Grupo 01

Diogo Carvalho
Guilherme Pereira
Gonçalo Nogueira

Principais tópicos a abordar

TÓPICOS

- Resultados Obtidos
- Análise Interna da Equipa
- Processo de Engenharia Utilizado
- Qualidade Final do Produto

Resultados Obtidos

Proposta

Ao longo do semestre, como nos foi proposto, desenvolvemos um projeto onde criamos um sistema capaz de:

Gerir produtos vendidos por uma empresa, incluindo o armazenamento dos mesmos realizando AGV's

Gerir as encomendas de um determinado armazém, desde o momento em que a encomenda é feita até à chegada ao cliente

Criar formulários e questionários sobre um tópico desejado de forma fácil e integrada dentro na scope do projeto

Analise Interna

Pontos Fortes

Honestidade

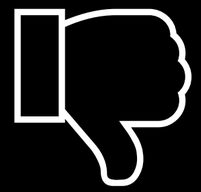
Vontade

Paciência

Entreajuda

Compreensão

Oportunidades



Fraquezas

Procrastinação

Alguma desorganização

Mau começo

Problemas

Falta de tempo

Desistência de um colega

Carga de trabalho

Conciliação pessoal e profissional

Processos de Engenharia Utilizados

Para o desenvolvimento do sistema seguimos um metodo de Desenvolvimento Ágil seguindo o padrão de design DDD

Desenvolvimento Agil

Durante o semestre fomos aplicando técnicas de metodologias ágeis que nos permitiram ter um contacto mais aproximado à experiência real do mundo de trabalho.

Como tal,

para cada iteração do projeto

Design DDD

A implementação do programa seguiu um Domain Driven Design

Apresentação

Aplicação

Base de Dados

Apresentação

```
private Menu buildProductOrderMenu() {
    final Menu menu = new Menu( title: "Product Order >");

    menu.addItem(REGISTER_PRODUCT_ORDER_OPTION, text: "Register a new Product Order", new RegisterProductOrderAction());
    menu.addItem(EXIT_OPTION, RETURN_LABEL, Actions.SUCCESS);
    return menu;
}
```

```
public class RegisterProductAction implements Action{

    @Override
    public boolean execute() { return new RegisterProductUI().show(); }
}
```

```
+= Base [ @salesClerk ] ==========================================================+

| 1. My account > | 2. Customers > | 3. Products > | 4. Product Order > | 0. Exit |
Please choose an option
```

Aplicação

RegisterProductUI.java

```
try {
    this.controller.registerNewProduct(internalCode,
} catch (Exception e) {
    System.out.println("----------------------------
    e.printStackTrace();
}
```

```
public class RegisterProductController {

    private final ProductRepository repository = PersistenceContext.repositories().productRepository();

    public void registerNewProduct(String internalCode, String productionCode, int barCode, String brandName, String reference, String productShortDescription,
                                   String productExtendedDescription, String productTechnicalDescription, List<Photo> photoPath, String productCategory,
                                   double priceNoTaxes, double priceTaxes, Location location) {
        final Product newProduct = new Product(internalCode, productionCode, barCode, brandName, reference, productShortDescription, productExtendedDescription,
                productTechnicalDescription, photoPath, productCategory, priceNoTaxes, priceTaxes, location);

        this.repository.save(newProduct);
    }
```

Persistência de Dados

RegisterProductController.java

```
ProductRepository repository = PersistenceContext.repositories().productRepository();
this.repository.save(newProduct);
```

PRODUCTTECHNICALDESCRIPTION	PRODUCTIONCODE	REFERENCE
Caixa de Lenços de Papel	A343FSD	343FD4GF
Lata de alumínio de coca-cola	3434FDF	654VSDFS3
Rato preto gaming	RTRW452	452FDF4DF

Value Objetcs

Os Objetos da nossa aplicação seguem a modelação do modelo de domínio da mesma, fazendo uso de Value Objets para uma maior riqueza do comportamento do objeto

```java
 * PaymentMethod - é a class value object que define o tipo de pagamento de uma encomenda
 */
@Embeddable
public class PaymentMethod implements ValueObject {

    /**
     * Tipo de pagamento
     */
    PaymentMethodEnum paymentMethod;

    /**
     * Construtor para PaymentMethod
     * @param paymentMethod metodo de pagamento da encomenda
     */
    public PaymentMethod(PaymentMethodEnum paymentMethod) {
        verify(paymentMethod);
        this.paymentMethod = paymentMethod;
    }

    public PaymentMethod() { }


    public boolean verify(PaymentMethodEnum paymentMethod){
        try{
            if(paymentMethod==null){
                throw new Exception("Payment method cannot be null");
            }
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }

        return false;
```

Qualidade Final do Produto

70%

Foram aplicados os padrões de Software propostos

Os Use Cases demonstrados ficaram funcionais

Não houve funcionalidades demonstradas que apresentem "bugs"

Houve Use Cases que ficaram por demonstrar ao cliente

Houve pouco cuidado com o deployment da aplicação fora do IDE

Aspetos a melhorar

Preparação prévia das tarefas

Podíamos ter feito uma melhor organização das tarefas a cumprir

Melhoramento do Deployment

Devíamos ter dado maior prioridade ao deployment da aplicação

Melhor primeira apresentação ao cliente

Devia ter havido uma maior preocupação com a primeira apresentação da aplicação ao cliente

0- very weak
10- fair
15- Very Good
5- weak
15- Good
15- Excellent

Auto avaliação

Auto avaliação antes e depois do projeto em relação às capacidades técnicas e comportamentais

0- very weak 10- fair 15- Very Good

5- weak 15- Good 15- Excellent

Auto avaliação

Auto avaliação antes e depois do projeto em relação às capacidades técnicas e comportamentais

Fim

Obrigado pelo vosso tempo!