IMPLEMENTAÇÃO DO REPOSITORY PATTERN PARA A INTERAÇÃO COM BANCO DE DADOS UTILIZANDO PYTHON E SQLITE3: ÊNFASE NO MÉTODO DAS VELOCIDADES INDEXADAS

JAMILSON DO NASCIMENTO[1]; ARLENE FEHRENBACH[2]; GEORGE MARINO SOARES GONÇALVES[3] GUILHERME KRUGER BARTELS[4], LUKAS DOS SANTOS BOEIRA[5], GIBERTO LOGUERCIO COLLARES[6]

[1]Universidade Federal de Pelotas – jamilson.nascimento.ufpel@gmail.com
[2]Universidade Federal de Pelotas – arlenefehrenbach@outlook.com
[3]Universidade Federal de Pelotas – george.marino.goncalves@gmail.com
[4]Universidade Federal de Pelotas – guilhermebartels@gmail.com
[5]Universidade Federal de Pelotas – lukasdossantosboeira@gmail.com
[6]Universidade Federal de Pelotas – gilbertocollares@gmail.com

1. INTRODUÇÃO

O paradigma de programação adotado no processo de desenvolvimento de algoritmos para a composição de um sistema, conduz o padrão estrutural de escrita do código e estabelece diretrizes adaptáveis para a sua implementação. No Paradigma Imperativo (também chamado de programação procedural), a estruturação do sistema é realizada em módulos, funções e rotinas. Estes elementos do sistema, distintos entre comportamentos e atributos, podem ser chamados a qualquer momento dentro de seu arquivo principal, sendo assim, executados em maneira sequencial, de cima para baixo, da esquerda para direita (SILVA *et al., 2019;* BRÊDA, 2020; NASCIMENTO *et al., 2021*).

No Paradigma Orientado a Objetos (POO), a estruturação do sistema é realizada por meio da representação de objetos do mundo real, abstratos ou não, e implementadas em forma de Classes. Cada Classe possui, nome, atributos e/ou comportamentos (métodos), que as caracterizam. Estas Classes podem ser utilizadas no sistema por meio de suas instâncias, sem a necessidade de terem todos seus procedimentos executados de maneira sequencial. Tal paradigma tem se mostrado eficiente em sistematizar e propor soluções para diversas aplicações na engenharia e gestão de recursos hídricos (ZAMBONI *et al., 2005;* ALMEIDA *et al., 2008;* LOTT e PHILLIPS, 2021).

De modo geral, um sistema computacional pode ser simplificado em três processos, sendo eles a *entrada*, o *tratamento lógico* e a *saída* de dados, podendo contar com um sistema de gerenciador de banco de dados (SGBD), responsáveis por executar comandos *Standard Query Language* (SQL) e aproximar modelos de banco de dados relacionais ao paradigma de programação adotado. Desta forma, este trabalho tem por objetivo apresentar a implementação de uma Classe padronizada e reutilizável utilizando os conceitos do *Repository Pattern*, responsável por isolar o tratamento lógico necessário antes de executar o registro no banco de dados, com o auxílio de Python e do SGBD SQLite 3. A aplicação se dará no método das velocidades indexadas como seu domínio (EVANS, 2003; FARRELL, 2021; NASCIMENTO *et al., 2021*), o qual utiliza dados de equipamentos acústicos para medição de vazão em rios e canais.

2. METODOLOGIA

A modelagem do banco de dados foi realizada através da representação de suas entidades em diagramas de relação (Figura 1). Foram definidos os atributos

que as compõe após realização de tratamento lógico, assim como seus tipos e propriedades. Ambas entidades possuem o atributo cod do tipo inteiro, com incremento automático, servindo como chave primária de identificação (PK) para dat_table e chave estrangeira para mat_table. A entidade dat_table possui um atributo do tipo texto (*date_time*), não nulo, com espaço para 10 caracteres e dois atributos do tipo numérico (*velocity_x e level*). A entidade mat_table possui três atributos do tipo numérico (*flow_rate, area* e *mean_velocity*) não nulos. Todos atributos numéricos de ambas as classes possuem espaço para 10 caracteres e precisão de três dígitos. A relação entre as entidades é de 1 para 1.

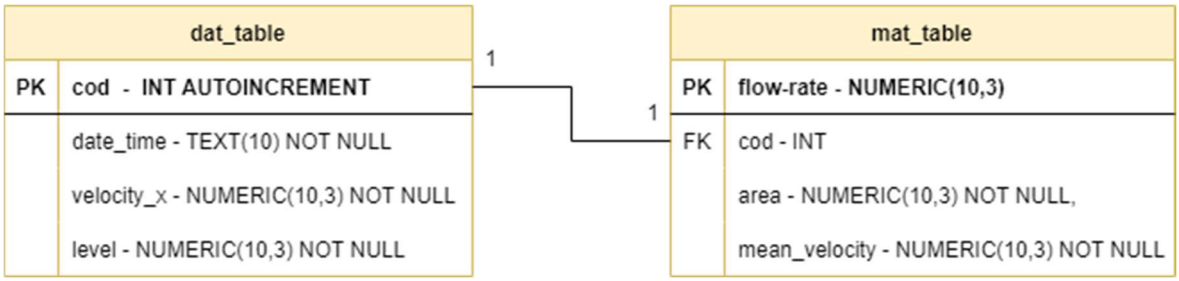


Figura 1 – Diagramas de relação *dat_table e mat_table.*
Fonte: Autores.

Posteriormente, foram realizadas a implementação das Classes *DBConnectionHandler* (SQLITE.org, 2022), responsável pela conexão com o SGBD, *DatEntity e MatEntity,* responsáveis pela criação de suas respectivas tabelas de registros dentro do SGBD, e por fim, *MatEntityRepository e DatEntityRepository,* responsáveis pelo encapsulamento da lógica de tratamento dos dados, antes de serem inseridos em suas tabelas por meio execução de instruções SQL. A Figura 2 ilustra o padrão adotado nas implementações de *DBConnectionHandler e* uma classe *Repository Pattern,* considerando uma entidade qualquer.

```python
import logging
import sqlite3


class DBConnectionHandler:
    """
    Conexão com a base de dados SQLite3
    """

    def __init__(self, file_name) -> None:
        self.file_name = file_name
        self.connection = sqlite3.connect(self.file_name)

    def __enter__(self):
        logging.info("Calling __enter__")
        return self.connection.cursor()

    def __exit__(self, exc_type, exc_value, traceback):
        logging.info("Calling __exit__")
        self.connection.commit()
        self.connection.close()
```

```python
class EntityRespository:

    def insert_data(self) -> List:
        """
        param - None
        return - Lista com modelo de entidade
        """
        with DBConnectionHandler(file_name="DataBase.db") as cursor :
            # Lógica de tratamento dos dado

            cursor.execute("""
            Comando SQL
            """)
            return List
```

Figura 2 – Implementação dos padrões das Classes DBConnectionHandler e EntityRepository.
Fonte: Autores.

3. RESULTADOS E DISCUSSÃO

A Figura 3 apresenta a relação entre as Classes no sistema, considerando a metodologia gráfica *Unified Modeling Language* (GUEDES, 2018).

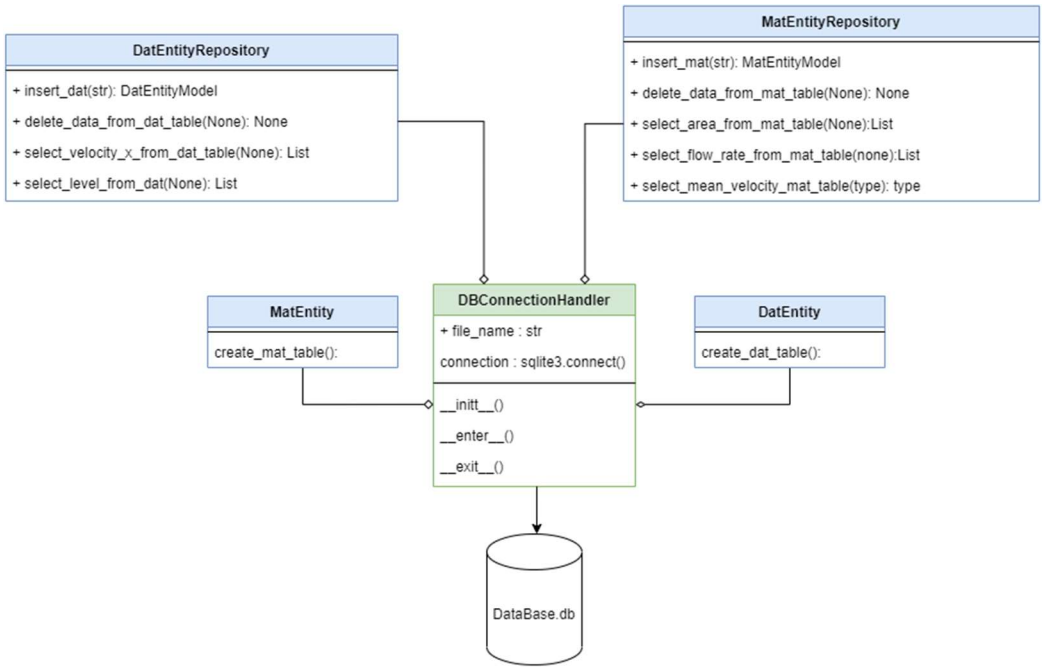
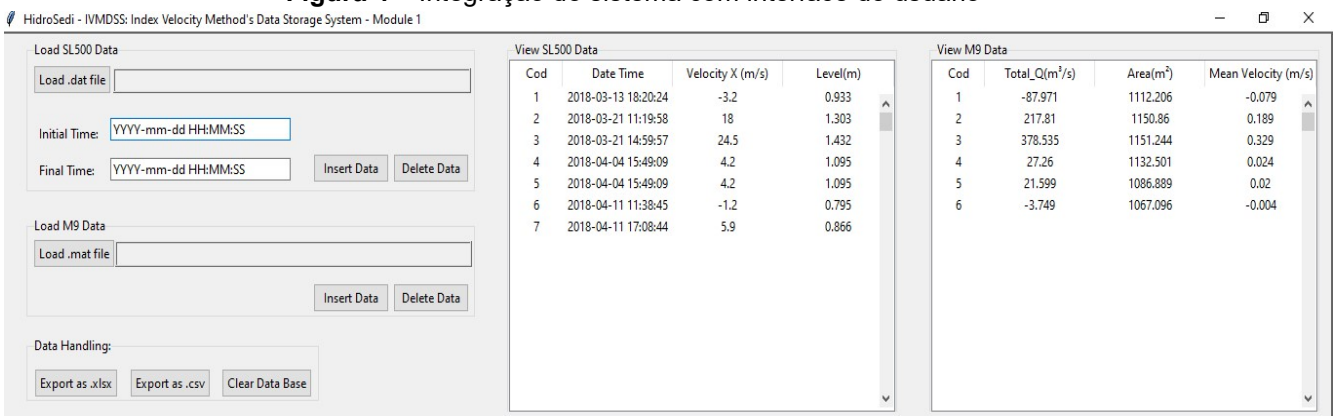


Figura 3 – Relação entre as Classes no sistema, a partir da metodologia gráfica Unified Modeling Language. Fonte: Autores.

Por fim, o sistema integrado com a implementação de interface do usuário, utilizando a biblioteca nativa do Python *tkinter-Tcl/Tk* (Python.org, 2022) é apresentado pela Figura 4.

Figura 4 – Integração do sistema com interface do usuário



Language. Fonte: Autores

4. CONCLUSÕES

A adoção de POO em conjunto com o padrão de código *Repository Pattern* e com o SGBD SQLite 3, se mostraram uma opção para a composição de um sistema para armazenamento de dados tratados. Funcionando de maneira independente, podendo ser reutilizado em diferentes tipos de projetos que envolvam a aplicação do método das velocidades indexadas.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Cristiano das Neves; SILANS, Alain Marie Bernard Passerat de; WENDLAND, Edson; ROEHRING, Jackson. Programação orientada a objetos para o desenvolvimento e integração de um modelo hidrológico distribuído chuva-vazão a um sistema de informações geográficas livre. **Revista Brasileira de Recursos Hídricos**, [S. L], v. 2, n. 13, p. 141-149, abr. 2008.

BREDA, João Paulo F. *et al*. **Guia prático de código-fonte**: rotinas do MGB. manual técnico. Porto Alegre: UFRGS, 2020. 50 p.

EVANS, Eric. **Domain Driven Design**: tackling complexity in the heart of software. [S. L]: Addison Wesley Professional, 2003.

FARRELL, Doug. **Data management with Python, SQLITE and SQLITE-Alchemy**. 2021. Disponível em: https://realpython.com/python-sqlite-sqlalchemy/. Acesso em: 21 jul. 2022.

GUEDES, Gilleanes T. A. **UML 2**: uma abordagem pratica. 3. ed. [S. L]: Novatec, 2018.

LOTT, Steven F.; PHILLIPS, Dusty. **Object-oriented programming**: build robust and maintainable object-oriented python applications and libraries. 4. ed. Birmingham, UK: Packet, 2021. 715 p.

NASCIMENTO, Jamilson do *et al*. Utilização do Python para processamento de dados obtidos com medidores acústicos de vazão: uma alternativa ao uso de softwares comerciais. In: CONGRESSO DE INICIAÇÃO CIENTÍFICA, 30., 2021, Pelotas. **Anais [...].** Pelotas: UFPel, 2022. p. 1-4.

TKINTER TCL-TK DOCUMENTATION. **SQLITE.ORG**, Disponível em: https://www.pyhon.org/3/libray/tkinter.html. Acesso: 10 de ago. 2022.

PYTHON.ORG. Python Software Foundation (org.). **Python Docs**. Disponível em: https://docs.python.org/3/. Acesso em: 14 ago. 2022.

SILVA, Antunes Dantas da *et al*. **Um estudo sobre diferentes linguagens de programação para a introdução da programação funcional**. 2019. 15 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal do Campina Grande, Campina Grande, 2019.

SQLITE DOCUMENTATION. **SQLITE.ORG**, Disponível em: https://www.sqlite.org/index.html. Acesso: 10 de ago. 2022.

ZAMBONI, Lincon C. *et al*. A programação orientada a objetos como ferramenta para o aprendizado e auxílio em projetos de engenharia. In: CONGRESSO BRASILEIRO DE ENSINO DE ENGENHARIA, 32., 2005, Campina Grande. **Anais [...].** [S. L]: Cobenge, 2006. p. 1-13.