UNIVERSIDADE FEDERAL DE ALAGOAS

JOÃO VICTOR GOMES DOS SANTOS
LUCAS HERON SANTOS ANCHIETA

RELATÓRIO DO PROJETO WEPAYU

Maceió
2024
JOÃO VICTOR GOMES DOS SANTOS
LUCAS HERON SANTOS ANCHIETA

RELATÓRIO DO PROJETO WEPAYU

Relatório apresentado ao Curso Superior Bacharelado em Ciência da Computação,

Universidade Federal de Alagoas, Campus A. C. Simões, como requisito para avaliação e obtenção de nota na disciplina Programação 2, ministrada pelo Profº Dr. Mario Hozano Lucas de Souza.

Maceió

2024

SUMÁRIO

Link do repositório no github: LukeHer0/WePayU (github.com)

1. DESIGN ARQUITETURAL, COMPONENTES E SUAS INTERAÇÕES

O projeto consistiu em duas milestones, nas quais a primeira encarregou-se das user stories (conjunto de tarefas e métodos que o programa precisa lidar) 1 a 7 e a segunda da 8 a 10. Para realizar o projeto desenvolveu-se três tipos principais de classes: As que serviriam para a criação de objetos, as de gerenciamento e as de métodos úteis.

1.1 Classes de gerenciamento

As de gerenciamento foram utilizadas para armazenar os métodos referentes a empregados e sindicato e realizar o intermédio entre a facade e os objetos. Destas tem-se:

- GerenciaEmpregados: possui métodos que envolvem manipulações nos empregados da empresa, tais como por exemplo, criar, remover ou alterar atributos dos empregados.
- GerenciaSindicato: possui métodos que envolvem ações em empregados sindicalizados, como por exemplo, lançar e obter as taxas de serviço.
- GerenciaVendas: possui métodos de obtenção e lançamento de vendas de um empregado comissionado.
- GerenciaAgenda: Possui métodos para alterar ou criar uma agenda de pagamento para os empregados.
- FolhadePagamento: possui métodos relacionados ao pagamento do salário dos funcionários da empresa.
- Sistema: possui métodos essenciais e que são comuns em todas as user stories: encerrarSistema e zerarSistema, sendo essas responsáveis por zerar o sistema para que a milestone consiga armazenar os dados corretamente e sem interferência de outras user stories.

1.2 Classes de métodos úteis

As classes de métodos úteis como foram utilizadas para a criação de métodos recorrentes de utilidade ao longo do desenvolvimento do projeto. Por sua vez, as classes de criação de objetos foram feitas para a criação de suas instâncias. São elas

- Aritmética: apresenta métodos de verificações numéricas e contas matemáticas que se repetem no programa.
- Erros: carrega algumas verificações de erros que são recorrentes no decorrer do programa.
- Exceptions: foi feito um arquivo de exception para cada exception que possa acontecer no programa para melhorar a legibilidade do código.
- XMLUse: técnica usada para gerar arquivos .xml e assim manter a persistência dos dados requerida nos testes.

1.3 Classes de criação de objetos

Por fim, é válido citar as classes de criação de objetos, pois, como o próprio nome sugere, essas têm como objetivo criar objetos e consequentemente obter seus atributos, getters e setters. As estruturas utilizadas para armazenar os objetos dessas classes são hashmap (foi escolhida porque ela é capaz de armazenar uma determinada chave e um determinado valor da maneira desejada) e um arraylist (em situações nas quais não era necessário armazenar chave e valor e pela simplicidade da estrutura). Essas são:

- Empregado (Horista, Comissionado e Assalariado)
- Sindicato
- Cartões (Ponto e Venda)
- Agenda de pagamento

2. PADRÕES DE PROJETO

2.1. Builder

2.1.1 Descrição Geral:

O padrão builder é utilizado para a criação de objetos complexos de forma organizada, dividindo o processo da definição de seus atributos em diferentes métodos. Especialmente objetos com vários atributos.

2.1.2 Problema Resolvido:

O principal problema resolvido foi a criação de objetos que recebiam vários atributos em sua criação. Por exemplo, Empregados.

2.1.3 Identificação de Oportunidade:

Em determinado momento durante o desenvolvimento, percebeu-se que ao criar empregados, não se tinha controle de quais atributos seriam necessários, visto que a quantidade de atributos na criação de empregados variava em cada caso. Então decidiu-se aplicar o padrão de projeto para facilitar o entendimento deste processo.

2.1.4 Aplicação no Projeto:

O builder foi utilizado principalmente na classe empregados, para uma melhor visualização dos atributos definidos na criação.

2.2. Facade

2.2.1 Descrição Geral:

O facade é utilizado para a simplificação da interação com diversas funções complexas de um projeto a partir de uma interface com mais usabilidade e coesão.

2.2.2 Problema Resolvido:

O padrão de projeto permitiu criar uma classe concisa que continha as funções necessárias para o funcionamento do sistema, sem a poluição existente da chamada da função em outras classes.

2.2.3 Identificação de Oportunidade:

No início do projeto, notou-se que era necessário uma classe para centralizar os métodos que seriam enviados pelo script de testes para o sistema geral.

2.2.4 Aplicação no Projeto:

A classe Facade criada no projeto possui chamada para todas as funções necessárias para a interação do script com o sistema. Possibilitando a interação do script com o sistema.

2.3. Command

2.3.1 Descrição Geral:

O padrão command encapsula as solicitações de um objeto, possibilitando a passagem de solicitações mais complexas a outros objetos, implementar recursos de cancelamento de operações e preservar o objeto da solicitação.

2.3.2 Problema Resolvido:

Utilizamos o padrão command principalmente para interagir com os métodos do padrão memento sem comprometer o encapsulamento das demais classes que se relacionavam com este padrão.

2.3.3 Identificação de Oportunidade:

Durante o desenvolvimento do milestone 2, ao utilizar o memento, percebemos que seus métodos poderiam violar os princípios de OO ligados às demais classes. Para evitar isto, recorremos ao padrão command.

2.3.4 Aplicação no Projeto:

Todas as solicitações de undo, redo e backup passam pela classe MementoCommands, antes de de fato interagir com os objetos para os quais foram chamadas.

2.4. Memento

2.4.1 Descrição Geral:

O padrão de projeto memento tem como objetivo salvar e restaurar o estado anterior de um objeto a partir da criação de uma espécie de backup das classes em que o memento será utilizado.

2.4.2 Problema Resolvido:

A partir do padrão de projeto foi possível criar métodos como Undo e Redo, que são capazes de desfazer e refazer uma ação realizada pelo usuário.

2.4.3 Identificação de Oportunidade:

No decorrer da resolução do milestone 2, notou-se o uso dos comandos Undo e Redo, que têm como objetivo alterar o sistema a partir de ações realizadas

pelo usuário. Para solucionar esse problema, criou-se duas pilhas que armazenam as ações do usuário e essas são manipuladas a partir dos comandos supracitados.

2.4.4 Aplicação no Projeto:

Basicamente após qualquer ação feita pelo usuário, que de certa forma altera a lista de empregados, é feito um backup de memento do sistema, onde armazena-se numa pilha presente na classe MementoCommands. Quando o usuário solicita um undo, a ação sai da pilha do undo e vai para a pilha do redo para caso o usuário solicite um redo. O programa executa os comandos a partir da alternância dessas ações fornecidas a partir do padrão de projeto memento.