

Escola Superior De Tecnologia e Gestão de Felgueiras

Licenciatura em Engenharia Informática

Engenharia de Software II

Repositório Git:  
<https://github.com/MiguelFreitas28/Projetofinal.git>

Miguel Freitas 8140059  
Bruno Cunha 8150395

­

Índice

[Introdução 3](#_Toc530421640)

[Conceitos fundamentais 3](#_Toc530421641)

[Equivalence Class Partitioning (ECP) 3](#_Toc530421642)

[Boundary Value Analysis (BVA) 3](#_Toc530421643)

[Plano de testes 4](#_Toc530421644)

[RENT 4](#_Toc530421645)

[GET BIKE 4](#_Toc530421646)

[RETURN BIKE 5](#_Toc530421647)

[RENTAL FEE 7](#_Toc530421648)

[CRÉDITO 8](#_Toc530421649)

[VERIFICAR CRÉDITO 8](#_Toc530421650)

[REGISTAR 9](#_Toc530421651)

[Resultados esperados vs Resultados obtidos 10](#_Toc530421652)

[Conclusão 11](#_Toc530421653)

# Introdução

O seguinte relatório baseia-se na representação de um projeto da cadeira de engenharia de software II do curso de Licenciatura em Engenharia Informática.

Foi-nos fornecido uma biblioteca chamada “BikeRentalSystem” com todos os requisitos do problema já implementados. O projeto consiste num sistema público de aluguer de bicicletas composto por vários depósitos espalhados pela cidade com 12 bicicletas em cada depósito.

Para se efetuar um aluguer, será preciso um registo do utilizador com o fornecimento de vários dados como por exemplo o nome e a informação do cartão de crédito.

## Conceitos fundamentais

### Equivalence Class Partitioning (ECP)

* Técnica destinada a reduzir o número de testes necessários;
* Técnica que divide o domínio de entrada (ou saída) em classe de dados em que os casos de teste podem ser derivados;

### Boundary Value Analysis (BVA)

* Técnica focada nos limites do domínio de entrada (ou saída) e imediatamente acima e abaixo (além de ou em vez de valores intermédios);
* Testa também valores especiais (null, 0, etc.);

# Plano de testes

## RENT

### GET BIKE

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t1)** | **Classe Inválida (t2)** |
| GetBicycle (Existência de Depósito) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor específico** | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (Não exista no array de Depósitos); |
| **Output Esperado:** | Retorna identificador da bicicleta | Retorna -1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t3)** | **Classe Inválida (t4)** |
| T2 - GetBicycle ( Existência do User ) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); | IDUser = NULL StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); |
| **Output Esperado:** | Retorna identificador da bicicleta | "Retorna Exceção" |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t5)** | **Classe Inválida(t6)** |
| GetBicycle (Existência de Crédito) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); GetCredit >= 1 | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); GetCredit < 1 |
| **Output Esperado:** | Retorna identificador da bicicleta | Retorna -1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t7)** | **Classe Inválida (t8)** |
| GetBicycle (Não existirem bicicletas disponíveis ou o utilizador tenha um aluguer ativo) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); GetCredit >= 1 isInUse = False || getBike!=NULL | IDUser > 0 (existente no array de Users), StartTime>=0 , ID Deposit>0 (existente no array de Depósitos); GetCredit >= 1 isInUse = True || getBike=NULL |
| **Output Esperado:** | Retorna identificador da bicicleta | Retorna -1 |

### RETURN BIKE

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t9)** | **Classe Inválida (t10)** |
| ReturnBicycle (Existência ID USER) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | idUser > 0 (existente no array de utilizadores); idDeposit > 0 (existente no array de depósitos); endtime >= 0; inUse =True | IDUser>0 (Não existente no array de Users) ID Deposit>0 (Existente no array de depósitos) Endtime >= 0; inUse =True |
| **Output Esperado:** | Retorna Pagamento e Saldo atual | retorna -1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t11)** | **Classe Inválida (t12)** |
| ReturnBicycle (Existência ID Depósito) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | idUser > 0 (existente no array de utilizadores); idDeposit > 0 (existente no array de depósitos); endtime >= 0; inUse =True | IDUser>0 (existente no array de Users) ID Deposit>0 (Não Existente no array de depósitos) Endtime >= 0; inUse =True |
| **Output Esperado:** | Retorna Pagamento e Saldo atual | retorna -1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t13)** | **Classe Inválida (t14)** |
| ReturnBicycle (Bicicleta está associada a um aluguer ativo) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | idUser > 0 (existente no array de utilizadores); idDeposit > 0 (existente no array de depósitos); endtime >= 0; inUse =True | idUser > 0 (existente no array de utilizadores); idDeposit > 0 (existente no array de depósitos); endtime >= 0; inUse =False |
| **Output Esperado:** | Retorna Pagamento e Saldo atual | retorna -1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t15)** | **Classe Inválida (t16)** |
| ReturnBicycle (lugares de entrega não existem) | **nºde inputs** | 3 | 3 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | idUser > 0 (existente no array de utilizadores); idDeposit > 0 (existente no array de depósitos); endtime >= 0; isInUse = true (existente num deposit no array de locks) | idUser > 0 (existente no array de utilizadores); idDeposit > 0 (existente no array de depósitos); endtime >= 0; isInUse = FALSE (existente num deposit no array de locks) |
| **Output Esperado:** | Retorna Pagamento e Saldo atual | Retorna -1 |

### RENTAL FEE

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t17)** | **Classe Inválida (t18)** |
| Rental Program = 1 | **nºde inputs** | 4 | 4 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | RentalProgram=1, StartTime>0, EndTime>0, nRentals>0; | RentalProgram!=1, StartTime>0, EndTime>0, nRentals>0; |
| **Output Esperado:** | retorna ( endtime - startime) \* rentalFee | retorna 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t19)** | **Classe Inválida (t20)** |
| Rental Program = 2 && resto da divisão inteira do nRentals por 10 é !=0 | **nºde inputs** | 4 | 4 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | RentalProgram=2, StartTime>0, EndTime>0, nRentals>0; | RentalProgram=2, StartTime>0, EndTime>0, nRentals=0; |
| **Output Esperado:** | retorna rentalFee \* ( endtime - startime) | retorna 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t21)** | **Classe Inválida (t22)** |
| Se ( endtime - startime) <= 10 | **nºde inputs** | 4 | 4 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | RentalProgram=2, StartTime>0, EndTime>0, nRentals>0; endtime - startime <= 10; | RentalProgram=2, StartTime>0, EndTime>0, nRentals>0; endtime - startime >10; |
| **Output Esperado:** | retorna rentalFee \* ( endtime - startime) | retorna 10 \*rentalFee + (( endtime - startime)- 10 )\* rentalFee |

### CRÉDITO

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t23)** | **Classe Inválida (t24)** |
| Adicionar Crédito | **nºde inputs** | 2 | 2 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | IDUser>0 (Existente no array de users); Amount > 0 GetCredit + Amount > 0 | IDUser>0 (Não existente no array de users); Amount > 0 GetCredit + Amount > 1 |
| **Output Esperado:** | Crédito de (amount) adicionado ao UserID | Cliente não existe |

### VERIFICAR CRÉDITO

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t25)** | **Classe Inválida (t26)** |
| Existir User | **nºde inputs** | 1 | 1 |
| **tipo de inputs** | Int | Int |
| **valor especifico** | idUser >= 0 (existente no array de utilizadores); getCredit >= 1 | idUser >= 0 (não existente no array de utilizadores); getCredit >= 1 |
| **Output Esperado:** | retorna verdadeiro | retorna falso |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t27)** | **Classe Inválida (t28)** |
| Existir Crédito > 1 | **nºde inputs** | 1 | 1 |
| **tipo de inputs** | int | Int |
| **valor especifico** | idUser >= 0 (existente no array de utilizadores); getCredit >= 1 | idUser >= 0 (existente no array de utilizadores); getCredit <1 |
| **Output Esperado:** | retorna verdadeiro | retorna falso |

### REGISTAR

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Critério** | **Classe Válida (t29)** |
| Criar Utilizador | **nºde inputs** | 3 |
| **tipo de inputs** | Int,string |
| **valor especifico** | IDUser>0 (Não existente no array de Users), String válida Rental Program=1 || Rental Program=2 ; |
| **Output Esperado:** | "Utilizador Criado" |
|  |  |  |
| **Test Case** | **Critério** | **Classe Inválida (t30)** |
| Utilizador já existe | **nºde inputs** | 3 |
| **tipo de inputs** | Int,string |
| **valor especifico** | IDUser>0 (Existente no array de Users), String válida Rental Program=1 || Rental Program=2 ; |
| **Output Esperado:** | "Utilizador já existe" |

# Resultados esperados vs Resultados obtidos

|  |  |  |
| --- | --- | --- |
| Teste | Resultado esperado | Resultado Obtido |
| 1 | Passar | “Passou” |
| 2 | Passar | “Passou” |
| 3 | Passar | “Passou” |
| 4 | Passar | “Passou” |
| 5 | Passar | “Passou” |
| 6 | Passar | “Passou” |
| 7 | Passar | “Passou” |
| 8 | Passar | “Falhou” |
| 9 | Passar | “Passou” |
| 10 | Passar | “Passou” |
| 11 | Passar | “Passou” |
| 12 | Passar | “Passou” |
| 13 | Passar | “Passou” |
| 14 | Passar | “Passou” |
| 15 | Passar | “Passou” |
| 16 | Passar | “Passou” |
| 17 | Passar | “Passou” |
| 18 | Passar | “Passou” |
| 19 | Passar | “Passou” |
| 20 | Passar | “Passou” |
| 21 | Passar | “Passou” |
| 22 | Passar | “Falhou” |
| 23 | Passar | “Passou” |
| 24 | Passar | “Falhou” |
| 25 | Passar | “Passou” |
| 26 | Passar | “Passou” |
| 27 | Passar | “Falhou” |
| 28 | Passar | “Falhou” |
| 29 | Passar | “Passou” |
| 30 | Passar | “Passou” |

# Conclusão

Com a concretização e elaboração deste projeto, concluímos que os testes de software são importantes para quaisquer tipos de projetos e planeamentos de projetos.

Cumprimos com todos os objetivos que nos propuseram, ainda assim sentimos algumas dificuldades no desenvolvimento do código para alguns dos testes e respetivos métodos.

Concluímos assim o nosso projeto com maiores competências e um melhor conhecimento e aprofundamento acerca da matéria abordada. Ajudou-nos ainda a aperfeiçoar as nossas competências de investigação, organização e trabalho em equipa.