Introdução ao projeto:

O trabalho realizado foi a criação de uma biblioteca por meio de C#, usando a aplicação Visual Studio Code, tendo que ser realizado algumas tarefas obrigatórias no mesmo, uma delas consistia em ser possível gerenciar os livros, tendo como cadastrá-los (contendo título, autor, ISBN, gênero do livro e a quantidade em estoque do mesmo.), deveria ter um controle em estoque de cada livro, uma forma para consultá-los por título, autor ou gênero e que fosse possível ter atualizações em relação aos livros, outra tarefa que o projeto deveria atender seria em relação ao gerenciamento de usuários, tendo como cadastrá-los por nome, número de identificação, endereço e contato, uma maneira para consultar os usuários já cadastrados e uma atualização de informações dos mesmos, a última tarefa requisitada consistia em criar um sistema de empréstimo dos livros, sendo registrado os empréstimos, e podendo ver histórico desses empréstimos filtrados por usuário, todas essas tarefas deveriam ser realizadas utilizando programação orientada a objetos, que seria essencial para a realização de tais tarefas, tornando o código mais organizado e eficiente, facilitando a manutenção e possível adição de novos recursos, os quatro pilares da POO (Programação orientada a objetos) são fundamentais para alcançar todos esses objetivos, sendo aplicados de forma integrada ao desenvolvimento deste sistema, sendo utilizados diversas vezes e com influência direta no funcionamento do software.

Os quatro pilares:

O Encapsulamento é o pilar que busca proteger o estado interno de um objeto, restringindo o acesso direto a suas propriedades e métodos. Isso é feito por meio do uso de modificadores de acesso, que determinam quais partes do código podem interagir com o objeto. O encapsulamento garante que a manipulação dos dados aconteça apenas por meio de métodos controlados, o que mantém uma maior segurança e integridade dos dados. Por expor apenas o necessário, o encapsulamento ajuda a reduzir a complexidade do sistema e evita efeitos colaterais indesejados.

O encapsulamento foi utilizado no sistema na aba de Usuário:

public class Usuario
public string Nome { get; private set; }

Sendo definidas como private set o nome, número de identificação, endereço, contato e histórico de empréstimo do usuário, ou seja, essas informações podem ser lidas fora da classe, mas só serão modificadas por meio de métodos específicos ou internamente.

A Herança é um mecanismo que permite que uma classe herde propriedades e métodos de outra classe, o pilar da herança realiza a reutilização de código e facilita a criação de hierarquias de classes. Através da herança, é possível criar classes mais específicas que compartilham comportamentos comuns definidos em uma classe base. Isso não só diminui a redundância no código, mas também permite que alterações feitas na classe base sejam refletidas nas classes derivadas, facilitando a manutenção e evolução do sistema. Um exemplo de onde a Herança foi utilizada no software pode ser visto na relação entre as classes ItemBiblioteca e Livro, onde ItemBiblioteca também herda de IEmprestavel e Livro de IPesquisavel:

public abstract class ItemBiblioteca : IEmprestavel
{ public string? Titulo { get; set;}
public string? Codigo { get; set;}
*continuação do código} – Classe ItemBiblioteca
public class Livro : ItemBiblioteca, IPesquisavel
{ *continuação do código } – Classe Livro

Neste caso, a classe livro herdou as propriedades “Titulo” e “Codigo” de ItemBiblioteca.

O Polimorfismo consiste na capacidade de um objeto assumir várias formas, permitindo que métodos em diferentes classes possam ser

chamados. Isso significa que um método pode ser definido em uma classe base e, em seguida, ser implementado de maneiras diferentes em classes derivadas. O polimorfismo proporciona flexibilidade ao código, permitindo que uma interface comum seja usada para tratar objetos de diferentes tipos. O polimorfismo é extremamente útil em sistemas complexos, onde diferentes tipos de objetos precisam interagir de forma coesa. No software o polimorfismo pode ser observado nos métodos Emprestar e Devolver, que estão presentes na classe ItemBiblioteca como abstratos e são implementados de forma específica na classe Livro:

Classe ItemBiblioteca–

```
public abstract void Emprestar(Usuario usuario);
public abstract void Devolver();
```

Classe Livro–

```
public override void Emprestar(Usuario usuario)
{ *continução do código }
public override void Devolver()
{ *continuação do código }
```

A Abstração é o pilar que permite simplificar o sistema, focando nas características essenciais de um objeto e ignorando detalhes desnecessários. Pela abstração, pode ser definido classes que representam conceitos do mundo real, concentrando-se nos aspectos que são relevantes para a aplicação. Facilitando a implementação de funcionalidades específicas, uma vez que a complexidade é gerida de maneira mais eficiente. A Abstração pode ser vista sendo utilizada na classe ItemBiblioteca:

Classe ItemBiblioteca–

```
public abstract class ItemBiblioteca
{ *continuação do código }
```

Já que a mesma é uma classe abstrata. Ela define os métodos e propriedades que os itens da biblioteca devem implementar, mas não é possível instanciá-las de forma direta.

Conclusão:

Com a realização do projeto, foi necessário muito tempo e muito estudo sobre a programação orientada a objetos, ensinando como a POO é utilizada e os motivos da mesma ser tão importante na hora de desenvolver o software, além de maior compreensão sobre o tema, o projeto também auxiliou no desenvolvimento de minhas habilidades como desenvolvedor, exigindo algo novo e diferente do habitua, a ideia da gravação para mostrarmos o código também influenciou positivamente, fazendo com que fosse preciso entender e ter autoridade no assunto e no próprio código, enfrentando junto disto a timidez, falando e explicando sobre nosso projeto e o que foi feito no mesmo, tornando tudo mais desafiador e divertido. No geral foi uma experiência prazerosa e muito desafiadora que trouxe juntamente dela muito conhecimento.

Referências bibliográficas utilizadas:

https://tasafo.wordpress.com/2015/03/09/a-importancia-da-programacao-orientada-a-objetos-parte-i/

https://meuartigo.brasilescola.uol.com.br/informatica/programacao-orientada-objetos.htm

https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264