Relatório de POO Projeto final

Aluno: Pedro Emanuel do Nascimento Lima

Do que se trata: O código é um sistema de gerenciamento de festivais usando POO. Ele tem classes para pessoas, clientes, funcionários, ingressos, palcos e festivais. Usa mixins para dar funções extras como gerar ID e registrar logs. A estrutura permite criar empresas de eventos, adicionar festivais, vender ingressos e auditar se a capacidade e equipe estão corretas.

Implementações: Implementei classes e objetos que servem para organizar e reutilizar os códigos de forma lógica. Usei o encapsulamento de forma correta, em algumas partes do código se encontram incorretas, por não saber como desenvolver aquela parte específica do código. Também usei heranças no código, além de abstração para criar modelos genéricos, deixando detalhes para subclasses.

Meu desenvolvimento: Eu tive dificuldades em entender como funciona a parte de abstração e polimorfismo, isso atrapalhou o meu entendimento do projeto, e que talvez mim atrasasse em relação aos meus colegas também. Por isso o meu código está incompleto em algumas partes específicas.

código: O código foi feito com base no diagrama mostrado no projeto, apesar de algumas dificuldades tentei manter com base no que estava no diagrama, todas as classes estão de forma correta ao que foi apresentado nos comentários que mim ajudaram bastante em algumas coisas pra fazer o código. Tive que pesquisar algumas coisas, por falta de conhecimento em algumas partes do projeto.

Aprendizado: Sinto que apesar de pouco, eu tive um melhor entendimento quando eu pesquisei sobre aquele assunto específico, e pode estudar a partir disso também.

Onde aparecem os 4 pilares: a abstração aparece na interface `Logavel`, que define um método mas não diz como ele deve funcionar. O encapsulamento está no `_id` do `IdentificávelMixin`, protegido e acessado só por método próprio. A herança está quando `Funcionario` e `Auditor` herdam características de outras classes para evitar repetição. O polimorfismo está no `logar_entrada`, que é o mesmo método em várias classes, mas com implementações diferentes.”