Relatório

Linguagem/bibliotecas:

Para realizar as nossa consultas na aplicação escolhemos a linguagem de programação Python, por ser fácil de codificar e possuir boas bibliotecas para se conectar com o banco de dados e imprimir tabelas.

As bibliotecas utilizadas foram:

- Psycopg2 - biblioteca utilizada para conectar com o banco de dados
- Prettytable - biblioteca utilizada para imprimir tabelas bem formatadasa no terminal
- os - biblioteca utilizada para possibilitar comandos Unix a partir de um código Python
- passlib.hash - biblioteca para gerar as senhas em SHA256

Conexão com o banco de dados:

Como banco de dados utilizamos o SGBD PostgreSQL e para administrá-lo e fazer testes o pgAdmin. Além dessas ferramentas , como dito anteriormente, usamos a biblioteca Psycopg que nos permite estabelecer uma conexão com o banco de dados utilizando comandos em Python da seguinte forma:

```
import psycopg2  # Modulo para conectar com o banco de dados

# Abre conexão com o banco de dados goodreads
conn = psycopg2.connect(host='127.0.0.1', #ip do host
                        password='12345',
                        user='postgres',
                        dbname='goodreads')#nome do banco de dados

# Criação de um cursor para especificação de comandos SQL
cursor = conn.cursor()

...

# Encerra conexão com o banco de dados
cursor.close()
conn.close()
```

Consultas e Inserções:

Consultas e inserções são feitas apartir do Psycopg2 e podem ser executadas da seguintes formas:

Consultas:

```
# Exemplo de consulta com passagem de parâmetro
cursor.execute("SELECT * FROM exemplo WHERE foo = %s;", (parametro,))

# O método fetchall() devolve todos as linhas resultantes da query
# em uma estrutura de dados do tipo Lista onde cada elemento da lista
# é uma tupla representativa das linhas resultantes da consulta.
result = cursor.fetchall()
```

Inserções:

```
cursor.execute("INSERT INTO foo(bar, gar) VALUES (%s,%s);", (val1, val2,))
#como precisamos que os valores sejam inseridos no banco

# Atualiza as modificações no banco de dados
conn.commit()

#como o nosso SI utiliza o padrão SERIAL em grande parte das tabelas
#é interessante que retornemos o valor gerado pela chave primária, para
#fazer inserções encadeadas
cursor.execute("""INSERT foo(bar, gar) VALUES (%s,%s)
                  RETURNING foo.cod""", (val1, val2,))
# fetchone() funciona de forma semelhante a fetchall() exceto que apenas uma
# tupla representando a primeira linha da consulta é adicionada a lista resultante
codigo_inserido = cursor.fetchone()

cursor.execute("""INSERT rel_foo(fkey_gar, fkey_bar) VALUES (%s, %s)""", (codigo_inserido[0], outro_codigo,))

# Atualiza as modificações no banco de dados
conn.commit()
```

Estrutura de dados e procedimentos:

Não criamos ou utilizamos uma estrutura de dados complexa a ponto de ser explicada, no mais foi usado estruturas padrões do Python como tuplas e listas para armazenar valores das tabelas para mostrar no terminal.

Como se trata de um programa no terminal, foram criadas vários menus que são navegados através de opções que são selecionadas pelo usuários, desta forma utilizamos alguns procedimentos parecidos para pegar as opções do usuário como:

```
import os

# Comando Unix executado através do método system() para limpar a tela do terminal
os.system('clear')

# Para dar uma ideia de menu, entramos em loop até que o usuário digite algo válido ou algo que saia do menu
menu = True


while menu:
  message = "Hello world!"
  opt = input(message) #pegamos a opção do usuário e com isso criamos outros menus, consultamos e inserimos valores em tabelas e

  if opt == '1':
    do_1()
    menu = False
  elif opt == '2':
    do_2()
    menu = False
  else:
    print('Comando inválido')
```