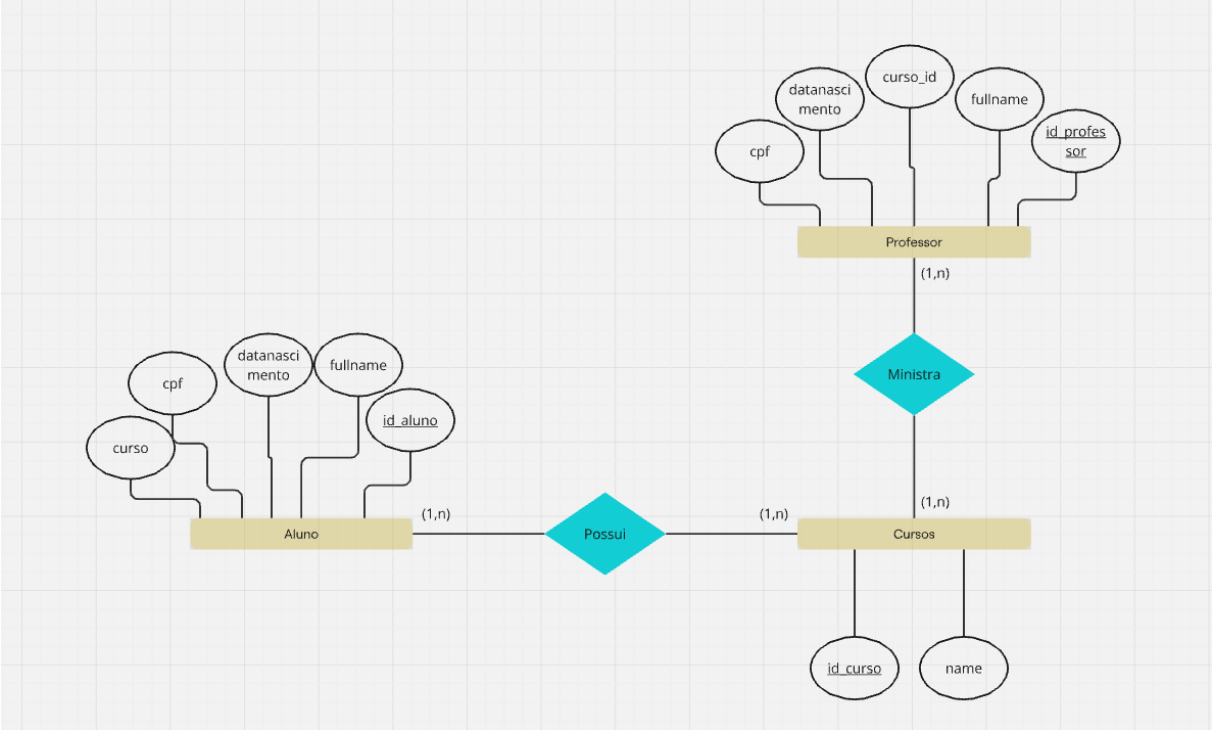
Sumário

Introdução

Neste trabalho abordarei como intruduzir uma interface gráfica a um sistema de banco de dados que realiza operações CRUD. Nele optei por ultilizar QTpython e SQLAlchemy para contrução do banco de dados e da interface. Mostrarei mais a fundo o que a lógica dos códigos escritos produzem e como interagem umas com as outras.

1 Modelo Conceitual

Neste modelo conceitual foi desenvolvido uma relação em que mostrasse estudantes e professores pertecentes a um determinado curso, com a tabela alunos e professores sendo conectadas a curso.



2 Bibliotecas

Foi adicionado as depêndencias necessárias para o funcionamento do código e removido outras que acabaram por ficar a toa em comparação ao código antigo. Dessa forma foi possível ter melhot organização das linhas de código e deixar o projeto mais limpo.

```
1  import sys
2  from PyQt5 import QtWidgets, uic
3  from sqlalchemy.orm import sessionmaker
4  from sqlalchemy import create_engine, Integer, String, Date
5  from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column
6  from datetime import datetime
7
```

3 Conexões

Aqui é abordado o conceito de conexão com o banco de dados e as sessoes criadas para fazer as alterações e comunicações entre o código e o BD.

```
8   #Conexão com o banco de dados
9   engine = create_engine("sqlite:///data_bank.db", echo=True)
10  Session = sessionmaker(bind=engine)
```

4 Tabelas

Nesta parte de criação das classes que crião as tabelas tudo se manteve igual, mantendo suas colunas e atribuições a elas.

```
13  class Base(DeclarativeBase):
14      pass
15
16  class Aluno(Base):
17      __tablename__ = "Alunos"
18      id: Mapped[int]                 = mapped_column(Integer, primary_key=True, aut
19      fullname: Mapped[str]           = mapped_column(String(30), nullable=False)
20      datanascimento: Mapped[Date]    = mapped_column(Date)
21      cpf: Mapped[str]                = mapped_column(String, unique=True, nullable=
22
23  class Professor(Base):
24      __tablename__ = "Professores"
25      id: Mapped[int]                 = mapped_column(Integer, primary_key=True, aut
26      fullname: Mapped[str]           = mapped_column(String(30), nullable=False)
27      curso: Mapped[str]              = mapped_column(String(30), nullable=False)
28      datanascimento: Mapped[Date]    = mapped_column(Date)
29      cpf: Mapped[str]                = mapped_column(String, unique=True, nullable=
30
31  class Curso(Base):
32      __tablename__ = "Cursos"
33      id: Mapped[int]         = mapped_column(Integer, primary_key=True, autoincreme
34      name: Mapped[str]       = mapped_column(String(30), nullable=False)
35
36  #Cria as tabelas no BD
37  Base.metadata.create_all(engine)
```

5 Interface Gráfica

Chegamos na parte do código que é iniciada a criação da interface gráfica que vai interagir com o banco de dados, montando inicialmente seu tamanho, as de mais abas de inserção de cada tabela no banco de dados e a estruturação de como elas vão ser montadas.

```
40  class CrudApp(QtWidgets.QTabWidget):
41      def __init__(self):
42          super().__init__()
43          self.init_ui()
44
45      def init_ui(self):
46          #abas do Aluno, Professor e Curso
47          self.tab_aluno = QtWidgets.QWidget()
48          self.tab_professor = QtWidgets.QWidget()
49          self.tab_curso = QtWidgets.QWidget()
50
51          self.addTab(self.tab_aluno, "Aluno")
52          self.addTab(self.tab_professor, "Professor")
53          self.addTab(self.tab_curso, "Curso")
54
55          #Chama as funções que criam as interfaces das abas
56          self.init_aluno_tab()
57          self.init_professor_tab()
58          self.init_curso_tab()
59
60          self.setWindowTitle("CRUD Aluno, Professor e Curso")
61          #definindo o tamanho inicial da janela e posição
62          self.setGeometry(100, 100, 600, 400)
63
```

5.1 Aba Aluno

Nesta criação de aba, será feito a criação da aba de alunos, alinhando os elementos de input, os botões e as colunas da tabela e como serão exibidas, de forma que fique intuitivo como interagir com a interface.

```
65      def init_aluno_tab(self):
66          layout = QtWidgets.QVBoxLayout() #usado para empilhar os widgets de froma
67
68          #Inputs de Aluno
69          self.input_nome_aluno = QtWidgets.QLineEdit()
70          self.input_nome_aluno.setPlaceholderText("Nome do Aluno")
71          self.input_birth_aluno = QtWidgets.QLineEdit()
72          self.input_birth_aluno.setPlaceholderText("Data de Nascimento (dd-mm-yyyy)
73          self.input_cpf_aluno = QtWidgets.QLineEdit()
74          self.input_cpf_aluno.setPlaceholderText("CPF")
```

5.2 Funções Aluno

A seguir ja são definidas as funções de CRUD utilizadas e adaptadas para a tabela de alunos, que foram conectadas com as ações da interface gráfica para responderem ao serem chamadas com o envio de novas alterações.

```
111     def create_aluno(self):
112         session = Session()
113         nome = self.input_nome_aluno.text()
114         nascimento = self.input_birth_aluno.text()
115         cpf = self.input_cpf_aluno.text()
116 
117         try:
118             aluno = Aluno(  fullname=nome,
119                             datanascimento=datetime.strptime(nascimento, "%d-%m-%Y
120                             cpf=cpf)
121             session.add(aluno)
122             session.commit()
123             self.read_aluno()  #Atualiza a tabela
124             self.clear_inputs(self.input_nome_aluno, self.input_birth_aluno, self.
125         except Exception as e:
126             print(f"Erro ao criar aluno: {e}")
127             session.rollback()
128         finally:
129             session.close()
130 
131 >   def read_aluno(self):···
142 
143 >   def update_aluno(self):···
161 
162 >   def delete_aluno(self):···
```

5.3 Aba Professor

Aqui o padrão se repete como a "Aba Alunos"apenas com adaptações à tabela de Professor.

```
177     def init_professor_tab(self):
178         layout = QtWidgets.QVBoxLayout()
179 
180         #Inputs de Professor
181         self.input_nome_professor = QtWidgets.QLineEdit()
182         self.input_nome_professor.setPlaceholderText("Nome do Professor")
183         self.input_birth_professor = QtWidgets.QLineEdit()
184         self.input_birth_professor.setPlaceholderText("Data de Nascimento (dd-mm-y
185         self.input_cpf_professor = QtWidgets.QLineEdit()
186         self.input_cpf_professor.setPlaceholderText("CPF")
187         self.input_curso_professor = QtWidgets.QLineEdit()
188         self.input_curso_professor.setPlaceholderText("Curso")
```

5.4 Funções Professor

Da mesma forma, o padrão se repete ao de Funções Alunos apenas com adaptações para a tabela de Professor.

```
226     def create_professor(self):
227         session = Session()
228         nome = self.input_nome_professor.text()
229         nascimento = self.input_birth_professor.text()
230         cpf = self.input_cpf_professor.text()
231         curso = self.input_curso_professor.text()
232 
233         try:
234             professor = Professor(  fullname=nome,
235                                     datanascimento=datetime.strptime(nascimento, "
236                                     cpf=cpf, curso=curso)
237             session.add(professor)
238             session.commit()
239             self.read_professor()  #atualiza a tabela
240             self.clear_inputs(self.input_nome_professor, self.input_birth_professo
241         except Exception as e:
242             print(f"Erro ao criar professor: {e}")
243             session.rollback()
244         finally:
245             session.close()
246 
247 >   def read_professor(self):···
259 
260 >   def update_professor(self):···
280 
281 >   def delete_professor(self):···
```

5.5 Aba Cursos

O mesmo padrão de criação das ultimas ambas abas é mantido, apenas adaptado a tabela Cursos.

```
296     def init_curso_tab(self):
297         layout = QtWidgets.QVBoxLayout()
298 
299         #inputs de Curso
300         self.input_nome_curso = QtWidgets.QLineEdit()
301         self.input_nome_curso.setPlaceholderText("Nome do Curso")
```

5.6 Funções Cursos

A mesma lógica CRUD é aplicada a estas funções, apenas sendo adaptada para a tabela cursos.

```
336     def create_curso(self):
337         session = Session()
338         nome = self.input_nome_curso.text()
339 
340         try:
341             curso = Curso(name=nome)
342             session.add(curso)
343             session.commit()
344             self.read_curso()  #Atualiza a tabela
345             self.clear_inputs(self.input_nome_curso)
346         except Exception as e:
347             print(f"Erro ao criar curso: {e}")
348             session.rollback()
349         finally:
350             session.close()
351 
352 >   def read_curso(self):···
361 
362 >   def update_curso(self):···
376 
377 >   def delete_curso(self):···
```

6 Executando a Aplicação

Por fim, executamos o código que utilizamos para criar toda a interface e iniciar a execução dela de modo que ela interaja com o banco de dados fazendo operações CRUD.

```
395 #Executando a aplicação
396 if __name__ == "__main__":
397     app = QtWidgets.QApplication(sys.argv)
398     window = CrudApp()
399     window.show()
400     sys.exit(app.exec_())
401 
```

Referências

www.pythonguis.com/
doc.qt.io/
www.youtube.com
www.youtube.com
www.youtube.com
www.youtube.com
www.youtube.com