

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT

DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO – DSC

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Caio Vitor Brasileiro Torres – 116111245

Francivaldo Cabral de Andrade – 116111544

Luan Carlos da Silva Bezerra - 116110100

RELATÓRIO DO PROJETO DE LABORATÓRIO

**Projeto de Laboratório – Quem me Ajuda (QMA)**

**Resumo introdutório**

O desenvolvimento do sistema QMA tinha como objetivo central a independência das classes, logo de início, foi feito alguns esboços de diagramas para a primeira parte (US1, US2, US3), tentando encontrar o melhor design que solucionasse a problemática inicial e também pudesse ser expansível, pensando na manutenção e no crescimento do sistema, contudo, na estrutura do projeto adotamos uma fachada para abstrair/simplificar a complexidade das camadas mais internas do sistema, segunda parte a criação de controladores (controller) para o tratamento de dados de entrada e de saída do sistema para o usuário, para a lógica do sistema foi criado classes de serviço (services), que tinha como objetivo a lógica de negócio, onde busca as classes bases e outras classes de serviço para retornar o que é pedido pelo controlador, além da lógica eles também são responsáveis por guardar informações em estrutura de dados, e interligar as partes distintas do sistema, que conversam pelos seus respectivos services. Não vimos a necessidade inicial de criar mais uma camada no sistema para exercer o papel de banco de dados, portanto, cada serviço tem sua estrutura para o armazenamento das informações de acordo com a necessidade e particularidade do serviço, sendo assim, a estrutura do projeto é expansível, desacoplado, com fácil leitura e manutenção do código, ou seja, procuramos ao máximo respeitar o GRASP, e seus padrões.

**Classes Base**

Todas as classes bases tem seu próprio tratamento de erro lançando exceções, também são serializáveis para poder ser salva em um arquivo “.ser” e consequentemente poder ser carregada em um sistema, e todas tentam adotar o máximo possível do padrão expert e o baixo acoplamento, tentando ser o mais reutilizáveis possível.

A **classe Aluno** é a base do sistema, possui atribuições básicas como nome, matricula, código do curso, telefone e e-mail, onde telefone é opcional, existe a implementação de um comparable para que a ordem natural de ordenação fosse por nome e em caso de empate, pela matricula. Também foi implementado um serializable.

A **classe Tutor** é praticamente a classe base central do sistema, ela tem uma ligação indireta com a classe aluno, quem faz essa intermediação são os services, um tutor possui uma matricula, um e-mail, uma carteira, um id, um nível e é composto por disciplinas. Na criação de um tutor é exigido uma matricula, um email, uma disciplina e uma id, não se pode ser um tutor sem ter essas informações básicas, onde é possivel receber avaliações, receber doações, e obter informações dos atributos base.

A **classe** **Nivel** é responsável pelos níveis que o tutor pode ter de acordo com a avaliação recebida dos usuários, ela atualiza de acordo com cada avaliação recebida e retorna taxas, nível, e avaliação atual, a avaliação varia acima de zero e vai até 5. A classe é composta por uma interface Doação, onde cada nível implementa uma taxa de doação, onde a mesma é retornada por essa classe.

A **classe Disciplina** é responsável por guardar informações como nome da disciplina e nivel de proficiencia na disciplina.

A **classe Horário** guarda informações de horário e dia, dentro existe um database para guardar informações dos cadastros.

A **classe Local** guarda informações de locais e poderem ser cadastrados vários locais pois possui um database.

A **classe Ajuda** é abstrata, sendo possível a criação de ajudas concretas baseada na abstrata, foram criadas duas ajudas, a online e a presencial. A classe ajuda é o coração da classe base, ela precisa de informações de outras classes bases, na sua criação é nescessaria apenas um matricula de um aluno, mas consequentemte é nescessario setar um tutor para tal ajuda.

A **classe AjudaOnline** é basicamente a classe Ajuda porém concretizada, para o cenário que não exista a necessidade de um horário nem local para o atendimento.

A **classe AjudaPresencial,** ao contrário de AjudaOnline, recebe horário, dia e local de interesse, fora o básico da Ajuda abstrata já que é subclasse da mesma, contendo essas informações extras, sendo possível retornar esses atritubutos.

**Controladores**

Os controladores tem a responsábilidade do tratamento de entrada e de saida de dados, é a camada intermediaria entre a leitura de dados na camada acima e a saida de dados da logica na camada abaixo, sendo assim, foram criados controladores de acordo com a responsábilidade, por exemplo, o controlador caixa é um controlador que trata entradas que envolve a parte financeira do sistema, já o controlador de HorarioLocal, é responsável pelas solicitações de horário e locais, sendo assim é muito mais fácil de expandir, de localizar erros e de tratar exceções.

**Services**

Os services ficaram responsáveis pela lógica de négocio e por ter estrutura de dados com composições das classes base, recebendo solicitações dos controladores e retornando uma possível solução. E eles também ficaram com a responsabilidade da troca de informações entre as diversas áreas do sistema, abstraindo essa responsabilidade dos controladores e diminuindo o acoplamento e deixando as responsabilidades melhor definidas.

**Conclusão**

Por fim, com a chegada do US8, decidimos construir uma classe que instancia e cria os objetos, tendo em vista que seria mais fácil de salvar, carregar e limpar os arquivos ".ser", e tirar essa responsabilidade da Facade, por tanto, foi criado a classe FabricaSistema, que é responsável por isto. O diagrama parece bem complexo, mas o padrão que talvez seja complexo, porém qualquer desenvolvedor pode entender o fluxo do sistema. O sistema então fica concluído, podendo receber a qualquer momento novos US’s sem precisar ter alterações drásticas e lógicas complexas.