Trabalho Final
INF01120 - TCP

Bruno Trindade
Fábio Gomes
Paulo Valcarenghi
Rafael Audibert

Índice

Visão geral

versão designer

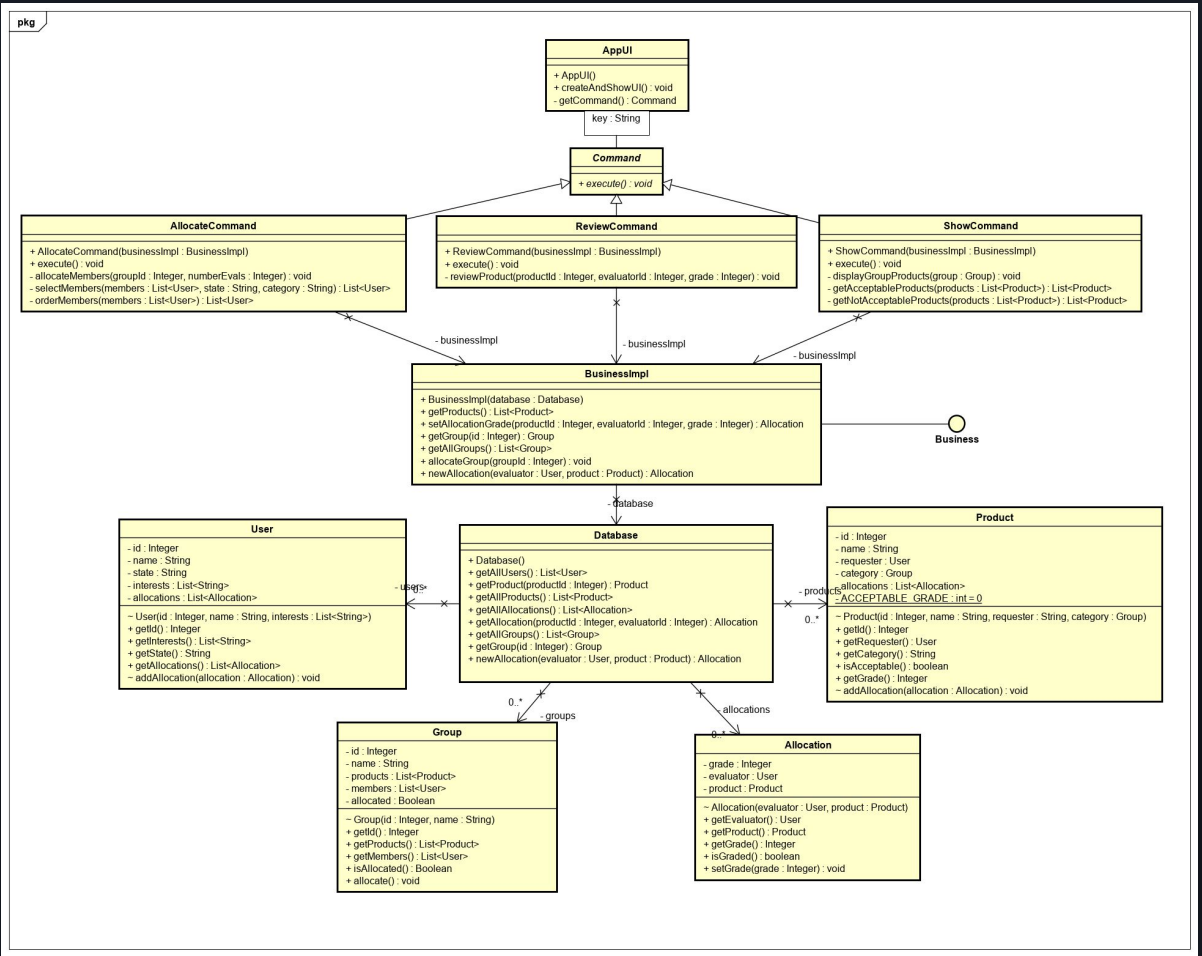


Diagrama de Classes

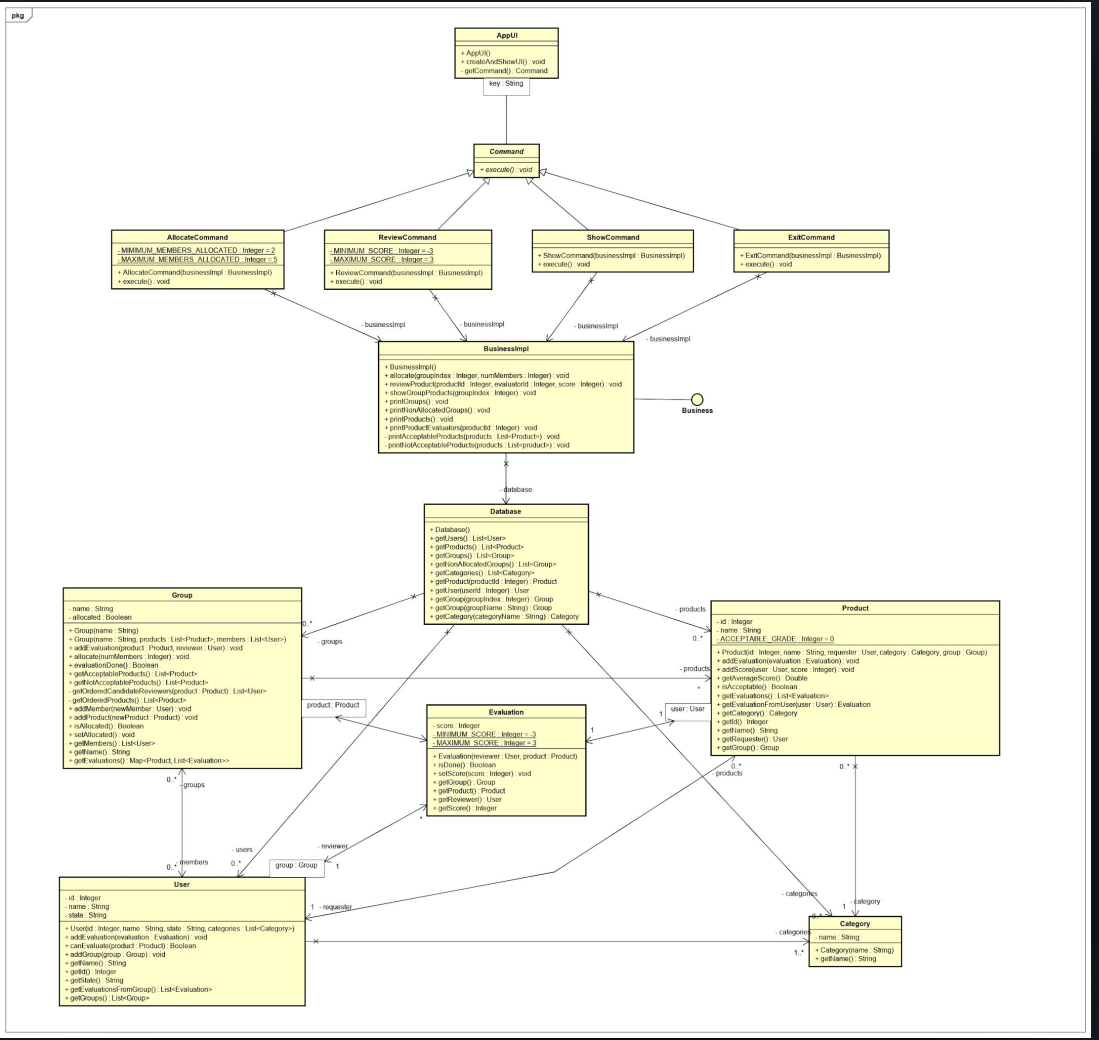


Diagrama de Classes

cosmetic.ui

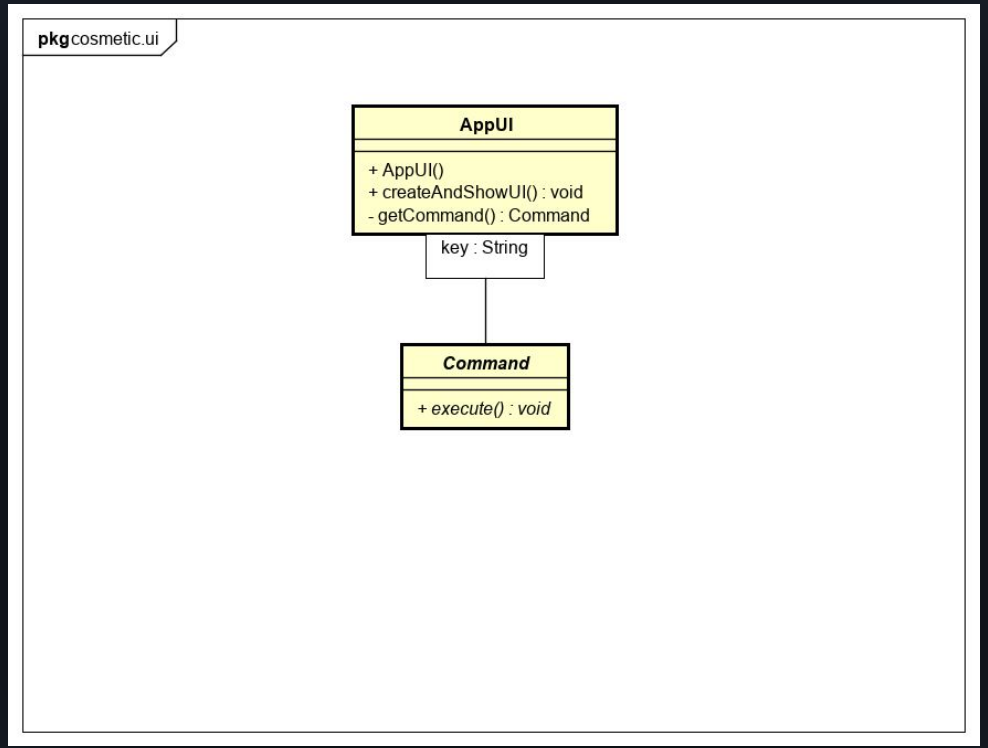


Diagrama de Classes

cosmetic.command

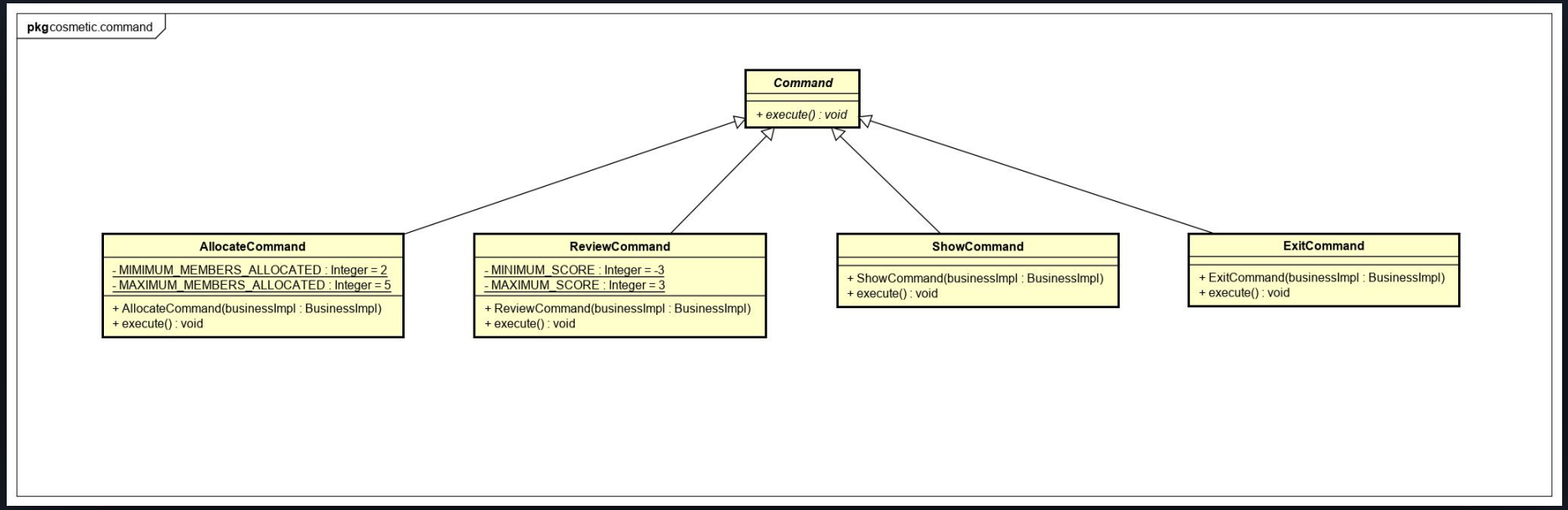


Diagrama de Classes

cosmetic.business

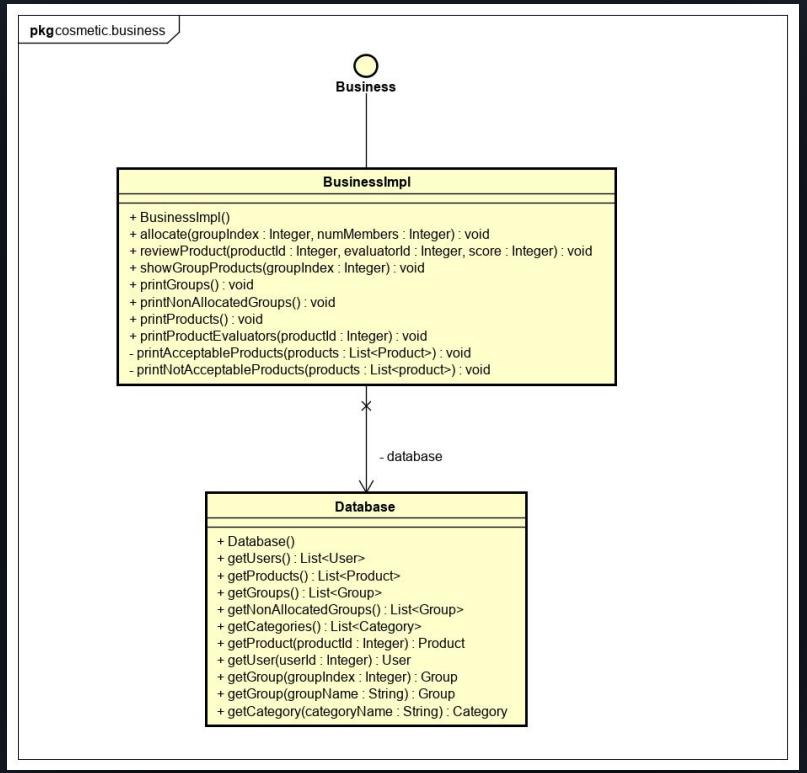


Diagrama de Classes

cosmetic.database

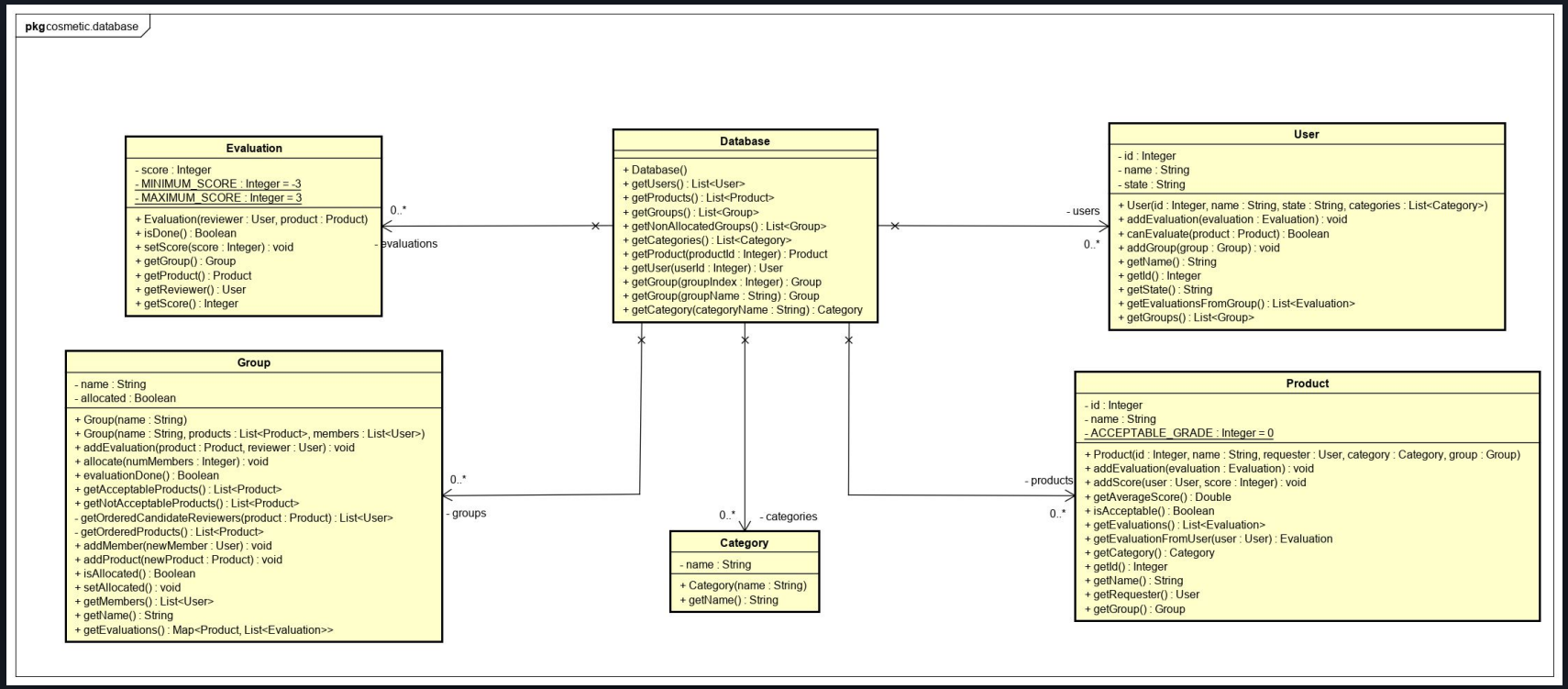


Diagrama de Sequência

cosmetic.command.allocate

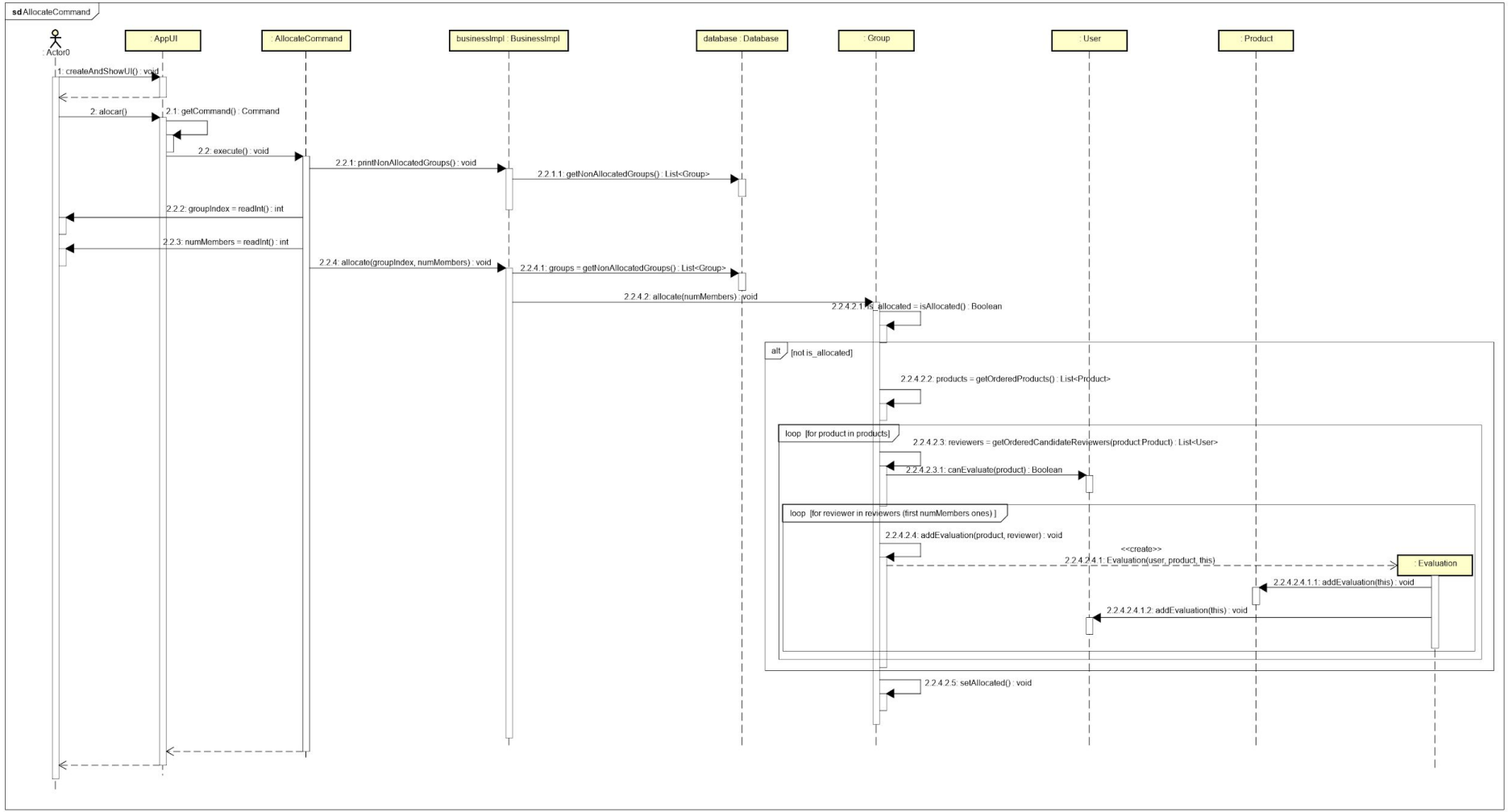

sd AllocateCommand
: Actor0
: AppUI
: AllocateCommand
businessImpl : BusinessImpl
database : Database
: Group
: User
: Product
1: createAndShowUI() : void
2: alocar()
2.1: getCommand() : Command
2.2: execute() : void
2.2.1: printNonAllocatedGroups() : void
2.2.1.1: getNonAllocatedGroups() : List<Group>
2.2.2: groupIndex = readInt() : int
2.2.3: numMembers = readInt() : int
2.2.4: allocate(groupIndex, numMembers) : void
2.2.4.1: groups = getNonAllocatedGroups() : List<Group>
2.2.4.2: allocate(numMembers) : void
2.2.4.2.1: is_allocated = isAllocated() : Boolean
alt
[not is_allocated]
2.2.4.2.2: products = getOrderedProducts() : List<Product>
loop [for product in products]
2.2.4.2.3: reviewers = getOrderedCandidateReviewers(product:Product) : List<User>
2.2.4.2.3.1: canEvaluate(product) : Boolean
loop [for reviewer in reviewers (first numMembers ones)]
2.2.4.2.4: addEvaluation(product, reviewer) : void
<<create>>
2.2.4.2.4.1: Evaluation(user, product, this)
: Evaluation
2.2.4.2.4.1.1: addEvaluation(this) : void
2.2.4.2.4.1.2: addEvaluation(this) : void
2.2.4.2.5: setAllocated() : void

Diagrama de Sequência

cosmetic.command.review

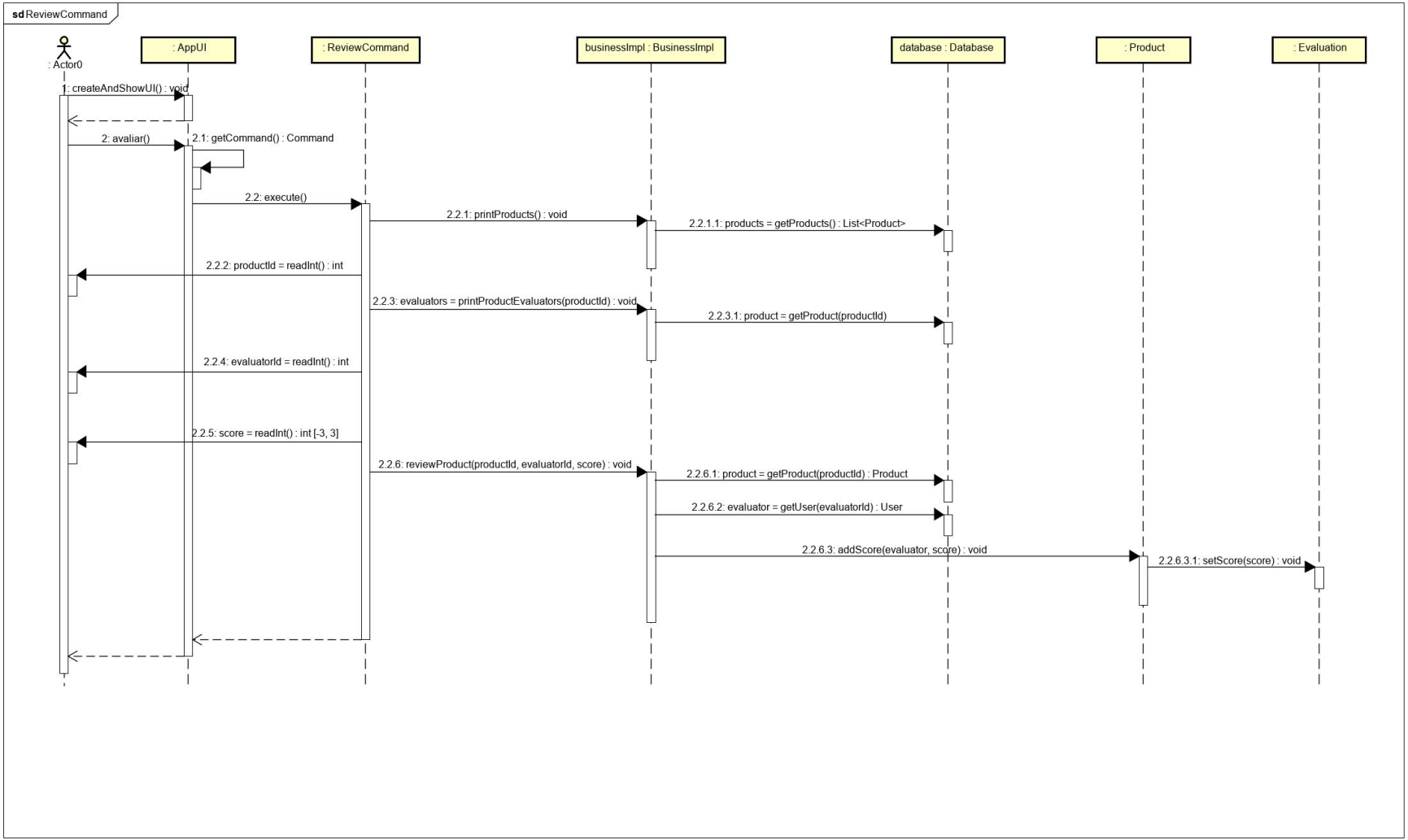
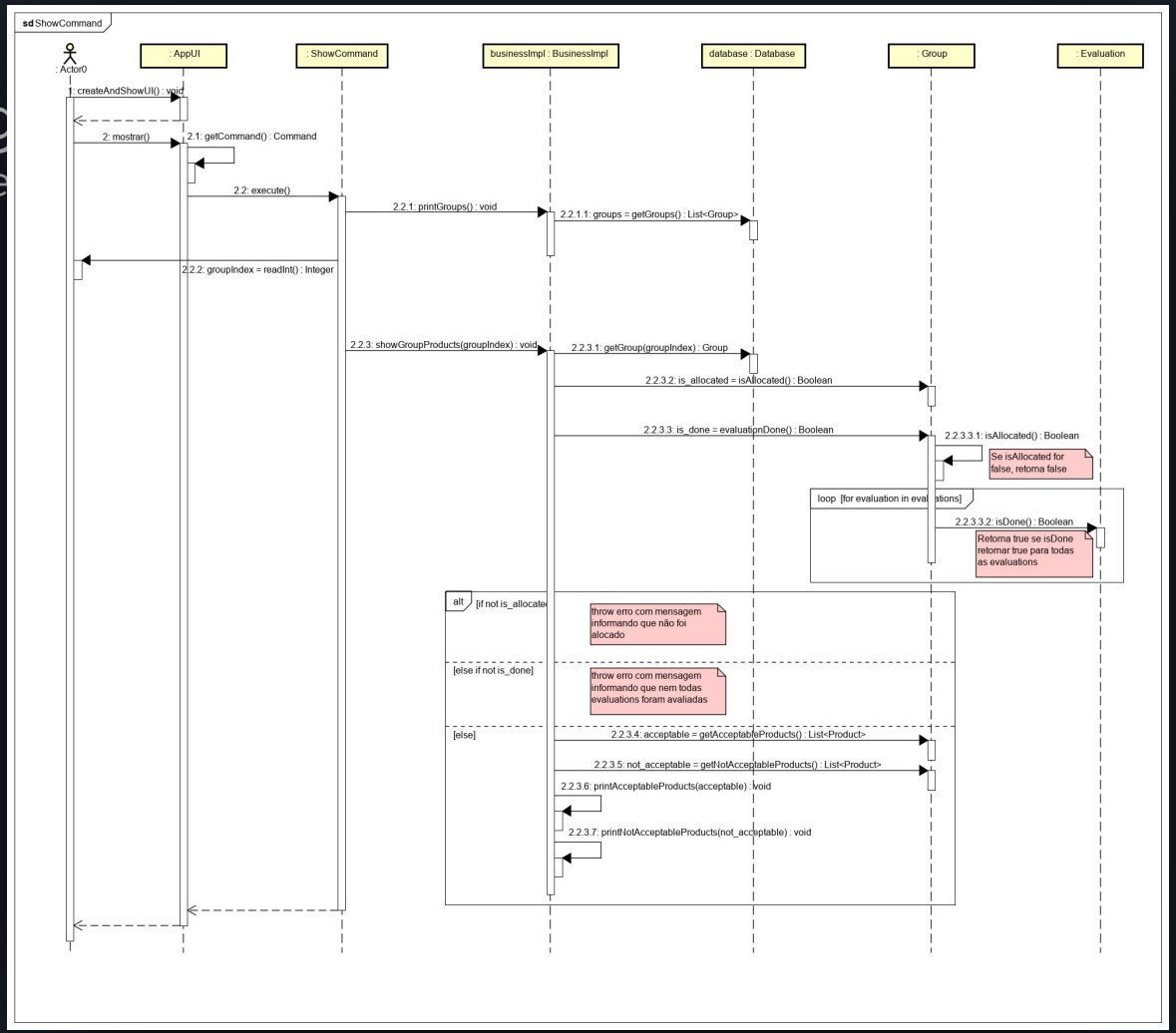

sd ReviewCommand
: Actor0
: AppUI
: ReviewCommand
businessImpl : BusinessImpl
database : Database
: Product
: Evaluation
1: createAndShowUI() : void
2: avaliar()
2.1: getCommand() : Command
2.2: execute()
2.2.1: printProducts() : void
2.2.1.1: products = getProducts() : List<Product>
2.2.2: productId = readInt() : int
2.2.3: evaluators = printProductEvaluators(productId) : void
2.2.3.1: product = getProduct(productId)
2.2.4: evaluatorId = readInt() : int
2.2.5: score = readInt() : int [-3, 3]
2.2.6: reviewProduct(productId, evaluatorId, score) : void
2.2.6.1: product = getProduct(productId) : Product
2.2.6.2: evaluator = getUser(evaluatorId) : User
2.2.6.3: addScore(evaluator, score) : void
2.2.6.3.1: setScore(score) : void

Diagrama de Sequência

cosmetic.command.show

Diag
cosme


sd ShowCommand
: Actor0
: AppUI
: ShowCommand
businessImpl : BusinessImpl
database : Database
: Group
: Evaluation
1: createAndShowUI() : void
2: mostrar()
2.1: getCommand() : Command
2.2: execute()
2.2.1: printGroups() : void
2.2.1.1: groups = getGroups() : List<Group>
2.2.2: groupIndex = readInt() : Integer
2.2.3: showGroupProducts(groupIndex) : void
2.2.3.1: getGroup(groupIndex) : Group
2.2.3.2: is_allocated = isAllocated() : Boolean
2.2.3.3: is_done = evaluationDone() : Boolean
2.2.3.3.1: isAllocated() : Boolean
Se isAllocated for false, retorna false
loop [for evaluation in eva ations]
2.2.3.3.2: isDone() : Boolean
Retorna true se isDone retornar true para todas as evaluations
alt [if not is_allocate
throw erro com mensagem informando que não foi alocado
[else if not is_done]
throw erro com mensagem informando que nem todas evaluations foram avaliadas
[else]
2.2.3.4: acceptable = getAcceptableProducts() : List<Product>
2.2.3.5: not_acceptable = getNotAcceptableProducts() : List<Product>
2.2.3.6: printAcceptableProducts(acceptable) : void
2.2.3.7: printNotAcceptableProducts(not_acceptable) : void

Código

estruturação do código-fonte

- Cosmetics [Trabalho Final master]
  - src
    - cosmetics
      - Cosmetics.java
    - cosmetics.business
      - Business.java
      - BusinessException.java
      - Category.java
      - Evaluation.java
      - Group.java
      - Product.java
      - User.java
    - cosmetics.business.database
      - Database.java
    - cosmetics.business.impl
      - BusinessImpl.java
    - cosmetics.test
      - DatabaseTest.java
      - EvaluationTest.java
      - GroupTest.java
      - ProductTest.java
      - UserTest.java
    - cosmetics.ui
      - AppUI.java
    - cosmetics.ui.command
      - AllocateCommand.java
      - Command.java
      - ExitCommand.java
      - ReviewCommand.java
      - ShowCommand.java

Código

custom exception

```java
package cosmetics.business;

public class BusinessException extends Exception {

    /**
     * Generated serialVersionUID
     */
    private static final long serialVersionUID = -5049174296539833065L;

    /**
     * Attribute to store the Exception arguments
     */
    private String[] args;

    /**
     * Create a new instance of BusinessException.
     */
    public BusinessException() {
        this.args = new String[0];
    }

    /**
     * Create a new instance of BusinessException.
     *
     * @param message the message to show.
     */
    public BusinessException(final String message) {
        super(message);
        this.args = new String[0];
    }

    /**
     * Create a new instance of BusinessException.
     *
     * @param message the message to show.
     * @param arg     the message argument
```

Código
overview

```java
package cosmetics.business;

public class Evaluation {

    private Integer score;
    private User reviewer;
    private Group group;
    private Product product;

    private static Integer MINIMUM_SCORE = -3;
    private static Integer MAXIMUM_SCORE = 3;

    public Evaluation(User reviewer, Product product) throws BusinessException {
        this.group = product.getGroup();
        this.product = product;
        this.reviewer = reviewer;
        this.score = null;

        product.addEvaluation(this);
        reviewer.addEvaluation(this);
    }

    public Boolean isDone() {
        if (this.getScore() == null) {
            return false;
        }

        return true;
    }

    public void setScore(Integer score) throws BusinessException {
        if (this.score != null) {
            throw new BusinessException("This evaluation has already been given an score");
        }
```

Código

interfaces funcionais + lambda

```
78      public List<Product> getAcceptableProducts() {
79          return products.parallelStream()
80                  .filter(product -> product.isAcceptable())
81                  .collect(Collectors.toCollection(ArrayList::new));
82      }
83
84      public List<Product> getNotAcceptableProducts() {
85          return products.parallelStream()
86                  .filter(product -> !product.isAcceptable())
87                  .collect(Collectors.toCollection(ArrayList::new));
88      }
89
90      private List<User> getOrderedCandidateReviewers(Product product) {
91          return members.parallelStream()
92                  .filter(user -> user.canEvaluate(product))
93                  .distinct()
94                  .sorted(Comparator.comparing((User user) -> user.getEvaluationsFromGroup(this).size())
95                          .thenComparing(User::getId))
96                  .collect(Collectors.toCollection(ArrayList::new));
97
98      }
99
100     private List<Product> getOrderedProducts() {
101         return products.parallelStream()
102                 .sorted(Comparator.comparing(Product::getId))
103                 .collect(Collectors.toCollection(ArrayList::new));
104     }
```

Código

Uso de toString()

```
@Override
public String toString() {
    return "Evaluation \t| User: " + getReviewer().getId() + "\t| Product: " + getProduct().getId() + "\t| Grade: " + (isDone() ?
}
```

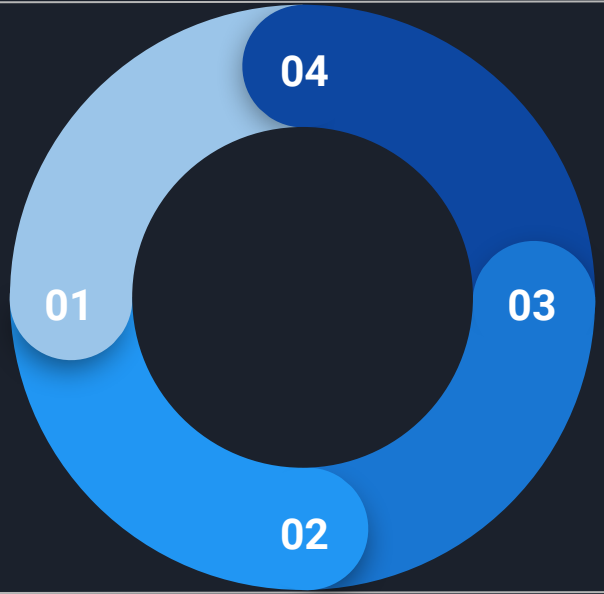
Testes Unitários

Criar/melhorar caso de teste

Tentando seguir as classes de equivalência, além de testes de valores limites

Implementação Trivial

Para forçar o teste passar



Implementação completa

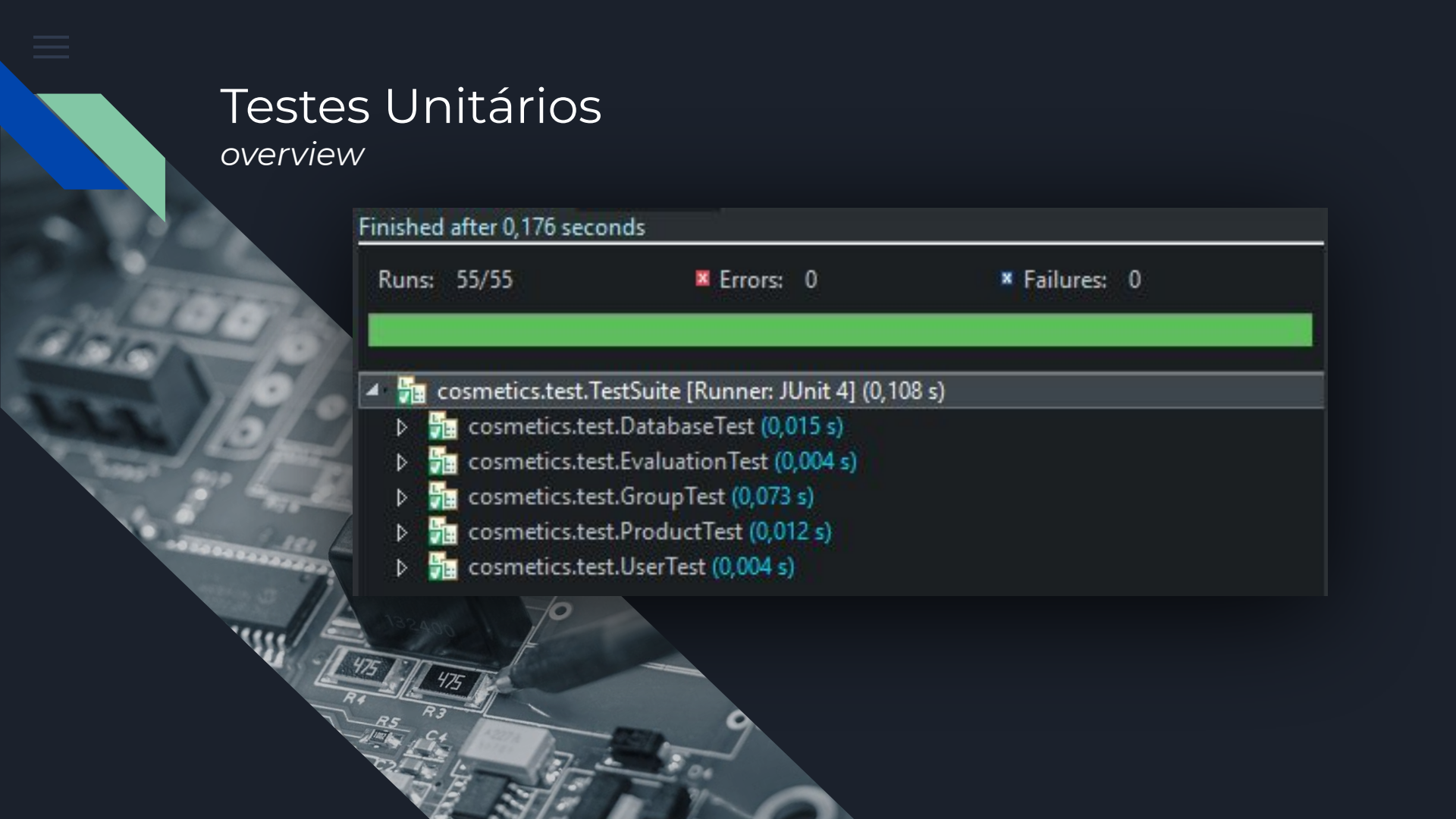
Buscar passar o caso de teste de maneira correta

Mais casos de teste

Para podermos detectar mais edge cases

Testes Unitários

overview



Testes Unitários

estatísticas

- **55** casos de teste
- **889** linhas de testes *(21.5%)*
- **0.15 ± (0.04)s** de runtime

Testes Unitários

suite

```java
package cosmetics.test;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ DatabaseTest.class,
        EvaluationTest.class,
        GroupTest.class,
        ProductTest.class,
        UserTest.class})

public class TestSuite {

}
```

Testes Unitários

sequenciais

```java
package cosmetics.test;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.Test;

import cosmetics.business.BusinessException;
import cosmetics.business.database.Database;

public class DatabaseTest {

    @Test
    public void testDatabase() throws BusinessException {
        Database database = new Database();

        // Check that all insertions happened correctly
        assertTrue(database.getCategories().size() == 6);
        assertTrue(database.getGroups().size() == 3);
        assertTrue(database.getUsers().size() == 10);
        assertTrue(database.getProducts().size() == 11);

        // Check that SPF B and SPF C are allocated and SPF A not
        assertFalse(database.getGroup("SPF A").isAllocated());
        assertTrue(database.getGroup("SPF B").isAllocated());
        assertTrue(database.getGroup("SPF C").isAllocated());
    }

}
```

Testes Unitários

paralelos

```java
// Test evaluationDone
@Test
public void evaluationDoneTestComplete() throws BusinessException {
    // Evaluations foram completas
    groupNEUsers.allocate(2);
    Map<Product, List<Evaluation>> evaluations = groupNEUsers.getEvaluations();
    evaluations.get(productNotCovered).get(0).setScore(3);
    assertTrue(groupNEUsers.evaluationDone());
}

@Test
public void evaluationDoneTestNotComplete() throws BusinessException {
    // Evaluations não foram completas
    groupNEUsers.allocate(2);
    assertFalse(groupNEUsers.evaluationDone());
}

@Test
public void evalluationDoneTestPartiallyComplete() throws BusinessException {
    spfA.allocate(2);
    Map<Product, List<Evaluation>> evaluations = spfA.getEvaluations();
    evaluations.get(laRocheCCCream).get(0).setScore(1);
    assertFalse(spfA.evaluationDone());
}

@Test
public void evaluationDoneTestNoEvaluation() throws BusinessException {
    // Não há evaluations
    assertFalse(groupNEUsers.evaluationDone());
}
```

Testes Unitários

expected error

```java
// Test addEvaluation
@Test(expected = BusinessException.class)
public void addEvaluationTestNull() throws BusinessException {
    userJoao.addEvaluation(null);
}

@Test(expected = BusinessException.class)
public void addEvaluationTestWrongUser() throws BusinessException {
    evalMateusCreamProduct = new Evaluation(userMateus, creamProduct);
    userJoao.addEvaluation(evalMateusCreamProduct);
}

@Test(expected = BusinessException.class)
public void addEvaluationTestIncompatibleProduct() throws BusinessException {
    evalJoaoShampooProduct = new Evaluation(userJoao, shampooProduct);
    userJoao.addEvaluation(evalJoaoShampooProduct);
}

@Test(expected = BusinessException.class)
public void addEvaluationTestNotExistingGroup() throws BusinessException {
    evalMateusCreamProduct = new Evaluation(userMateus, creamProduct);
    userMateus.addEvaluation(evalMateusCreamProduct);
}
```

DEMONSTRAÇÃO

Obrigado!

Técnicas de Construção de Programas
Prof. Ingrid Nunes

2019/1