Trab TP1

Generated by Doxygen 1.9.3

1 TP1-Projeto-1 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 Avaliacao Class Reference . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Member Function Documentation . . . 7

4.1.2.1 getCodigo() . . . 8

4.1.2.2 getDescricao() . . . 8

4.1.2.3 getNota() . . . 8

4.1.2.4 setCodigo() . . . 8

4.1.2.5 setDescricao() . . . 9

4.1.2.6 setNota() . . . 9

4.2 Cidade Class Reference . . . 9

4.2.1 Detailed Description . . . 10

4.2.2 Member Function Documentation . . . 10

4.2.2.1 getValor() . . . 10

4.2.2.2 setValor() . . . 10

4.3 Codigo Class Reference . . . 11

4.3.1 Detailed Description . . . 11

4.3.2 Member Function Documentation . . . 11

4.3.2.1 getValor() . . . 11

4.3.2.2 setValor() . . . 11

4.4 Data Class Reference . . . 12

4.4.1 Detailed Description . . . 12

4.4.2 Member Function Documentation . . . 12

4.4.2.1 getValor() . . . 13

4.4.2.2 setValor() . . . 13

4.5 Descricao Class Reference . . . 13

4.5.1 Detailed Description . . . 14

4.5.2 Member Function Documentation . . . 14

4.5.2.1 getValor() . . . 14

4.5.2.2 setValor() . . . 14

4.6 Duracao Class Reference . . . 15

4.6.1 Detailed Description . . . 15

4.6.2 Member Function Documentation . . . 15

4.6.2.1 getValor() . . . 15

4.6.2.2 setValor() . . . 15

4.7 Email Class Reference . . . 16
4.7.1 Detailed Description . . . 16
4.7.2 Member Function Documentation . . . 16
4.7.2.1 getValor() . . . 17
4.7.2.2 setValor() . . . 17
4.8 Endereco Class Reference . . . 17
4.8.1 Detailed Description . . . 18
4.8.2 Member Function Documentation . . . 18
4.8.2.1 getValor() . . . 18
4.8.2.2 setValor() . . . 18
4.9 Excursao Class Reference . . . 19
4.9.1 Detailed Description . . . 19
4.9.2 Member Function Documentation . . . 19
4.9.2.1 getCidade() . . . 20
4.9.2.2 getCodigo() . . . 20
4.9.2.3 getDescricao() . . . 20
4.9.2.4 getDuracao() . . . 20
4.9.2.5 getEndereco() . . . 21
4.9.2.6 getNota() . . . 21
4.9.2.7 getTitulo() . . . 21
4.9.2.8 setCidade() . . . 21
4.9.2.9 setCodigo() . . . 22
4.9.2.10 setDescricao() . . . 22
4.9.2.11 setDuracao() . . . 22
4.9.2.12 setEndereco() . . . 23
4.9.2.13 setNota() . . . 23
4.9.2.14 setTitulo() . . . 23
4.10 Horario Class Reference . . . 24
4.10.1 Detailed Description . . . 24
4.10.2 Member Function Documentation . . . 24
4.10.2.1 getValor() . . . 24
4.10.2.2 setValor() . . . 24
4.11 Idioma Class Reference . . . 25
4.11.1 Detailed Description . . . 25
4.11.2 Member Function Documentation . . . 25
4.11.2.1 getValor() . . . 25
4.11.2.2 setValor() . . . 25
4.12 Nome Class Reference . . . 26
4.12.1 Detailed Description . . . 26
4.12.2 Member Function Documentation . . . 26
4.12.2.1 getValor() . . . 27
4.12.2.2 setValor() . . . 27

4.13 Nota Class Reference . . . 27
4.13.1 Detailed Description . . . 28
4.13.2 Member Function Documentation . . . 28
4.13.2.1 getValor() . . . 28
4.13.2.2 setValor() . . . 28
4.14 Senha Class Reference . . . 29
4.14.1 Detailed Description . . . 29
4.14.2 Member Function Documentation . . . 29
4.14.2.1 getValor() . . . 29
4.14.2.2 setValor() . . . 29
4.15 Sessao Class Reference . . . 30
4.15.1 Detailed Description . . . 30
4.15.2 Member Function Documentation . . . 30
4.15.2.1 getCodigo() . . . 31
4.15.2.2 getData() . . . 31
4.15.2.3 getHorario() . . . 31
4.15.2.4 getIdioma() . . . 31
4.15.2.5 setCodigo() . . . 31
4.15.2.6 setData() . . . 32
4.15.2.7 setHorario() . . . 32
4.15.2.8 setIdioma() . . . 32
4.16 Titulo Class Reference . . . 33
4.16.1 Detailed Description . . . 33
4.16.2 Member Function Documentation . . . 33
4.16.2.1 getValor() . . . 33
4.16.2.2 setValor() . . . 33
4.17 TUAvaliacao Class Reference . . . 34
4.17.1 Detailed Description . . . 34
4.17.2 Member Function Documentation . . . 34
4.17.2.1 run() . . . 34
4.17.3 Member Data Documentation . . . 34
4.17.3.1 FALHA . . . 35
4.17.3.2 SUCESSO . . . 35
4.18 TUCidade Class Reference . . . 35
4.18.1 Detailed Description . . . 35
4.18.2 Member Function Documentation . . . 35
4.18.2.1 run() . . . 35
4.18.3 Member Data Documentation . . . 36
4.18.3.1 FALHA . . . 36
4.18.3.2 SUCESSO . . . 36
4.19 TUCodigo Class Reference . . . 36
4.19.1 Detailed Description . . . 36

4.19.2 Member Function Documentation . . . 36
4.19.2.1 run() . . . 37
4.19.3 Member Data Documentation . . . 37
4.19.3.1 FALHA . . . 37
4.19.3.2 SUCESSO . . . 37
4.20 TUData Class Reference . . . 37
4.20.1 Detailed Description . . . 37
4.20.2 Member Function Documentation . . . 38
4.20.2.1 run() . . . 38
4.20.3 Member Data Documentation . . . 38
4.20.3.1 FALHA . . . 38
4.20.3.2 SUCESSO . . . 38
4.21 TUDescricao Class Reference . . . 38
4.21.1 Detailed Description . . . 39
4.21.2 Member Function Documentation . . . 39
4.21.2.1 run() . . . 39
4.21.3 Member Data Documentation . . . 39
4.21.3.1 FALHA . . . 39
4.21.3.2 SUCESSO . . . 39
4.22 TUDuracao Class Reference . . . 39
4.22.1 Detailed Description . . . 40
4.22.2 Member Function Documentation . . . 40
4.22.2.1 run() . . . 40
4.22.3 Member Data Documentation . . . 40
4.22.3.1 FALHA . . . 40
4.22.3.2 SUCESSO . . . 40
4.23 TUEmail Class Reference . . . 41
4.23.1 Detailed Description . . . 41
4.23.2 Member Function Documentation . . . 41
4.23.2.1 run() . . . 41
4.23.3 Member Data Documentation . . . 41
4.23.3.1 FALHA . . . 41
4.23.3.2 SUCESSO . . . 41
4.24 TUEndereco Class Reference . . . 42
4.24.1 Detailed Description . . . 42
4.24.2 Member Function Documentation . . . 42
4.24.2.1 run() . . . 42
4.24.3 Member Data Documentation . . . 42
4.24.3.1 FALHA . . . 42
4.24.3.2 SUCESSO . . . 42
4.25 TUExcursao Class Reference . . . 43
4.25.1 Detailed Description . . . 43

4.25.2 Member Function Documentation . . . 43
4.25.2.1 run() . . . 43
4.25.3 Member Data Documentation . . . 43
4.25.3.1 FALHA . . . 43
4.25.3.2 SUCESSO . . . 43
4.26 TUHorario Class Reference . . . 44
4.26.1 Detailed Description . . . 44
4.26.2 Member Function Documentation . . . 44
4.26.2.1 run() . . . 44
4.26.3 Member Data Documentation . . . 44
4.26.3.1 FALHA . . . 44
4.26.3.2 SUCESSO . . . 44
4.27 TUIdioma Class Reference . . . 45
4.27.1 Detailed Description . . . 45
4.27.2 Member Function Documentation . . . 45
4.27.2.1 run() . . . 45
4.27.3 Member Data Documentation . . . 45
4.27.3.1 FALHA . . . 45
4.27.3.2 SUCESSO . . . 45
4.28 TUNome Class Reference . . . 46
4.28.1 Detailed Description . . . 46
4.28.2 Member Function Documentation . . . 46
4.28.2.1 run() . . . 46
4.28.3 Member Data Documentation . . . 46
4.28.3.1 FALHA . . . 46
4.28.3.2 SUCESSO . . . 46
4.29 TUNota Class Reference . . . 47
4.29.1 Detailed Description . . . 47
4.29.2 Member Function Documentation . . . 47
4.29.2.1 run() . . . 47
4.29.3 Member Data Documentation . . . 47
4.29.3.1 FALHA . . . 47
4.29.3.2 SUCESSO . . . 47
4.30 TUSenha Class Reference . . . 48
4.30.1 Detailed Description . . . 48
4.30.2 Member Function Documentation . . . 48
4.30.2.1 run() . . . 48
4.30.3 Member Data Documentation . . . 48
4.30.3.1 FALHA . . . 48
4.30.3.2 SUCESSO . . . 48
4.31 TUSessao Class Reference . . . 49
4.31.1 Detailed Description . . . 49

4.31.2 Member Function Documentation . . . 49
4.31.2.1 run() . . . 49
4.31.3 Member Data Documentation . . . 49
4.31.3.1 FALHA . . . 49
4.31.3.2 SUCESSO . . . 49
4.32 TUTitulo Class Reference . . . 50
4.32.1 Detailed Description . . . 50
4.32.2 Member Function Documentation . . . 50
4.32.2.1 run() . . . 50
4.32.3 Member Data Documentation . . . 50
4.32.3.1 FALHA . . . 50
4.32.3.2 SUCESSO . . . 50
4.33 TUUsuario Class Reference . . . 51
4.33.1 Detailed Description . . . 51
4.33.2 Member Function Documentation . . . 51
4.33.2.1 run() . . . 51
4.33.3 Member Data Documentation . . . 51
4.33.3.1 FALHA . . . 51
4.33.3.2 SUCESSO . . . 51
4.34 Usuario Class Reference . . . 52
4.34.1 Detailed Description . . . 52
4.34.2 Member Function Documentation . . . 52
4.34.2.1 getEmail() . . . 52
4.34.2.2 getNome() . . . 53
4.34.2.3 getSenha() . . . 53
4.34.2.4 setEmail() . . . 53
4.34.2.5 setNome() . . . 53
4.34.2.6 setSenha() . . . 54

5 File Documentation **55**

5.1 dominios.h . . . 55
5.2 entidades.h . . . 58
5.3 testes_dominios.h . . . 61
5.4 testes_entidades.h . . . 64
5.5 dominios.cpp . . . 65
5.6 entidades.cpp . . . 71
5.7 main.cpp . . . 71
5.8 testes_dominios.cpp . . . 74
5.9 testes_entidades.cpp . . . 81

Index **85**

Chapter 1

TP1-Projeto-1

Primeiro projeto desenvolvido na disciplina de Técnicas de Programação 1, na Universidade de Brasília

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Avaliacao
Padrão para a representação da avaliação de uma excursão . . . 7
Cidade
Padrão para a representação de uma cidade . . . 9
Codigo
Padrão para a representação de um código . . . 11
Data
Padrão para a representação de uma data . . . 12
Descricao
Padrão para a representação de uma descrição . . . 13
Duracao
Padrão para a representação de uma duração . . . 15
Email
Padrão para a representação de um email . . . 16
Endereco
Padrão para a representação de uma duração . . . 17
Excursao
Padrão para a representação de uma excursão . . . 19
Horario
Padrão para a representação de um horário . . . 24
Idioma
Padrão para a representação de um idioma . . . 25
Nome
Padrão para a representação de um nome . . . 26
Nota
Padrão para a representação de uma nota . . . 27
Senha
Padrão para a representação de uma senha . . . 29
Sessao
Padrão para a representação de uma sessão de uma excursão . . . 30
Titulo
Padrão para a representação de uma duração . . . 33
TUAvaliacao . . . 34
TUCidade . . . 35
TUCodigo . . . 36

TUData 37
TUDescricao 38
TUDuracao 39
TUEmail 41
TUEndereco 42
TUExcursao 43
TUHorario 44
TUIdioma 45
TUNome 46
TUNota 47
TUSenha 48
TUSessao 49
TUTitulo 50
TUUsuario 51
Usuario
Padrão para representação de um usuário 52

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

headers/dominios.h 55
headers/entidades.h 58
headers/testes_dominios.h 61
headers/testes_entidades.h 64
source/dominios.cpp 65
source/entidades.cpp 71
source/main.cpp 71
source/testes_dominios.cpp 74
source/testes_entidades.cpp 81

Chapter 4

Class Documentation

4.1 Avaliacao Class Reference

Padrão para a representação da avaliação de uma excursão.

`#include <entidades.h>`

Public Member Functions

- void setCodigo (const Codigo &)
  Armazena o código da avaliação.
- Codigo getCodigo () const
  Retorna o código da avaliação.
- void setNota (const Nota &)
  Armazena a nota da avaliação.
- Nota getNota () const
  Retorna a nota da avaliação.
- void setDescricao (const Descricao &)
  Armazena a descrição da avaliação.
- Descricao getDescricao () const
  Retorna a descrição da avaliação.

4.1.1 Detailed Description

Padrão para a representação da avaliação de uma excursão.

Definition at line 116 of file entidades.h.

4.1.2 Member Function Documentation

4.1.2.1 getCodigo()

```
Codigo Avaliacao::getCodigo ( ) const  [inline]
```

Retorna o código da avaliação.

Returns

Codigo Código da avaliação

Definition at line 178 of file entidades.h.

4.1.2.2 getDescricao()

```
Descricao Avaliacao::getDescricao ( ) const  [inline]
```

Retorna a descrição da avaliação.

Returns

Descricao Descrição da avaliação

Definition at line 186 of file entidades.h.

4.1.2.3 getNota()

```
Nota Avaliacao::getNota ( ) const  [inline]
```

Retorna a nota da avaliação.

Returns

Nota Nota da avaliação

Definition at line 182 of file entidades.h.

4.1.2.4 setCodigo()

```
void Avaliacao::setCodigo (
            const Codigo & codigo )  [inline]
```

Armazena o código da avaliação.

Parameters

codigo	Código da avaliação

Definition at line 166 of file entidades.h.

4.1.2.5 setDescricao()

```
void Avaliacao::setDescricao (
            const Descricao & descricao )  [inline]
```

Armazena a descrição da avaliação.

Parameters

descricao	Descrição da avaliação

Definition at line 174 of file entidades.h.

4.1.2.6 setNota()

```
void Avaliacao::setNota (
            const Nota & nota )  [inline]
```

Armazena a nota da avaliação.

Parameters

nota	Nota da avaliação

Definition at line 170 of file entidades.h.

The documentation for this class was generated from the following file:

- headers/entidades.h

4.2 Cidade Class Reference

Padrão para a representação de uma cidade.

```
#include <dominios.h>
```

Public Member Functions

- void setValor (string)

  Armazena a cidade.
- string getValor () const

  Retorna a cidade.

4.2.1 Detailed Description

Padrão para a representação de uma cidade.

Regras de formato:

- Os valores válidos de cidade são: Hong Kong, Bangkok, Macau, Singapura, Londres, Paris, Dubai, Delhi, Istambul, Kuala Lumpur, Nova Iorque, Antalya, Mumbai, Shenzhen e Phuket.

Definition at line 38 of file dominios.h.

4.2.2 Member Function Documentation

4.2.2.1 getValor()

```
string Cidade::getValor ( ) const  [inline]
```

Retorna a cidade.

Returns

Cidade Cidade a ser retornada

Definition at line 59 of file dominios.h.

4.2.2.2 setValor()

```
void Cidade::setValor (
            string valor )
```

Armazena a cidade.

Parameters

valor	Cidade a ser armazenada

Definition at line 48 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.3 Codigo Class Reference

Padrão para a representação de um código.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)

  Armazena o código.
- string getValor () const

  Retorna a código.

4.3.1 Detailed Description

Padrão para a representação de um código.

Regras de formato:

- O código é representado por DDDDDDX, onde D é dígito (0-9) e X é dígito verificador
- Não existe código 000000

Definition at line 73 of file dominios.h.

4.3.2 Member Function Documentation

4.3.2.1 getValor()

```
string Codigo::getValor ( ) const  [inline]
```

Retorna a código.

Returns

Código Código a ser retornado

Definition at line 94 of file dominios.h.

4.3.2.2 setValor()

```
void Codigo::setValor (
            string valor )
```

Armazena o código.

Parameters

valor	Código a ser armazenado

Definition at line 77 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.4 Data Class Reference

Padrão para a representação de uma data.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)
  Armazena a data.
- string getValor () const
  Retorna a data.

4.4.1 Detailed Description

Padrão para a representação de uma data.

Regras de formato:

- A data é representada por DD-MES-ANO
- DD é o dia com faixa de valores entre 01 e 31
- MES é o mês com valores válidos iguais a: Jan, Fev, Mar, Abr, Mai, Jun, Jul, Ago, Set, Out, Nov, Dez
- ANO é o ano com faixa de valores entre 2000 e 9999
- A data considera a ocorrência de anos bissextos

Definition at line 111 of file dominios.h.

4.4.2 Member Function Documentation

4.4.2.1 getValor()

```
string Data::getValor ( ) const  [inline]
```

Retorna a data.

Returns

Data Data a ser retornada

Definition at line 132 of file dominios.h.

4.4.2.2 setValor()

```
void Data::setValor (
            string valor )
```

Armazena a data.

Parameters

valor	Data a ser armazenada

Definition at line 154 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.5 Descricao Class Reference

Padrão para a representação de uma descrição.

```
#include <dominios.h>
```

Public Member Functions

- void setValor (string)
  Armazena a descrição.
- string getValor () const
  Retorna a descrição.

4.5.1 Detailed Description

Padrão para a representação de uma descrição.

Regras de formato:

- Possui de 0 a 30 caracteres
- Não há espaços brancos em sequência
- Não há pontos em sequência

Definition at line 147 of file dominios.h.

4.5.2 Member Function Documentation

4.5.2.1 getValor()

```
string Descricao::getValor ( ) const  [inline]
```

Retorna a descrição.

Returns

Descrição Descrição a ser retornada

Definition at line 168 of file dominios.h.

4.5.2.2 setValor()

```
void Descricao::setValor (
            string valor )
```

Armazena a descrição.

Parameters

valor	Descrição a ser armazenada

Definition at line 192 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.6 Duracao Class Reference

Padrão para a representação de uma duração.

`#include <dominios.h>`

Public Member Functions

- void setValor (int)

  Armazena a duração.
- int getValor () const

  Retorna a duração.

4.6.1 Detailed Description

Padrão para a representação de uma duração.

Regras de formato:

- Os valores válidos de duração são: 30, 60, 90, 120 ou 180

Definition at line 181 of file dominios.h.

4.6.2 Member Function Documentation

4.6.2.1 getValor()

```
int Duracao::getValor ( ) const  [inline]
```

Retorna a duração.

Returns

Duração Duração a ser retornada

Definition at line 202 of file dominios.h.

4.6.2.2 setValor()

```
void Duracao::setValor (
            int valor )
```

Armazena a duração.

Parameters

valor	Duração a ser armazenada

Definition at line 204 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.7 Email Class Reference

Padrão para a representação de um email.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)

  Armazena o email.
- string getValor () const

  Retorna o email.

4.7.1 Detailed Description

Padrão para a representação de um email.

Regras de formato:

- O email é representado por parte_local@domínio
- parte-local é composta por até 64 caracteres
- Caractere de parte-local pode ser: ∼Letra maiúscula ou minúscula ∼Dígito ∼Caracteres especiais: !#$%&'+-/=?^_`{|}∼ ∼Ponto desde que não seja o primeiro ou o último caractere e que não ocorra em sequência
- domínio é composto por até 253 caracteres
- Caractere de domínio pode ser: ∼Letra maiúscula ou minúscula ∼Dígito ∼Hífen ∼Ponto desde que não seja o primeiro e que não ocorra em sequência

Definition at line 227 of file dominios.h.

4.7.2 Member Function Documentation

4.7.2.1 getValor()

```
string Email::getValor ( ) const  [inline]
```

Retorna o email.

Returns

Email Email a ser retornado

Definition at line 248 of file dominios.h.

4.7.2.2 setValor()

```
void Email::setValor (
            string valor )
```

Armazena o email.

Parameters

valor	Email a ser armazenado

Definition at line 280 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.8 Endereco Class Reference

Padrão para a representação de uma duração.

```
#include <dominios.h>
```

Public Member Functions

- void setValor (string)
  Armazena o endereço.
- string getValor () const
  Retorna o endereço.

4.8.1 Detailed Description

Padrão para a representação de uma duração.

Regras de formato:

- Possui de 0 a 20 caracteres
- Não há espaços em branco em sequência
- Não há pontos em sequência

Definition at line 263 of file dominios.h.

4.8.2 Member Function Documentation

4.8.2.1 getValor()

```
string Endereco::getValor ( ) const  [inline]
```

Retorna o endereço.

Returns

Endereço Endereço a ser retornado

Definition at line 284 of file dominios.h.

4.8.2.2 setValor()

```
void Endereco::setValor (
            string valor )
```

Armazena o endereço.

Parameters

valor	Endereço a ser armazenado

Definition at line 321 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.9 Excursao Class Reference

Padrão para a representação de uma excursão.

`#include <entidades.h>`

Public Member Functions

- void setCodigo (const Codigo &)
  Armazena o código da excursão.
- Codigo getCodigo () const
  Retorna o código da excursão.
- void setTitulo (const Titulo &)
  Armazena o título da excursão.
- Titulo getTitulo () const
  Retorna o título da excursão.
- void setNota (const Nota &)
  Armazena a nota da excursão.
- Nota getNota () const
  Retorna a nota da excursão.
- void setCidade (const Cidade &)
  Armazena a cidade da excursão.
- Cidade getCidade () const
  Retorna a cidade da excursão.
- void setDuracao (const Duracao &)
  Armazena a duração da excursão.
- Duracao getDuracao () const
  Retorna a duração da excursão.
- void setDescricao (const Descricao &)
  Armazena a descrição da excursão.
- Descricao getDescricao () const
  Retorna a descrição da excursão.
- void setEndereco (const Endereco &)
  Armazena o endereço da excursão.
- Endereco getEndereco () const
  Retorna o endereço da excursão.

4.9.1 Detailed Description

Padrão para a representação de uma excursão.

Definition at line 297 of file entidades.h.

4.9.2 Member Function Documentation

4.9.2.1 getCidade()

```
Cidade Excursao::getCidade ( ) const  [inline]
```

Retorna a cidade da excursão.

Returns

Cidade Cidade da excursão

Definition at line 446 of file entidades.h.

4.9.2.2 getCodigo()

```
Codigo Excursao::getCodigo ( ) const  [inline]
```

Retorna o código da excursão.

Returns

Codigo Código da excursão

Definition at line 434 of file entidades.h.

4.9.2.3 getDescricao()

```
Descricao Excursao::getDescricao ( ) const  [inline]
```

Retorna a descrição da excursão.

Returns

Descricao Descrição da excursão

Definition at line 454 of file entidades.h.

4.9.2.4 getDuracao()

```
Duracao Excursao::getDuracao ( ) const  [inline]
```

Retorna a duração da excursão.

Returns

Duracao Duração da excursão

Definition at line 450 of file entidades.h.

4.9.2.5 getEndereco()

```
Endereco Excursao::getEndereco ( ) const  [inline]
```

Retorna o endereço da excursão.

Returns

Endereco Endereço da excursão

Definition at line 458 of file entidades.h.

4.9.2.6 getNota()

```
Nota Excursao::getNota ( ) const  [inline]
```

Retorna a nota da excursão.

Returns

Nota Nota da excursão

Definition at line 442 of file entidades.h.

4.9.2.7 getTitulo()

```
Titulo Excursao::getTitulo ( ) const  [inline]
```

Retorna o título da excursão.

Returns

Titulo Título da excursão

Definition at line 438 of file entidades.h.

4.9.2.8 setCidade()

```
void Excursao::setCidade (
            const Cidade & cidade )  [inline]
```

Armazena a cidade da excursão.

Parameters

cidade	Cidade a ser armazenada na excursão

Definition at line 418 of file entidades.h.

4.9.2.9 setCodigo()

```
void Excursao::setCodigo (
            const Codigo & codigo )  [inline]
```

Armazena o código da excursão.

Parameters

codigo	Código a ser armazenado na excursão

Definition at line 406 of file entidades.h.

4.9.2.10 setDescricao()

```
void Excursao::setDescricao (
            const Descricao & descricao )  [inline]
```

Armazena a descrição da excursão.

Parameters

descricao	Descrição a ser armazenada na excursão

Definition at line 426 of file entidades.h.

4.9.2.11 setDuracao()

```
void Excursao::setDuracao (
            const Duracao & duracao )  [inline]
```

Armazena a duração da excursão.

Parameters

duracao	Duração a ser armazenada na excursão

Definition at line 422 of file entidades.h.

4.9.2.12 setEndereco()

```
void Excursao::setEndereco (
            const Endereco & endereco )  [inline]
```

Armazena o endereço da excursão.

Parameters

endereco	Endereço a ser armazenado na excursão

Definition at line 430 of file entidades.h.

4.9.2.13 setNota()

```
void Excursao::setNota (
            const Nota & nota )  [inline]
```

Armazena a nota da excursão.

Parameters

nota	Nota a ser armazenada na excursão

Definition at line 414 of file entidades.h.

4.9.2.14 setTitulo()

```
void Excursao::setTitulo (
            const Titulo & titulo )  [inline]
```

Armazena o título da excursão.

Parameters

titulo	Título a ser armazenado na excursão

Definition at line 410 of file entidades.h.

The documentation for this class was generated from the following file:

- headers/entidades.h

4.10 Horario Class Reference

Padrão para a representação de um horário.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)
  Armazena o horário.
- string getValor () const
  Retorna o horário.

4.10.1 Detailed Description

Padrão para a representação de um horário.

Regras de formato:

- O horário é representado por HH:MM
- HH é o horário com faixa de valores entre 00 e 23
- MM é o minuto com faixa de valores entre 00 e 59

Definition at line 299 of file dominios.h.

4.10.2 Member Function Documentation

4.10.2.1 getValor()

```
string Horario::getValor ( ) const  [inline]
```

Retorna o horário.

Returns

Horário Horário a ser retornado

Definition at line 320 of file dominios.h.

4.10.2.2 setValor()

```
void Horario::setValor (
            string valor )
```

Armazena o horário.

Parameters

valor	Horário a ser armazenado

Definition at line 355 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.11 Idioma Class Reference

Padrão para a representação de um idioma.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)
  Armazena o idioma.
- string getValor () const
  Retorna o idioma.

4.11.1 Detailed Description

Padrão para a representação de um idioma.

Regras de formato:

- Os valores válidos de idioma são: Ingles, Chines, Mandarim, Hindi, Espanhol, Frances, Arabe, Bengali, Russo, Portugues, Indonesio

Definition at line 333 of file dominios.h.

4.11.2 Member Function Documentation

4.11.2.1 getValor()

```
string Idioma::getValor ( ) const  [inline]
```

Retorna o idioma.

Returns

Idioma Idioma a ser retornado

Definition at line 354 of file dominios.h.

4.11.2.2 setValor()

```
void Idioma::setValor (
            string valor )
```

Armazena o idioma.

Parameters

valor	Idioma a ser armazenado

Definition at line 367 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.12 Nome Class Reference

Padrão para a representação de um nome.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)
  Armazena o nome.
- string getValor () const
  Retorna o nome.

4.12.1 Detailed Description

Padrão para a representação de um nome.

Regras de formato:

- Possui de 5 a 20 caracteres
- Cada caractere é letra, ponto ou espaço em branco
- Ponto é precedido por letra, é último caractere ou é seguido por um espaço em branco
- Não há espaços em branco em sequência
- A primeira letra de cada termo é maiúscula

Definition at line 371 of file dominios.h.

4.12.2 Member Function Documentation

4.12.2.1 getValor()

```
string Nome::getValor ( ) const  [inline]
```

Retorna o nome.

Returns

Nome Nome a ser retornado

Definition at line 392 of file dominios.h.

4.12.2.2 setValor()

```
void Nome::setValor (
            string valor )
```

Armazena o nome.

Parameters

valor	Nome a ser armazenado

Definition at line 415 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.13 Nota Class Reference

Padrão para a representação de uma nota.

```
#include <dominios.h>
```

Public Member Functions

- void setValor (int)
  Armazena a nota.
- int getValor () const
  Retorna a nota.

4.13.1 Detailed Description

Padrão para a representação de uma nota.

Regras de formato:

- Os valores válidos de nota são: 0, 1, 2, 3, 4 ou 5

Definition at line 405 of file dominios.h.

4.13.2 Member Function Documentation

4.13.2.1 getValor()

```
int Nota::getValor ( ) const  [inline]
```

Retorna a nota.

Returns

Nota Nota a ser retornada

Definition at line 426 of file dominios.h.

4.13.2.2 setValor()

```
void Nota::setValor (
            int valor )
```

Armazena a nota.

Parameters

valor	Nota a ser armazenada

Definition at line 427 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.14 Senha Class Reference

Padrão para a representação de uma senha.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)
  Armazena a senha.
- string getValor () const
  Retorna a senha.

4.14.1 Detailed Description

Padrão para a representação de uma senha.

Regras de formato:

- A senha é representada por XXXXXX
- Cada caractere X é letra ou dígito
- Não existe caracter repetido
- Existe pelo menos uma letra maiúscula, uma letra minúscula e um dígito

Definition at line 442 of file dominios.h.

4.14.2 Member Function Documentation

4.14.2.1 getValor()

```
string Senha::getValor ( ) const  [inline]
```

Retorna a senha.

Returns

Senha Senha a ser retornada

Definition at line 463 of file dominios.h.

4.14.2.2 setValor()

```
void Senha::setValor (
            string valor )
```

Armazena a senha.

Parameters

valor	Senha a ser armazenada

Definition at line 468 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.15 Sessao Class Reference

Padrão para a representação de uma sessão de uma excursão.

`#include <entidades.h>`

Public Member Functions

- void setCodigo (const Codigo &)
  Armazena o código da sessão.
- Codigo getCodigo () const
  Retorna o código da sessão.
- void setData (const Data &)
  Armazena a data da sessão.
- Data getData () const
  Retorna a data da sessão.
- void setHorario (const Horario &)
  Armazena o horário da sessão.
- Horario getHorario () const
  Retorna o horário da sessão.
- void setIdioma (const Idioma &)
  Armazena o idioma da sessão.
- Idioma getIdioma () const
  Retorna o idioma da sessão.

4.15.1 Detailed Description

Padrão para a representação de uma sessão de uma excursão.

Definition at line 195 of file entidades.h.

4.15.2 Member Function Documentation

4.15.2.1 getCodigo()

```
Codigo Sessao::getCodigo ( ) const  [inline]
```

Retorna o código da sessão.

Returns

Codigo Código da sessão

Definition at line 276 of file entidades.h.

4.15.2.2 getData()

```
Data Sessao::getData ( ) const  [inline]
```

Retorna a data da sessão.

Returns

Data Data da sessão

Definition at line 280 of file entidades.h.

4.15.2.3 getHorario()

```
Horario Sessao::getHorario ( ) const  [inline]
```

Retorna o horário da sessão.

Returns

Horario Horário da sessão

Definition at line 284 of file entidades.h.

4.15.2.4 getIdioma()

```
Idioma Sessao::getIdioma ( ) const  [inline]
```

Retorna o idioma da sessão.

Returns

Idioma Idioma da sessão

Definition at line 288 of file entidades.h.

4.15.2.5 setCodigo()

```
void Sessao::setCodigo (
            const Codigo & codigo )  [inline]
```

Armazena o código da sessão.

Parameters

codigo	Código da sessão

Definition at line 260 of file entidades.h.

4.15.2.6 setData()

```
void Sessao::setData (
            const Data & data )  [inline]
```

Armazena a data da sessão.

Parameters

data	Data da sessão

Definition at line 264 of file entidades.h.

4.15.2.7 setHorario()

```
void Sessao::setHorario (
            const Horario & horario )  [inline]
```

Armazena o horário da sessão.

Parameters

horario	Horário da sessão

Definition at line 268 of file entidades.h.

4.15.2.8 setIdioma()

```
void Sessao::setIdioma (
            const Idioma & idioma )  [inline]
```

Armazena o idioma da sessão.

Parameters

idioma	Idioma da sessão

Definition at line 272 of file entidades.h.

The documentation for this class was generated from the following file:

- headers/entidades.h

4.16 Titulo Class Reference

Padrão para a representação de uma duração.

`#include <dominios.h>`

Public Member Functions

- void setValor (string)
  Armazena o título.
- string getValor () const
  Retorna o título.

4.16.1 Detailed Description

Padrão para a representação de uma duração.

Regras de formato:

- Possui de 5 a 20 letras (não caracteres)
- Não há espaços em branco em sequência
- Não há pontos em sequência

Definition at line 478 of file dominios.h.

4.16.2 Member Function Documentation

4.16.2.1 getValor()

```
string Titulo::getValor ( ) const  [inline]
```

Retorna o título.

Returns

Título Título a ser retornado

Definition at line 499 of file dominios.h.

4.16.2.2 setValor()

```
void Titulo::setValor (
            string valor )
```

Armazena o título.

Parameters

valor	Título a ser armazenado

Definition at line 503 of file dominios.cpp.

The documentation for this class was generated from the following files:

- headers/dominios.h
- source/dominios.cpp

4.17 TUAvaliacao Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.17.1 Detailed Description

Definition at line 48 of file testes_entidades.h.

4.17.2 Member Function Documentation

4.17.2.1 run()

```
int TUAvaliacao::run ( )
```

Definition at line 107 of file testes_entidades.cpp.

4.17.3 Member Data Documentation

4.17.3.1 FALHA

```
const int TUAvaliacao::FALHA = -1  [static]
```

Definition at line 60 of file testes_entidades.h.

4.17.3.2 SUCESSO

```
const int TUAvaliacao::SUCESSO = 0  [static]
```

Definition at line 59 of file testes_entidades.h.

The documentation for this class was generated from the following files:

- headers/testes_entidades.h
- source/testes_entidades.cpp

4.18 TUCidade Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.18.1 Detailed Description

Definition at line 29 of file testes_dominios.h.

4.18.2 Member Function Documentation

4.18.2.1 run()

```
int TUCidade::run ( )
```

Definition at line 82 of file testes_dominios.cpp.

4.18.3 Member Data Documentation

4.18.3.1 FALHA

```
const int TUCidade::FALHA = -1  [static]
```

Definition at line 42 of file testes_dominios.h.

4.18.3.2 SUCESSO

```
const int TUCidade::SUCESSO = 0  [static]
```

Definition at line 41 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.19 TUCodigo Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.19.1 Detailed Description

Definition at line 46 of file testes_dominios.h.

4.19.2 Member Function Documentation

4.19.2.1 run()

```
int TUCodigo::run ( )
```

Definition at line 122 of file testes_dominios.cpp.

4.19.3 Member Data Documentation

4.19.3.1 FALHA

```
const int TUCodigo::FALHA = -1  [static]
```

Definition at line 59 of file testes_dominios.h.

4.19.3.2 SUCESSO

```
const int TUCodigo::SUCESSO = 0  [static]
```

Definition at line 58 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.20 TUData Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.20.1 Detailed Description

Definition at line 63 of file testes_dominios.h.

4.20.2 Member Function Documentation

4.20.2.1 run()

```
int TUData::run ( )
```

Definition at line 162 of file testes_dominios.cpp.

4.20.3 Member Data Documentation

4.20.3.1 FALHA

```
const int TUData::FALHA = -1  [static]
```

Definition at line 76 of file testes_dominios.h.

4.20.3.2 SUCESSO

```
const int TUData::SUCESSO = 0  [static]
```

Definition at line 75 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.21 TUDescricao Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.21.1 Detailed Description

Definition at line 80 of file testes_dominios.h.

4.21.2 Member Function Documentation

4.21.2.1 run()

```
int TUDescricao::run ( )
```

Definition at line 202 of file testes_dominios.cpp.

4.21.3 Member Data Documentation

4.21.3.1 FALHA

```
const int TUDescricao::FALHA = -1  [static]
```

Definition at line 93 of file testes_dominios.h.

4.21.3.2 SUCESSO

```
const int TUDescricao::SUCESSO = 0  [static]
```

Definition at line 92 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.22 TUDuracao Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.22.1 Detailed Description

Definition at line 97 of file testes_dominios.h.

4.22.2 Member Function Documentation

4.22.2.1 run()

```
int TUDuracao::run ( )
```

Definition at line 242 of file testes_dominios.cpp.

4.22.3 Member Data Documentation

4.22.3.1 FALHA

```
const int TUDuracao::FALHA = -1  [static]
```

Definition at line 110 of file testes_dominios.h.

4.22.3.2 SUCESSO

```
const int TUDuracao::SUCESSO = 0  [static]
```

Definition at line 109 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.23 TUEmail Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.23.1 Detailed Description

Definition at line 114 of file testes_dominios.h.

4.23.2 Member Function Documentation

4.23.2.1 run()

```
int TUEmail::run ( )
```

Definition at line 282 of file testes_dominios.cpp.

4.23.3 Member Data Documentation

4.23.3.1 FALHA

```
const int TUEmail::FALHA = -1  [static]
```

Definition at line 127 of file testes_dominios.h.

4.23.3.2 SUCESSO

```
const int TUEmail::SUCESSO = 0  [static]
```

Definition at line 126 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.24 TUEndereco Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.24.1 Detailed Description

Definition at line 131 of file testes_dominios.h.

4.24.2 Member Function Documentation

4.24.2.1 run()

```
int TUEndereco::run ( )
```

Definition at line 322 of file testes_dominios.cpp.

4.24.3 Member Data Documentation

4.24.3.1 FALHA

```
const int TUEndereco::FALHA = -1  [static]
```

Definition at line 144 of file testes_dominios.h.

4.24.3.2 SUCESSO

```
const int TUEndereco::SUCESSO = 0  [static]
```

Definition at line 143 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.25 TUExcursao Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.25.1 Detailed Description

Definition at line 81 of file testes_entidades.h.

4.25.2 Member Function Documentation

4.25.2.1 run()

```
int TUExcursao::run ( )
```

Definition at line 209 of file testes_entidades.cpp.

4.25.3 Member Data Documentation

4.25.3.1 FALHA

```
const int TUExcursao::FALHA = -1  [static]
```

Definition at line 97 of file testes_entidades.h.

4.25.3.2 SUCESSO

```
const int TUExcursao::SUCESSO = 0  [static]
```

Definition at line 96 of file testes_entidades.h.

The documentation for this class was generated from the following files:

- headers/testes_entidades.h
- source/testes_entidades.cpp

4.26 TUHorario Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.26.1 Detailed Description

Definition at line 148 of file testes_dominios.h.

4.26.2 Member Function Documentation

4.26.2.1 run()

```
int TUHorario::run ( )
```

Definition at line 362 of file testes_dominios.cpp.

4.26.3 Member Data Documentation

4.26.3.1 FALHA

```
const int TUHorario::FALHA = -1  [static]
```

Definition at line 161 of file testes_dominios.h.

4.26.3.2 SUCESSO

```
const int TUHorario::SUCESSO = 0  [static]
```

Definition at line 160 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.27 TUIdioma Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.27.1 Detailed Description

Definition at line 165 of file testes_dominios.h.

4.27.2 Member Function Documentation

4.27.2.1 run()

```
int TUIdioma::run ( )
```

Definition at line 402 of file testes_dominios.cpp.

4.27.3 Member Data Documentation

4.27.3.1 FALHA

```
const int TUIdioma::FALHA = -1  [static]
```

Definition at line 178 of file testes_dominios.h.

4.27.3.2 SUCESSO

```
const int TUIdioma::SUCESSO = 0  [static]
```

Definition at line 177 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.28 TUNome Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.28.1 Detailed Description

Definition at line 182 of file testes_dominios.h.

4.28.2 Member Function Documentation

4.28.2.1 run()

```
int TUNome::run ( )
```

Definition at line 442 of file testes_dominios.cpp.

4.28.3 Member Data Documentation

4.28.3.1 FALHA

```
const int TUNome::FALHA = -1  [static]
```

Definition at line 195 of file testes_dominios.h.

4.28.3.2 SUCESSO

```
const int TUNome::SUCESSO = 0  [static]
```

Definition at line 194 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.29 TUNota Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.29.1 Detailed Description

Definition at line 199 of file testes_dominios.h.

4.29.2 Member Function Documentation

4.29.2.1 run()

```
int TUNota::run ( )
```

Definition at line 482 of file testes_dominios.cpp.

4.29.3 Member Data Documentation

4.29.3.1 FALHA

```
const int TUNota::FALHA = -1  [static]
```

Definition at line 212 of file testes_dominios.h.

4.29.3.2 SUCESSO

```
const int TUNota::SUCESSO = 0  [static]
```

Definition at line 211 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.30 TUSenha Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.30.1 Detailed Description

Definition at line 216 of file testes_dominios.h.

4.30.2 Member Function Documentation

4.30.2.1 run()

```
int TUSenha::run ( )
```

Definition at line 522 of file testes_dominios.cpp.

4.30.3 Member Data Documentation

4.30.3.1 FALHA

```
const int TUSenha::FALHA = -1  [static]
```

Definition at line 229 of file testes_dominios.h.

4.30.3.2 SUCESSO

```
const int TUSenha::SUCESSO = 0  [static]
```

Definition at line 228 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.31 TUSessao Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.31.1 Detailed Description

Definition at line 64 of file testes_entidades.h.

4.31.2 Member Function Documentation

4.31.2.1 run()

```
int TUSessao::run ( )
```

Definition at line 149 of file testes_entidades.cpp.

4.31.3 Member Data Documentation

4.31.3.1 FALHA

```
const int TUSessao::FALHA = -1  [static]
```

Definition at line 77 of file testes_entidades.h.

4.31.3.2 SUCESSO

```
const int TUSessao::SUCESSO = 0  [static]
```

Definition at line 76 of file testes_entidades.h.

The documentation for this class was generated from the following files:

- headers/testes_entidades.h
- source/testes_entidades.cpp

4.32 TUTitulo Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.32.1 Detailed Description

Definition at line 233 of file testes_dominios.h.

4.32.2 Member Function Documentation

4.32.2.1 run()

```
int TUTitulo::run ( )
```

Definition at line 562 of file testes_dominios.cpp.

4.32.3 Member Data Documentation

4.32.3.1 FALHA

```
const int TUTitulo::FALHA = -1  [static]
```

Definition at line 246 of file testes_dominios.h.

4.32.3.2 SUCESSO

```
const int TUTitulo::SUCESSO = 0  [static]
```

Definition at line 245 of file testes_dominios.h.

The documentation for this class was generated from the following files:

- headers/testes_dominios.h
- source/testes_dominios.cpp

4.33 TUUsuario Class Reference

Public Member Functions

- int run ()

Static Public Attributes

- static const int SUCESSO = 0
- static const int FALHA = -1

4.33.1 Detailed Description

Definition at line 32 of file testes_entidades.h.

4.33.2 Member Function Documentation

4.33.2.1 run()

```
int TUUsuario::run ( )
```

Definition at line 71 of file testes_entidades.cpp.

4.33.3 Member Data Documentation

4.33.3.1 FALHA

```
const int TUUsuario::FALHA = -1  [static]
```

Definition at line 44 of file testes_entidades.h.

4.33.3.2 SUCESSO

```
const int TUUsuario::SUCESSO = 0  [static]
```

Definition at line 43 of file testes_entidades.h.

The documentation for this class was generated from the following files:

- headers/testes_entidades.h
- source/testes_entidades.cpp

4.34 Usuario Class Reference

Padrão para representação de um usuário.

`#include <entidades.h>`

Public Member Functions

- void setNome (const Nome &)
  Armazena nome do usuário.
- Nome getNome () const
  Retorna o nome do usuário.
- void setEmail (const Email &)
  Armazena email do usuário.
- Email getEmail () const
  Retorna o email do usuário.
- void setSenha (const Senha &)
  Armazena a senha do usuário.
- Senha getSenha () const
  Retorna a senha.

4.34.1 Detailed Description

Padrão para representação de um usuário.

Definition at line 38 of file entidades.h.

4.34.2 Member Function Documentation

4.34.2.1 getEmail()

```
Email Usuario::getEmail ( ) const  [inline]
```

Retorna o email do usuário.

Returns

Email Email a ser retornado

Definition at line 103 of file entidades.h.

4.34.2.2 getNome()

```
Nome Usuario::getNome ( ) const  [inline]
```

Retorna o nome do usuário.

Returns

Nome Nome a ser retornado

Definition at line 99 of file entidades.h.

4.34.2.3 getSenha()

```
Senha Usuario::getSenha ( ) const  [inline]
```

Retorna a senha.

Returns

Senha Senha a ser retornada

Definition at line 107 of file entidades.h.

4.34.2.4 setEmail()

```
void Usuario::setEmail (
            const Email & email )  [inline]
```

Armazena email do usuário.

Parameters

email	Email a ser armazenado

Definition at line 91 of file entidades.h.

4.34.2.5 setNome()

```
void Usuario::setNome (
            const Nome & nome )  [inline]
```

Armazena nome do usuário.

Parameters

nome	Nome a ser armazenado

Definition at line 87 of file entidades.h.

4.34.2.6 setSenha()

```
void Usuario::setSenha (
            const Senha & senha )  [inline]
```

Armazena a senha do usuário.

Parameters

senha	Senha a ser armazenada

Definition at line 95 of file entidades.h.

The documentation for this class was generated from the following file:

- headers/entidades.h

Chapter 5

File Documentation

5.1 dominios.h

```
00001 #ifndef DOMINIOS_H_INCLUDED
00002 #define DOMINIOS_H_INCLUDED
00003
00004 #include <stdexcept>
00005
00006 using namespace std;
00007
00008
00009 /* ------------------------------------------------------------------------
00010 // Estrutura de c◆digo para declara◆◆o de classe dom◆nio.
00011 // Substituir Dominio por nome da classe.
00012 // Substituir Tipo.
00013
00014 class Dominio {
00015     private:
00016         Tipo valor;                     // Atributo para armazenar valor.
00017         void validar(Tipo);             // M◆todo para validar valor.
00018     public:
00019         void setValor(Tipo);            // M◆todo para atribuir valor.
00020         Tipo getValor() const;          // M◆todo para recuperar valor.
00021 };
00022
00023 inline Tipo Dominio::getValor() const{
00024     return valor;
00025 }
00026
00027 ------------------------------------------------------------------------ */
00028
00037 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00038 class Cidade {
00039     private:
00040         string valor;
00041         void validar(string);
00042     public:
00043
00049         void setValor(string);
00050
00056         string getValor() const;
00057 };
00058
00059 inline string Cidade::getValor() const{
00060     return valor;
00061 }
00062
00072 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00073 class Codigo {
00074     private:
00075         string valor;
00076         void validar(string);
00077     public:
00078
00084         void setValor(string);
00085
00091         string getValor() const;
00092 };
00093
00094 inline string Codigo::getValor() const{
00095     return valor;
```

```
00096 }
00097
00110 // Classe implementada pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00111 class Data {
00112     private:
00113         string valor;
00114         void validar(string);
00115     public:
00116
00122         void setValor(string);
00123
00129         string getValor() const;
00130 };
00131
00132 inline string Data::getValor() const{
00133     return valor;
00134 }
00135
00146 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00147 class Descricao {
00148     private:
00149         string valor;
00150         void validar(string);
00151     public:
00152
00158         void setValor(string);
00159
00165         string getValor() const;
00166 };
00167
00168 inline string Descricao::getValor() const{
00169     return valor;
00170 }
00171
00180 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00181 class Duracao {
00182     private:
00183         int valor;
00184         void validar(int);
00185     public:
00186
00192         void setValor(int);
00193
00199         int getValor() const;
00200 };
00201
00202 inline int Duracao::getValor() const{
00203     return valor;
00204 }
00205
00226 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00227 class Email {
00228     private:
00229         string valor;
00230         void validar(string);
00231     public:
00232
00238         void setValor(string);
00239
00245         string getValor() const;
00246 };
00247
00248 inline string Email::getValor() const{
00249     return valor;
00250 }
00251
00262 // Classe implementada pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00263 class Endereco {
00264     private:
00265         string valor;
00266         void validar(string);
00267     public:
00268
00274         void setValor(string);
00275
00281         string getValor() const;
00282 };
00283
00284 inline string Endereco::getValor() const{
00285     return valor;
00286 }
00287
00298 // Classe implementada pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00299 class Horario {
00300     private:
00301         string valor;
00302         void validar(string);
```

```
00303     public:
00304
00310         void setValor(string);
00311
00317         string getValor() const;
00318 };
00319
00320 inline string Horario::getValor() const{
00321     return valor;
00322 }
00323
00332 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00333 class Idioma {
00334     private:
00335         string valor;
00336         void validar(string);
00337     public:
00338
00344         void setValor(string);
00345
00351         string getValor() const;
00352 };
00353
00354 inline string Idioma::getValor() const{
00355     return valor;
00356 }
00357
00370 // Classe implementada pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00371 class Nome {
00372     private:
00373         string valor;
00374         void validar(string);
00375     public:
00376
00382         void setValor(string);
00383
00389         string getValor() const;
00390 };
00391
00392 inline string Nome::getValor() const{
00393     return valor;
00394 }
00395
00404 // Classe implementada pelo aluno David Fanchic Chatelard, matrícula 180138863
00405 class Nota {
00406     private:
00407         int valor;
00408         void validar(int);
00409     public:
00410
00416         void setValor(int);
00417
00423         int getValor() const;
00424 };
00425
00426 inline int Nota::getValor() const{
00427     return valor;
00428 }
00429
00441 // Classe implementada pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00442 class Senha {
00443     private:
00444         string valor;
00445         void validar(string);
00446     public:
00447
00453         void setValor(string);
00454
00460         string getValor() const;
00461 };
00462
00463 inline string Senha::getValor() const{
00464     return valor;
00465 }
00466
00477 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00478 class Titulo {
00479     private:
00480         string valor;
00481         void validar(string);
00482     public:
00483
00489         void setValor(string);
00490
00496         string getValor() const;
00497 };
00498
```

```
00499 inline string Titulo::getValor() const{
00500     return valor;
00501 }
00502
00503 #endif
```

5.2 entidades.h

```
00001 #ifndef ENTIDADES_H_INCLUDED
00002 #define ENTIDADES_H_INCLUDED
00003
00004 #include "dominios.h"
00005
00006 #include <string>
00007
00008 using namespace std;
00009
00010 /* ------------------------------------------------------------------------
00011 // Estrutura de c◆digo sugerida para declara◆◆o de classe entidade.
00012 // Substituir Entidade por nome da classe.
00013 // Substituir Dominio.
00014 // Substituir nomeAtributo.
00015
00016 class Entidade {
00017     private:
00018             Dominio nomeAtributo;
00019     public:
00020             void setnomeAtributo(const Dominio&);
00021             Dominio getnomeAtributo() const;
00022 };
00023
00024 inline void Entidade::setnomeAtributo(const Dominio &nomeAtributo){
00025     this->nomeAtributo = nomeAtributo;
00026 }
00027
00028 inline Dominio Entidade::getnomeAtributo() const{
00029     return nomeAtributo;
00030 }
00031
00032 ------------------------------------------------------------------------ */
00033
00038 class Usuario {
00039     private:
00040             Nome nome;
00041             Email email;
00042             Senha senha;
00043     public:
00049             void setNome(const Nome&);
00050
00056             Nome getNome() const;
00057
00063             void setEmail(const Email&);
00064
00070             Email getEmail() const;
00071
00077             void setSenha(const Senha&);
00078
00084             Senha getSenha() const;
00085 };
00086
00087 inline void Usuario::setNome(const Nome &nome){
00088     this->nome = nome;
00089 }
00090
00091 inline void Usuario::setEmail(const Email &email){
00092     this->email = email;
00093 }
00094
00095 inline void Usuario::setSenha(const Senha &senha){
00096     this->senha = senha;
00097 }
00098
00099 inline Nome Usuario::getNome() const{
00100     return nome;
00101 }
00102
00103 inline Email Usuario::getEmail() const{
00104     return email;
00105 }
00106
00107 inline Senha Usuario::getSenha() const{
00108     return senha;
00109 }
```

```
00110
00111
00116 class Avaliacao {
00117     private:
00118             Codigo codigo;
00119             Nota nota;
00120             Descricao descricao;
00121     public:
00127             void setCodigo(const Codigo&);
00128
00134             Codigo getCodigo() const;
00135
00141             void setNota(const Nota&);
00142
00148             Nota getNota() const;
00149
00155             void setDescricao(const Descricao&);
00156
00162             Descricao getDescricao() const;
00163 };
00164
00165
00166 inline void Avaliacao::setCodigo(const Codigo &codigo){
00167     this->codigo = codigo;
00168 }
00169
00170 inline void Avaliacao::setNota(const Nota &nota){
00171     this->nota = nota;
00172 }
00173
00174 inline void Avaliacao::setDescricao(const Descricao &descricao){
00175     this->descricao = descricao;
00176 }
00177
00178 inline Codigo Avaliacao::getCodigo() const{
00179     return codigo;
00180 }
00181
00182 inline Nota Avaliacao::getNota() const{
00183     return nota;
00184 }
00185
00186 inline Descricao Avaliacao::getDescricao() const{
00187     return descricao;
00188 }
00189
00190
00195 class Sessao {
00196     private:
00197             Codigo codigo;
00198             Data data;
00199             Horario horario;
00200             Idioma idioma;
00201     public:
00207             void setCodigo(const Codigo&);
00208
00214             Codigo getCodigo() const;
00215
00221             void setData(const Data&);
00222
00228             Data getData() const;
00229
00235             void setHorario(const Horario&);
00236
00242             Horario getHorario() const;
00243
00249             void setIdioma(const Idioma&);
00250
00256             Idioma getIdioma() const;
00257
00258 };
00259
00260 inline void Sessao::setCodigo(const Codigo &codigo){
00261     this->codigo = codigo;
00262 }
00263
00264 inline void Sessao::setData(const Data &data){
00265     this->data = data;
00266 }
00267
00268 inline void Sessao::setHorario(const Horario &horario){
00269     this->horario = horario;
00270 }
00271
00272 inline void Sessao::setIdioma(const Idioma &idioma){
00273     this->idioma = idioma;
00274 }
```

```
00275
00276 inline Codigo Sessao::getCodigo() const{
00277     return codigo;
00278 }
00279
00280 inline Data Sessao::getData() const{
00281     return data;
00282 }
00283
00284 inline Horario Sessao::getHorario() const{
00285     return horario;
00286 }
00287
00288 inline Idioma Sessao::getIdioma() const{
00289     return idioma;
00290 }
00291
00292
00297 class Excursao {
00298     private:
00299             Codigo codigo;
00300             Titulo titulo;
00301             Nota nota;
00302             Cidade cidade;
00303             Duracao duracao;
00304             Descricao descricao;
00305             Endereco endereco;
00306     public:
00312             void setCodigo(const Codigo&);
00313
00319             Codigo getCodigo() const;
00320
00326             void setTitulo(const Titulo&);
00327
00333             Titulo getTitulo() const;
00334
00340             void setNota(const Nota&);
00341
00347             Nota getNota() const;
00348
00354             void setCidade(const Cidade&);
00355
00361             Cidade getCidade() const;
00362
00368             void setDuracao(const Duracao&);
00369
00375             Duracao getDuracao() const;
00376
00382             void setDescricao(const Descricao&);
00383
00389             Descricao getDescricao() const;
00390
00396             void setEndereco(const Endereco&);
00397
00403             Endereco getEndereco() const;
00404 };
00405
00406 inline void Excursao::setCodigo(const Codigo &codigo){
00407     this->codigo = codigo;
00408 }
00409
00410 inline void Excursao::setTitulo(const Titulo &titulo){
00411     this->titulo = titulo;
00412 }
00413
00414 inline void Excursao::setNota(const Nota &nota){
00415     this->nota = nota;
00416 }
00417
00418 inline void Excursao::setCidade(const Cidade &cidade){
00419     this->cidade = cidade;
00420 }
00421
00422 inline void Excursao::setDuracao(const Duracao &duracao){
00423     this->duracao = duracao;
00424 }
00425
00426 inline void Excursao::setDescricao(const Descricao &descricao){
00427     this->descricao = descricao;
00428 }
00429
00430 inline void Excursao::setEndereco(const Endereco &endereco){
00431     this->endereco = endereco;
00432 }
00433
00434 inline Codigo Excursao::getCodigo() const{
00435     return codigo;
```

```
00436 }
00437
00438 inline Titulo Excursao::getTitulo() const{
00439     return titulo;
00440 }
00441
00442 inline Nota Excursao::getNota() const{
00443     return nota;
00444 }
00445
00446 inline Cidade Excursao::getCidade() const{
00447     return cidade;
00448 }
00449
00450 inline Duracao Excursao::getDuracao() const{
00451     return duracao;
00452 }
00453
00454 inline Descricao Excursao::getDescricao() const{
00455     return descricao;
00456 }
00457
00458 inline Endereco Excursao::getEndereco() const{
00459     return endereco;
00460 }
00461
00462
00463 #endif // ENTIDADES_H_INCLUDED
```

5.3 testes_dominios.h

```
00001 #ifndef TESTES_DOMINIOS_H_INCLUDED
00002 #define TESTES_DOMINIOS_H_INCLUDED
00003
00004 #include "dominios.h"
00005
00006 using namespace std;
00007
00008 // ------------------------------------------------------------------------
00009 // Exemplo de declara��o de classe para teste de unidade de classe dom�nio.
00010
00011 // class TUCodigo {
00012 // private:
00013 //     const static int VALOR_VALIDO   = 20;   // Defini��o de constante para evitar n�mero m�gico.
00014 //     const static int VALOR_INVALIDO = 30;   // Defini��o de constante para evitar n�mero m�gico.
00015 //     Codigo *codigo;                         // Refer�ncia para unidade em teste.
00016 //     int estado;                             // Estado do teste.
00017 //     void setUp();                           // M�todo para criar unidade em teste.
00018 //     void tearDown();                        // M�todo para destruir unidade em teste.
00019 //     void testarCenarioSucesso();            // Cen�rio de teste.
00020 //     void testarCenarioFalha();              // Cen�rio de teste.
00021
00022 // public:
00023 //     const static int SUCESSO =  0;          // Defini��o de constante para reportar resultado de
       teste.
00024 //     const static int FALHA   = -1;          // Defini��o de constante para reportar resultado de
       teste.
00025 //     int run();                              // M�todo para executar teste.
00026 // };
00027
00028
00029 class TUCidade {
00030 private:
00031     const string VALOR_VALIDO = "Paris";
00032     const string VALOR_INVALIDO = "Brasilia";
00033     Cidade *cidade;
00034     int estado;
00035     void setUp();
00036     void tearDown();
00037     void testarCenarioSucesso();
00038     void testarCenarioFalha();
00039
00040 public:
00041     const static int SUCESSO =  0;
00042     const static int FALHA   = -1;
00043     int run();
00044 };
00045
00046 class TUCodigo {
00047 private:
00048     const string VALOR_VALIDO = "5347164";
00049     const string VALOR_INVALIDO = "5347162";
00050     Codigo *codigo;
```

```
00051     int estado;
00052     void setUp();
00053     void tearDown();
00054     void testarCenarioSucesso();
00055     void testarCenarioFalha();
00056
00057 public:
00058     const static int SUCESSO =  0;
00059     const static int FALHA   = -1;
00060     int run();
00061 };
00062
00063 class TUData {
00064 private:
00065     const string VALOR_VALIDO = "04-Fev-2022";
00066     const string VALOR_INVALIDO = "30-Fev-2022";
00067     Data *data;
00068     int estado;
00069     void setUp();
00070     void tearDown();
00071     void testarCenarioSucesso();
00072     void testarCenarioFalha();
00073
00074 public:
00075     const static int SUCESSO =  0;
00076     const static int FALHA   = -1;
00077     int run();
00078 };
00079
00080 class TUDescricao {
00081 private:
00082     const string VALOR_VALIDO = "Esta e uma descricao valida.";
00083     const string VALOR_INVALIDO = "Esta  e uma descricao invalida..";
00084     Descricao *descricao;
00085     int estado;
00086     void setUp();
00087     void tearDown();
00088     void testarCenarioSucesso();
00089     void testarCenarioFalha();
00090
00091 public:
00092     const static int SUCESSO =  0;
00093     const static int FALHA   = -1;
00094     int run();
00095 };
00096
00097 class TUDuracao {
00098 private:
00099     const static int VALOR_VALIDO = 90;
00100     const static int VALOR_INVALIDO = 50;
00101     Duracao *duracao;
00102     int estado;
00103     void setUp();
00104     void tearDown();
00105     void testarCenarioSucesso();
00106     void testarCenarioFalha();
00107
00108 public:
00109     const static int SUCESSO =  0;
00110     const static int FALHA   = -1;
00111     int run();
00112 };
00113
00114 class TUEmail {
00115 private:
00116     const string VALOR_VALIDO = "dAvid.2!#$%&'*+-/=?^_`{|}~@gMail-3.com";
00117     const string VALOR_INVALIDO = ".dAvid..2!#$%&'*+-/=?^_`{|}~.@.gMail-3..com";
00118     Email *email;
00119     int estado;
00120     void setUp();
00121     void tearDown();
00122     void testarCenarioSucesso();
00123     void testarCenarioFalha();
00124
00125 public:
00126     const static int SUCESSO =  0;
00127     const static int FALHA   = -1;
00128     int run();
00129 };
00130
00131 class TUEndereco {
00132 private:
00133     const string VALOR_VALIDO = "Asa Norte.";
00134     const string VALOR_INVALIDO = "Asa  Norte..";
00135     Endereco *endereco;
00136     int estado;
00137     void setUp();
```

```
00138     void tearDown();
00139     void testarCenarioSucesso();
00140     void testarCenarioFalha();
00141
00142 public:
00143     const static int SUCESSO =  0;
00144     const static int FALHA   = -1;
00145     int run();
00146 };
00147
00148 class TUHorario {
00149 private:
00150     const string VALOR_VALIDO = "18:16";
00151     const string VALOR_INVALIDO = "24:60";
00152     Horario *horario;
00153     int estado;
00154     void setUp();
00155     void tearDown();
00156     void testarCenarioSucesso();
00157     void testarCenarioFalha();
00158
00159 public:
00160     const static int SUCESSO =  0;
00161     const static int FALHA   = -1;
00162     int run();
00163 };
00164
00165 class TUIdioma {
00166 private:
00167     const string VALOR_VALIDO = "Frances";
00168     const string VALOR_INVALIDO = "Hebraico";
00169     Idioma *idioma;
00170     int estado;
00171     void setUp();
00172     void tearDown();
00173     void testarCenarioSucesso();
00174     void testarCenarioFalha();
00175
00176 public:
00177     const static int SUCESSO =  0;
00178     const static int FALHA   = -1;
00179     int run();
00180 };
00181
00182 class TUNome {
00183 private:
00184     const string VALOR_VALIDO = "David. Chatelard.";
00185     const string VALOR_INVALIDO = "David..  chatelard.";
00186     Nome *nome;
00187     int estado;
00188     void setUp();
00189     void tearDown();
00190     void testarCenarioSucesso();
00191     void testarCenarioFalha();
00192
00193 public:
00194     const static int SUCESSO =  0;
00195     const static int FALHA   = -1;
00196     int run();
00197 };
00198
00199 class TUNota {
00200 private:
00201     const static int VALOR_VALIDO = 5;
00202     const static int VALOR_INVALIDO = -2;
00203     Nota *nota;
00204     int estado;
00205     void setUp();
00206     void tearDown();
00207     void testarCenarioSucesso();
00208     void testarCenarioFalha();
00209
00210 public:
00211     const static int SUCESSO =  0;
00212     const static int FALHA   = -1;
00213     int run();
00214 };
00215
00216 class TUSenha {
00217 private:
00218     const string VALOR_VALIDO = "Abc123";
00219     const string VALOR_INVALIDO = "aaa111";
00220     Senha *senha;
00221     int estado;
00222     void setUp();
00223     void tearDown();
00224     void testarCenarioSucesso();
```

```
00225     void testarCenarioFalha();
00226
00227 public:
00228     const static int SUCESSO =  0;
00229     const static int FALHA   = -1;
00230     int run();
00231 };
00232
00233 class TUTitulo {
00234 private:
00235     const string VALOR_VALIDO = "Teste de Unidade.";
00236     const string VALOR_INVALIDO = "Teste de  Unidade..";
00237     Titulo *titulo;
00238     int estado;
00239     void setUp();
00240     void tearDown();
00241     void testarCenarioSucesso();
00242     void testarCenarioFalha();
00243
00244 public:
00245     const static int SUCESSO =  0;
00246     const static int FALHA   = -1;
00247     int run();
00248 };
00249
00250
00251 #endif // TESTES_DOMINIOS_H_INCLUDED
```

5.4 testes_entidades.h

```
00001 #ifndef TESTES_ENTIDADES_H_INCLUDED
00002 #define TESTES_ENTIDADES_H_INCLUDED
00003
00004 #include "entidades.h"
00005
00006 #include <string>
00007
00008 using namespace std;
00009
00010 /*-------------------------------------------------------------------------------------------------
00011
00012 // ---------------------------------------------------------------------
00013 // Exemplo de declara��o de classe para teste de unidade de classe entidade.
00014 // Tem que adicionar as linhas 16 e 17 para cada atributo da entidade a ser testada(por exemplo, no
       caso do usuario teria que ter pro nome, email e senha)
00015
00016 class TUProjeto {
00017 private:
00018     const static int VALOR_VALIDO   = 20;   // Defini��o de constante para evitar n�mero m�gico.
00019     Projeto *projeto;                       // Refer�ncia para unidade em teste.
00020     int estado;                             // Estado do teste.
00021     void setUp();                           // M�todo para criar unidade em teste.
00022     void tearDown();                        // M�todo para destruir unidade em teste.
00023     void testarCenarioSucesso();            // Cen�rio de teste.
00024 public:
00025     const static int SUCESSO =  0;          // Defini��o de constante para reportar resultado de
       teste.
00026     const static int FALHA   = -1;          // Defini��o de constante para reportar resultado de
       teste.
00027     int run();                              // M�todo para executar teste.
00028 };
00029
00030 -------------------------------------------------------------------------------------------------*/
00031
00032 class TUUsuario {
00033 private:
00034     const string VALOR_VALIDO_NOME    = "Alexandre Cunha.";
00035     const string VALOR_VALIDO_EMAIL   = "Email_Teste.123@gmail.com";
00036     const string VALOR_VALIDO_SENHA   = "BfA843";
00037     Usuario *usuario;
00038     int estado;
00039     void setUp();
00040     void tearDown();
00041     void testarCenarioSucesso();
00042 public:
00043     const static int SUCESSO =  0;
00044     const static int FALHA   = -1;
00045     int run();
00046 };
00047
00048 class TUAvaliacao {
00049 private:
00050     const string VALOR_VALIDO_CODIGO    = "5347164";
```

```
00051     const static int VALOR_VALIDO_NOTA   = 0;
00052     const string VALOR_VALIDO_DESCRICAO = "O aluno foi pessimo";
00053     Avaliacao *avaliacao;
00054     int estado;
00055     void setUp();
00056     void tearDown();
00057     void testarCenarioSucesso();
00058 public:
00059     const static int SUCESSO =  0;
00060     const static int FALHA   = -1;
00061     int run();
00062 };
00063
00064 class TUSessao {
00065 private:
00066     const string VALOR_VALIDO_CODIGO    = "5347164";
00067     const string VALOR_VALIDO_DATA      = "12-Out-2027";
00068     const string VALOR_VALIDO_HORARIO   = "00:00";
00069     const string VALOR_VALIDO_IDIOMA    = "Mandarim";
00070     Sessao *sessao;
00071     int estado;
00072     void setUp();
00073     void tearDown();
00074     void testarCenarioSucesso();
00075 public:
00076     const static int SUCESSO =  0;
00077     const static int FALHA   = -1;
00078     int run();
00079 };
00080
00081 class TUExcursao {
00082 private:
00083     const string VALOR_VALIDO_CODIGO      = "5347164";
00084     const string VALOR_VALIDO_TITULO      = "Trilha Chinesa";
00085     const static int VALOR_VALIDO_NOTA     = 3;
00086     const string VALOR_VALIDO_CIDADE      = "Bangkok";
00087     const static int VALOR_VALIDO_DURACAO  = 180;
00088     const string VALOR_VALIDO_DESCRICAO   = "Venha descobrir a China";
00089     const string VALOR_VALIDO_ENDERECO    = "A ser definido";
00090     Excursao *excursao;
00091     int estado;
00092     void setUp();
00093     void tearDown();
00094     void testarCenarioSucesso();
00095 public:
00096     const static int SUCESSO =  0;
00097     const static int FALHA   = -1;
00098     int run();
00099 };
00100
00101 #endif // TESTES_ENTIDADES_H_INCLUDED
```

5.5 dominios.cpp

```
00001 #include "../headers/dominios.h"
00002
00003 #include <algorithm>
00004 #include <set>
00005 #include <iostream>
00006 #include <fstream>
00007 #include <string>
00008 #include <iterator> // ver se precisa, pode ser util
00009 #include <cctype>
00010
00011 using namespace std;
00012
00013 /* ------------------------------------------------------------------------
00014 // Estrutura de c◆digo para implementacao de classe dom◆nio.
00015 // Substituir Dominio por nome da classe.
00016 // Substituir Tipo.
00017 // Implementar validacao
00018
00019 void Dominio::validar(Tipo valor){
00020     if (valor == INVALIDO)
00021         throw invalid_argument("Argumento invalido.");
00022 }
00023
00024 void Dominio::setValor(Tipo valor) {
00025     validar(valor);
00026     this->valor = valor;
00027 }
00028 ------------------------------------------------------------------------ */
00029
```

```
00030 // Sets
00031
00032 set<string> cidades {"Hong Kong", "Bangkok", "Macau", "Singapura", "Londres", "Paris", "Dubai",
       "Delhi", "Istambul", "Kuala Lumpur", "Nova Iorque", "Antalya", "Mumbai", "Shenzhen", "Phuket"};
00033
00034 set<string> meses {"Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out", "Nov",
       "Dez"};
00035 set<string> mes_31 {"Jan", "Mar", "Mai", "Jul", "Ago", "Out", "Dez"};
00036
00037 set<char> caracteres_parte_local {'!', '#', '$', '%', '&', '\", '*', '+', '-', '/', '=', '?', '^',
       '_', '`', '{', '}', '|', '~'};
00038
00039 set<string> idiomas {"Ingles", "Chines", "Mandarim", "Hindi", "Espanhol", "Frances", "Arabe",
       "Bengali", "Russo", "Portugues", "Indonesio"};
00040
00041 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00042 void Cidade::validar(string valor){
00043     if (!(cidades.find(valor) != cidades.end()))
00044       throw invalid_argument("Argumento invalido.");
00045 }
00046
00047 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00048 void Cidade::setValor(string valor) {
00049     validar(valor);
00050     this->valor = valor;
00051 }
00052
00053 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00054 void Codigo::validar(string valor){
00055     // O algoritmo de verificação utilizado será o UPC(https://stringfixer.com/pt/Check_digit)
00056     const int tam_codigo = 7;
00057     string aux = valor;
00058
00059     if (valor.length() != tam_codigo) {
00060         throw invalid_argument("Argumento invalido.");
00061     }
00062     aux.resize(aux.size() - 1);
00063     int soma_impar = 3 * ((valor[0]-'0') + (valor[2]-'0') + (valor[4]-'0'));
00064     int soma_par = (valor[1]-'0') + (valor[3]-'0') + (valor[5]-'0');
00065     int digito_verificacao = (soma_impar + soma_par) % 10;
00066
00067     if (digito_verificacao != 0) {
00068         digito_verificacao = 10 - digito_verificacao;
00069     }
00070
00071     // Verificar se a condição do str.compare() está funcionando, pois ela retorna 0, tem que ver se
       c++ aceita 0 e 1 como booleano
00072     if (((valor[6]-'0') != digito_verificacao) || (!(aux.compare("000000"))))
00073         throw invalid_argument("Argumento invalido.");
00074 }
00075
00076 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00077 void Codigo::setValor(string valor) {
00078     validar(valor);
00079     this->valor = valor;
00080 }
00081
00082 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00083 void Data::validar(string valor){
00084     string separa_data[3] = {"","",""};
00085     string buffer = "";
00086     int i = 0;
00087     bool is_bissexto;
00088
00089     // Divide a data em 3 strings (dia, mês e ano)
00090     for (auto eachchar:valor) {
00091         if (eachchar != '-') {
00092             buffer+=eachchar;
00093         }
00094         else if (eachchar == '-') {
00095             separa_data[i] = buffer;
00096             buffer = "";
00097             i++;
00098         }
00099     }
00100     separa_data[i] = buffer;
00101
00102     //Checa se o ano é bissexto
00103     if (stoi(separa_data[2]) % 4 == 0) {
00104         if (stoi(separa_data[2]) % 100 == 0) {
00105             if (stoi(separa_data[2]) % 400 == 0) {
00106                 is_bissexto = true;
00107             }
00108             else {
00109                 is_bissexto = false;
00110             }
00111         }
```

```
00112         else {
00113             is_bissexto = true;
00114         }
00115     }
00116     else {
00117         is_bissexto = false;
00118     }
00119
00120     //Testa DD
00121     if (stoi(separa_data[0]) < 1 || stoi(separa_data[0]) > 31) {
00122         throw invalid_argument("Argumento invalido.");
00123     }
00124
00125     //Testa Mes 30 ou 31
00126     else if (stoi(separa_data[0]) == 31) {
00127         if(!(mes_31.find(separa_data[1]) != mes_31.end())) {
00128             throw invalid_argument("Argumento invalido.");
00129         }
00130     }
00131
00132     //Testa MES
00133     else if (!(meses.find(separa_data[1]) != meses.end()))
00134         throw invalid_argument("Argumento invalido.");
00135
00136     //Testa AA
00137     else if (stoi(separa_data[2]) < 2000 || stoi(separa_data[2]) > 9999 )
00138         throw invalid_argument("Argumento invalido.");
00139
00140     //Testa Fevereiro e BISSEXTO
00141     else if (separa_data[1] == "Fev" && stoi(separa_data[0]) > 28) {
00142         if(stoi(separa_data[0]) > 29) {
00143             throw invalid_argument("Argumento invalido.");
00144         }
00145         //TESTA BISSEXTO
00146         //Se dia for 29 e não for bissexto, aponta erro
00147         else if(stoi(separa_data[0]) == 29 && !is_bissexto) {
00148             throw invalid_argument("Argumento invalido.");
00149         }
00150     }
00151 }
00152
00153 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00154 void Data::setValor(string valor) {
00155     validar(valor);
00156     this->valor = valor;
00157 }
00158
00159 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00160 void Descricao::validar(string valor){
00161     int num_pontos = 0;
00162     int num_espacos = 0;
00163     int tam_descricao = int(valor.length());
00164
00165     for (auto letra:valor) {
00166         if (num_pontos >= 2) {
00167             break;
00168         }
00169         else if (num_espacos >= 2) {
00170             break;
00171         }
00172
00173         if (letra == '.') {
00174             num_pontos++;
00175             num_espacos = 0;
00176         }
00177         else if (letra == ' ') {
00178             num_espacos++;
00179             num_pontos = 0;
00180         }
00181         else {
00182             num_pontos = 0;
00183             num_espacos = 0;
00184         }
00185     }
00186
00187     if ((tam_descricao > 30) || (tam_descricao < 0) || (num_pontos >= 2) || (num_espacos >= 2))
00188         throw invalid_argument("Argumento invalido.");
00189 }
00190
00191 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00192 void Descricao::setValor(string valor) {
00193     validar(valor);
00194     this->valor = valor;
00195 }
00196
00197 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00198 void Duracao::validar(int valor){
```

```
00199     if ((valor != 30) && (valor != 60) && (valor != 90) && (valor != 120) && (valor != 180))
00200         throw invalid_argument("Argumento invalido.");
00201 }
00202
00203 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00204 void Duracao::setValor(int valor) {
00205     validar(valor);
00206     this->valor = valor;
00207 }
00208
00209 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00210 void Email::validar(string valor){
00211     bool letra_invalida_local = false;
00212     bool letra_invalida_dominio = false;
00213     int num_pontos = 0;
00214     string separa_email[2] = {"","};
00215     string buffer;
00216     int i = 0;
00217
00218     // Divide o email em 2 strings (parte-local e dominio)
00219     for (auto letra:valor) {
00220         if(letra != '@') {
00221             buffer+=letra;
00222         }
00223         else if (letra == '@') {
00224             separa_email[i] = buffer;
00225             buffer = "";
00226             i++;
00227         }
00228     }
00229     separa_email[i] = buffer;
00230
00231     // Verifica a parte-local
00232     for (auto letra:separa_email[0]) {
00233         if (!isalnum(letra) && !(caracteres_parte_local.find(letra) != caracteres_parte_local.end()))
    { //Não é letra e nem número e nem caractere especial permitido. Ainda pode ser '.' ou '@'
00234             if (letra == '@') {
00235                 letra_invalida_local = true;
00236                 break;
00237             }
00238             else if (letra == '.') {
00239                 num_pontos++;
00240                 if (num_pontos >= 2) {
00241                     letra_invalida_local = true;
00242                     break;
00243                 }
00244             }
00245         }
00246         else {
00247             num_pontos = 0;
00248         }
00249     }
00250
00251     // Verifica o dominio
00252     num_pontos = 0;
00253     for (auto letra:separa_email[1]) {
00254         if (letra_invalida_local) {  // A parte-local ja eh invalida
00255             break;
00256         }
00257         if (!isalnum(letra) && !(letra == '-')) { // Nao eh letra e nem '-'. Ainda pode ser '.' ou
    caracter invalido
00258             if (letra == '.') {
00259                 num_pontos++;
00260                 if (num_pontos >= 2) {
00261                     letra_invalida_dominio = true;
00262                     break;
00263                 }
00264             }
00265             else {
00266                 letra_invalida_dominio = true;
00267                 break;
00268             }
00269         }
00270         else {
00271             num_pontos = 0;
00272         }
00273     }
00274
00275     if ((separa_email[0].length() > 64) || (separa_email[0][0] == '.') ||
    (separa_email[0][separa_email[0].length()-1] == '.') || (letra_invalida_local) ||
    (separa_email[1].length() > 253) || (separa_email[1][0] == '.') || (letra_invalida_dominio))
00276         throw invalid_argument("Argumento invalido.");
00277 }
00278
00279 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00280 void Email::setValor(string valor) {
00281     validar(valor);
```

```
00282     this->valor = valor;
00283 }
00284
00285 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00286 void Endereco::validar(string valor){
00287     string buffer = "";
00288     bool is_dot, erro_dot= false;
00289     bool is_space, erro_space = false;
00290
00291     for (auto eachchar:valor) {
00292             if (eachchar == ' ') {
00293                 if(is_space == true) {
00294                     erro_space = true;
00295                 }
00296                 else {
00297                     is_space = true;
00298                     buffer+=eachchar;
00299                 }
00300             }
00301             else if (eachchar == '.') {
00302                 if (is_dot == true) {
00303                     erro_dot = true;
00304                 }
00305                 else {
00306                     is_dot = true;
00307                     buffer+=eachchar;
00308                 }
00309             }
00310             else {
00311                 is_dot = false;
00312                 is_space = false;
00313                 buffer += eachchar;
00314             }
00315         }
00316         if(erro_dot == true || erro_space == true || buffer.size() > 20)
00317             throw invalid_argument("Argumento invalido.");
00318 }
00319
00320 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00321 void Endereco::setValor(string valor){
00322     validar(valor);
00323     this->valor = valor;
00324 }
00325
00326 //VERIFICAR A NECESSIDADE DE CONFERIR  SE A ENTRADA É DE FATO DADA POR NÚMEROS (PODE DAR PROBLEMA POR
      CAUSA DO STOI)
00327 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00328 void Horario::validar(string valor){
00329     string separa_hora[2] = {"",""};
00330     string buffer = "";
00331     int i = 0;
00332
00333     // Divide a hora em 2 strings (hora e minutos)
00334     for (auto eachchar:valor) {
00335         if (eachchar != ':') {
00336             buffer+=eachchar;
00337         }
00338         else if (eachchar == ':') {
00339             separa_hora[i] = buffer;
00340             buffer = "";
00341             i++;
00342         }
00343     }
00344     separa_hora[i] = buffer;
00345
00346     if (stoi(separa_hora[0]) < 0 || stoi(separa_hora[0]) > 23) {
00347         throw invalid_argument("Argumento invalido.");
00348     }
00349     else if (stoi(separa_hora[1]) < 0 || stoi(separa_hora[1]) > 59) {
00350         throw invalid_argument("Argumento invalido.");
00351     }
00352 }
00353
00354 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00355 void Horario::setValor(string valor){
00356     validar(valor);
00357     this->valor = valor;
00358 }
00359
00360 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00361 void Idioma::validar(string valor){
00362     if(!(idiomas.find(valor) != idiomas.end()))
00363         throw invalid_argument("Argumento invalido.");
00364 }
00365
00366 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00367 void Idioma::setValor(string valor){
```

```
00368     validar(valor);
00369     this->valor = valor;
00370 }
00371
00372 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00373 void Nome::validar(string valor){
00374     int tam_nome = int(valor.length());
00375
00376     for (int i = 0; i < tam_nome; i++) {
00377
00378         //Verifica se é letra, espaço em branco ou ponto
00379         if (!isalpha(valor[i]) && !isblank(valor[i]) && valor[i] != '.')
00380             throw invalid_argument("Argumento invalido.");
00381
00382         //Verifica se o ponto é precedido por letra e seguido por um espaço em branco caso não seja o último caractere
00383         if (valor[i] == '.' && i == 0) {
00384             throw invalid_argument("Argumento invalido.");
00385         }
00386         else if (valor[i] == '.' && i < (tam_nome - 1)) {
00387             if (!isalpha(valor[i-1]) || !isblank(valor[i+1]))
00388                 throw invalid_argument("Argumento invalido.");
00389         }
00390         else if (valor[i] == '.' && i == (tam_nome - 1)) {
00391             if (!isalpha(valor[i-1]))
00392                 throw invalid_argument("Argumento invalido.");
00393         }
00394
00395         //Verifica se não há espaços em branco em sequência e se a primeira letra de cada termo é letra maiúscula
00396         if (valor[i] == ' ' && i == 0) {
00397             if (!isupper(valor[i + 1]))
00398                 throw invalid_argument("Argumento invalido.");
00399         }
00400         else if (valor[i] == ' ' && i < (tam_nome - 1)) {
00401             if (isblank(valor[i-1]) || isblank(valor[i+1]) || !isupper(valor[i+1]))
00402                 throw invalid_argument("Argumento invalido.");
00403         }
00404         else if (valor[i] == ' ' && i == (tam_nome - 1)) {
00405             if (isblank(valor[i-1]))
00406                 throw invalid_argument("Argumento invalido.");
00407         }
00408     }
00409     //Verifica se o nome possui entre 5 e 20 caracteres
00410     if (tam_nome < 5 || tam_nome > 20)
00411         throw invalid_argument("Argumento invalido.");
00412 }
00413
00414 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00415 void Nome::setValor(string valor){
00416     validar(valor);
00417     this->valor = valor;
00418 }
00419
00420 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00421 void Nota::validar(int valor){
00422     if(valor > 5 || valor < 0)
00423         throw invalid_argument("Argumento invalido.");
00424 }
00425
00426 // Método implementado pelo aluno David Fanchic Chatelard, matrícula 180138863
00427 void Nota::setValor(int valor){
00428     validar(valor);
00429     this->valor = valor;
00430 }
00431
00432 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00433 void Senha::validar(string valor){
00434     //Lenght da string é de 6
00435     const int tam_string = 6;
00436     string senha = "";
00437     bool has_lowercase = false, has_uppercase = false, has_digit = false, has_repeated_char = false, has_invalid_char = false;
00438     for(auto eachchar:valor) {
00439         if (senha.find(eachchar) != string::npos) {
00440             has_repeated_char = true;
00441         }
00442         if (!isalnum(eachchar)) {
00443             has_invalid_char= true;
00444             senha+=eachchar;
00445         }
00446         //Checa se tem pelo menos um digito
00447         if (isdigit(eachchar)) {
00448             has_digit = true;
00449             senha+=eachchar;
00450         }
00451         //Checa se tem pelo menos uma letra maiuscula
```

```
00452         else if (isupper(eachchar)) {
00453             has_uppercase = true;
00454             senha+=eachchar;
00455         }
00456         //Checa se tem pelo menos uma letra minuscula
00457         else if (islower(eachchar)) {
00458             has_lowercase = true;
00459             senha+=eachchar;
00460         }
00461     }
00462     if (valor.length() != tam_string || !has_lowercase || !has_uppercase || !has_digit ||
      has_repeated_char || has_invalid_char) {
00463         throw invalid_argument("Argumento invalido.");
00464     }
00465 }
00466
00467 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00468 void Senha::setValor(string valor){
00469     validar(valor);
00470     this->valor = valor;
00471 }
00472
00473 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00474 void Titulo::validar(string valor){
00475     int num_letras = 0, tam_titulo = int(valor.length());
00476
00477     for (int i = 0; i < tam_titulo; i++) {
00478
00479         //Conta o número de letras, já que pode haver outros caracteres
00480         if (isalpha(valor[i]))
00481             num_letras++;
00482
00483         //Verifica se não há pontos em sequência
00484         if (valor[i] == '.' && i != 0) {
00485             if (valor[i-1] == '.')
00486                 throw invalid_argument("Argumento invalido.");
00487         }
00488
00489         //Verifica se não há espaços em branco em sequência
00490         if (valor[i] == ' ' && i != 0) {
00491             if (isblank(valor[i-1]))
00492                 throw invalid_argument("Argumento invalido.");
00493         }
00494     }
00495
00496     //Verifica se o título possui entre 5 e 20 letras
00497     if (num_letras < 5 || num_letras > 20)
00498         throw invalid_argument("Argumento invalido.");
00499
00500 }
00501
00502 // Método implementado pelo aluno Alexandre Abrahami Pinto da Cunha, matrícula 180041169
00503 void Titulo::setValor(string valor) {
00504     validar(valor);
00505     this->valor = valor;
00506 }
```

5.6 entidades.cpp

```
00001 #include "../headers/entidades.h"
00002
00003
```

5.7 main.cpp

```
00001 #include <iostream>
00002
00003 #include "../headers/dominios.h"
00004 #include "dominios.cpp"
00005
00006 #include "../headers/entidades.h"
00007 #include "entidades.cpp"
00008
00009 #include "../headers/testes_dominios.h"
00010 #include "testes_dominios.cpp"
00011
00012 #include "../headers/testes_entidades.h"
00013 #include "testes_entidades.cpp"
00014
```

```
00015 using namespace std;
00016
00017 int main(int argc, char const *argv[]){
00018
00019     // Endereco endereco1;
00020
00021     // endereco1.setValor("Asa  Norte..");
00022
00023     // cout « endereco1.getValor() « endl;
00024
00025
00026     /*-----------------------------------------------------------------------------------------------
00027
00028     //-------------------------------------------------------------------
00029     // Exemplo de teste de classe dom•nio.
00030
00031     // Instanciar classe de teste de dom•nio.
00032
00033     TUCodigo testeA;
00034
00035     // Invocar m•todo e apresentar mensagem acerca do resultado do teste.
00036
00037     switch(testeA.run()){
00038         case TUCodigo::SUCESSO: cout « "SUCESSO - CODIGO" « endl;
00039                                 break;
00040         case TUCodigo::FALHA  : cout « "FALHA   - CODIGO" « endl;
00041                                 break;
00042     }
00043
00044     -----------------------------------------------------------------------------------------------*/
00045
00046     TUCidade cidade_teste;
00047
00048     switch(cidade_teste.run()){
00049         case TUCidade::SUCESSO: cout « "SUCESSO - CIDADE" « endl;
00050                                 break;
00051         case TUCidade::FALHA  : cout « "FALHA   - CIDADE" « endl;
00052                                 break;
00053     }
00054
00055
00056     TUCodigo codigo_teste;
00057
00058     switch(codigo_teste.run()){
00059         case TUCodigo::SUCESSO: cout « "SUCESSO - CODIGO" « endl;
00060                                 break;
00061         case TUCodigo::FALHA  : cout « "FALHA   - CODIGO" « endl;
00062                                 break;
00063     }
00064
00065
00066     TUData data_teste;
00067
00068     switch(data_teste.run()){
00069         case TUData::SUCESSO: cout « "SUCESSO - DATA" « endl;
00070                               break;
00071         case TUData::FALHA  : cout « "FALHA   - DATA" « endl;
00072                               break;
00073     }
00074
00075
00076     TUDescricao descricao_teste;
00077
00078     switch(descricao_teste.run()){
00079         case TUDescricao::SUCESSO: cout « "SUCESSO - DESCRICAO" « endl;
00080                                    break;
00081         case TUDescricao::FALHA  : cout « "FALHA   - DESCRICAO" « endl;
00082                                    break;
00083     }
00084
00085
00086     TUDuracao duracao_teste;
00087
00088     switch(duracao_teste.run()){
00089         case TUDuracao::SUCESSO: cout « "SUCESSO - DURACAO" « endl;
00090                                  break;
00091         case TUDuracao::FALHA  : cout « "FALHA   - DURACAO" « endl;
00092                                  break;
00093     }
00094
00095
00096     TUEmail email_teste;
00097
00098     switch(email_teste.run()){
00099         case TUEmail::SUCESSO: cout « "SUCESSO - EMAIL" « endl;
00100                                break;
00101         case TUEmail::FALHA  : cout « "FALHA   - EMAIL" « endl;
```

```
00102                               break;
00103     }
00104
00105
00106     TUEndereco endereco_teste;
00107
00108     switch(endereco_teste.run()){
00109         case TUEndereco::SUCESSO: cout « "SUCESSO - ENDERECO" « endl;
00110                               break;
00111         case TUEndereco::FALHA  : cout « "FALHA   - ENDERECO" « endl;
00112                               break;
00113     }
00114
00115
00116     TUHorario horario_teste;
00117
00118     switch(horario_teste.run()){
00119         case TUHorario::SUCESSO: cout « "SUCESSO - HORARIO" « endl;
00120                               break;
00121         case TUHorario::FALHA  : cout « "FALHA   - HORARIO" « endl;
00122                               break;
00123     }
00124
00125
00126     TUIdioma idioma_teste;
00127
00128     switch(idioma_teste.run()){
00129         case TUIdioma::SUCESSO: cout « "SUCESSO - IDIOMA" « endl;
00130                               break;
00131         case TUIdioma::FALHA  : cout « "FALHA   - IDIOMA" « endl;
00132                               break;
00133     }
00134
00135
00136     TUNome nome_teste;
00137
00138     switch(nome_teste.run()){
00139         case TUNome::SUCESSO: cout « "SUCESSO - NOME" « endl;
00140                               break;
00141         case TUNome::FALHA  : cout « "FALHA   - NOME" « endl;
00142                               break;
00143     }
00144
00145
00146     TUNota nota_teste;
00147
00148     switch(nota_teste.run()){
00149         case TUNota::SUCESSO: cout « "SUCESSO - NOTA" « endl;
00150                               break;
00151         case TUNota::FALHA  : cout « "FALHA   - NOTA" « endl;
00152                               break;
00153     }
00154
00155
00156     TUSenha senha_teste;
00157
00158     switch(senha_teste.run()){
00159         case TUSenha::SUCESSO: cout « "SUCESSO - SENHA" « endl;
00160                               break;
00161         case TUSenha::FALHA  : cout « "FALHA   - SENHA" « endl;
00162                               break;
00163     }
00164
00165
00166     TUTitulo titulo_teste;
00167
00168     switch(titulo_teste.run()){
00169         case TUTitulo::SUCESSO: cout « "SUCESSO - TITULO" « endl;
00170                               break;
00171         case TUTitulo::FALHA  : cout « "FALHA   - TITULO" « endl;
00172                               break;
00173     }
00174
00175
00176
00177     /*-------------------------------------------------------------------------------------------------
00178
00179     //---------------------------------------------------------------------
00180     // Exemplo de teste de classe entidade.
00181
00182     // Instanciar classe de teste de entidade.
00183
00184     TUProjeto testeB;
00185
00186     // Invocar m◆todo e apresentar mensagem acerca do resultado do teste.
00187
00188     switch(testeB.run()){
```

```
00189         case TUProjeto::SUCESSO: cout « "SUCESSO - PROJETO" « endl;
00190                                  break;
00191         case TUProjeto::FALHA  : cout « "FALHA   - PROJETO" « endl;
00192                                  break;
00193     }
00194
00195     ----------------------------------------------------------------------------------------------*/
00196
00197     TUUsuario usuario_teste;
00198
00199     switch(usuario_teste.run()){
00200         case TUUsuario::SUCESSO: cout « "SUCESSO - USUARIO" « endl;
00201                                  break;
00202         case TUUsuario::FALHA  : cout « "FALHA   - USUARIO" « endl;
00203                                  break;
00204     }
00205
00206     TUAvaliacao avaliacao_teste;
00207
00208     switch(avaliacao_teste.run()){
00209         case TUAvaliacao::SUCESSO: cout « "SUCESSO - AVALIACAO" « endl;
00210                                  break;
00211         case TUAvaliacao::FALHA  : cout « "FALHA   - AVALIACAO" « endl;
00212                                  break;
00213     }
00214
00215     TUSessao sessao_teste;
00216
00217     switch(sessao_teste.run()){
00218         case TUSessao::SUCESSO: cout « "SUCESSO - SESSAO" « endl;
00219                                  break;
00220         case TUSessao::FALHA  : cout « "FALHA   - SESSAO" « endl;
00221                                  break;
00222     }
00223
00224     TUExcursao excursao_teste;
00225
00226     switch(excursao_teste.run()){
00227         case TUExcursao::SUCESSO: cout « "SUCESSO - EXCURSAO" « endl;
00228                                  break;
00229         case TUExcursao::FALHA  : cout « "FALHA   - EXCURSAO" « endl;
00230                                  break;
00231     }
00232
00233     return 0;
00234 }
```

5.8 testes_dominios.cpp

```
00001 #include "../headers/testes_dominios.h"
00002
00003
00004 /*-----------------------------------------------------------------------------------------------
00005
00006 // ---------------------------------------------------------------------
00007 // Implementa��es de m�todos de classe de teste de unidade.
00008
00009 void TUCodigo::setUp(){
00010     codigo = new Codigo();
00011     estado = SUCESSO;
00012 }
00013
00014 void TUCodigo::tearDown(){
00015     delete codigo;
00016 }
00017
00018 void TUCodigo::testarCenarioSucesso(){
00019     try{
00020         codigo->setValor(VALOR_VALIDO);
00021         if (codigo->getValor() != VALOR_VALIDO)
00022             estado = FALHA;
00023     }
00024     catch(invalid_argument &excecao){
00025         estado = FALHA;
00026     }
00027 }
00028
00029 void TUCodigo::testarCenarioFalha(){
00030     try{
00031         codigo->setValor(VALOR_INVALIDO);
00032         estado = FALHA;
00033     }
00034     catch(invalid_argument &excecao){
```

```
00035         if (codigo->getValor() == VALOR_INVALIDO)
00036             estado = FALHA;
00037     }
00038 }
00039
00040 int TUCodigo::run(){
00041     setUp();
00042     testarCenarioSucesso();
00043     testarCenarioFalha();
00044     tearDown();
00045     return estado;
00046 }
00047
00048 ----------------------------------------------------------------------------------------------------*/
00049
00050
00051 void TUCidade::setUp(){
00052     cidade = new Cidade();
00053     estado = SUCESSO;
00054 }
00055
00056 void TUCidade::tearDown(){
00057     delete cidade;
00058 }
00059
00060 void TUCidade::testarCenarioSucesso(){
00061     try{
00062         cidade->setValor(VALOR_VALIDO);
00063         if (cidade->getValor() != VALOR_VALIDO)
00064             estado = FALHA;
00065     }
00066     catch(invalid_argument &excecao){
00067         estado = FALHA;
00068     }
00069 }
00070
00071 void TUCidade::testarCenarioFalha(){
00072     try{
00073         cidade->setValor(VALOR_INVALIDO);
00074         estado = FALHA;
00075     }
00076     catch(invalid_argument &excecao){
00077         if (cidade->getValor() == VALOR_INVALIDO)
00078             estado = FALHA;
00079     }
00080 }
00081
00082 int TUCidade::run(){
00083     setUp();
00084     testarCenarioSucesso();
00085     testarCenarioFalha();
00086     tearDown();
00087     return estado;
00088 }
00089
00090
00091 void TUCodigo::setUp(){
00092     codigo = new Codigo();
00093     estado = SUCESSO;
00094 }
00095
00096 void TUCodigo::tearDown(){
00097     delete codigo;
00098 }
00099
00100 void TUCodigo::testarCenarioSucesso(){
00101     try{
00102         codigo->setValor(VALOR_VALIDO);
00103         if (codigo->getValor() != VALOR_VALIDO)
00104             estado = FALHA;
00105     }
00106     catch(invalid_argument &excecao){
00107         estado = FALHA;
00108     }
00109 }
00110
00111 void TUCodigo::testarCenarioFalha(){
00112     try{
00113         codigo->setValor(VALOR_INVALIDO);
00114         estado = FALHA;
00115     }
00116     catch(invalid_argument &excecao){
00117         if (codigo->getValor() == VALOR_INVALIDO)
00118             estado = FALHA;
00119     }
00120 }
00121
```

```
00122 int TUCodigo::run(){
00123     setUp();
00124     testarCenarioSucesso();
00125     testarCenarioFalha();
00126     tearDown();
00127     return estado;
00128 }
00129
00130
00131 void TUData::setUp(){
00132     data = new Data();
00133     estado = SUCESSO;
00134 }
00135
00136 void TUData::tearDown(){
00137     delete data;
00138 }
00139
00140 void TUData::testarCenarioSucesso(){
00141     try{
00142         data->setValor(VALOR_VALIDO);
00143         if (data->getValor() != VALOR_VALIDO)
00144             estado = FALHA;
00145     }
00146     catch(invalid_argument &excecao){
00147         estado = FALHA;
00148     }
00149 }
00150
00151 void TUData::testarCenarioFalha(){
00152     try{
00153         data->setValor(VALOR_INVALIDO);
00154         estado = FALHA;
00155     }
00156     catch(invalid_argument &excecao){
00157         if (data->getValor() == VALOR_INVALIDO)
00158             estado = FALHA;
00159     }
00160 }
00161
00162 int TUData::run(){
00163     setUp();
00164     testarCenarioSucesso();
00165     testarCenarioFalha();
00166     tearDown();
00167     return estado;
00168 }
00169
00170
00171 void TUDescricao::setUp(){
00172     descricao = new Descricao();
00173     estado = SUCESSO;
00174 }
00175
00176 void TUDescricao::tearDown(){
00177     delete descricao;
00178 }
00179
00180 void TUDescricao::testarCenarioSucesso(){
00181     try{
00182         descricao->setValor(VALOR_VALIDO);
00183         if (descricao->getValor() != VALOR_VALIDO)
00184             estado = FALHA;
00185     }
00186     catch(invalid_argument &excecao){
00187         estado = FALHA;
00188     }
00189 }
00190
00191 void TUDescricao::testarCenarioFalha(){
00192     try{
00193         descricao->setValor(VALOR_INVALIDO);
00194         estado = FALHA;
00195     }
00196     catch(invalid_argument &excecao){
00197         if (descricao->getValor() == VALOR_INVALIDO)
00198             estado = FALHA;
00199     }
00200 }
00201
00202 int TUDescricao::run(){
00203     setUp();
00204     testarCenarioSucesso();
00205     testarCenarioFalha();
00206     tearDown();
00207     return estado;
00208 }
```

```
00209
00210
00211 void TUDuracao::setUp(){
00212     duracao = new Duracao();
00213     estado = SUCESSO;
00214 }
00215
00216 void TUDuracao::tearDown(){
00217     delete duracao;
00218 }
00219
00220 void TUDuracao::testarCenarioSucesso(){
00221     try{
00222         duracao->setValor(VALOR_VALIDO);
00223         if (duracao->getValor() != VALOR_VALIDO)
00224             estado = FALHA;
00225     }
00226     catch(invalid_argument &excecao){
00227         estado = FALHA;
00228     }
00229 }
00230
00231 void TUDuracao::testarCenarioFalha(){
00232     try{
00233         duracao->setValor(VALOR_INVALIDO);
00234         estado = FALHA;
00235     }
00236     catch(invalid_argument &excecao){
00237         if (duracao->getValor() == VALOR_INVALIDO)
00238             estado = FALHA;
00239     }
00240 }
00241
00242 int TUDuracao::run(){
00243     setUp();
00244     testarCenarioSucesso();
00245     testarCenarioFalha();
00246     tearDown();
00247     return estado;
00248 }
00249
00250
00251 void TUEmail::setUp(){
00252     email = new Email();
00253     estado = SUCESSO;
00254 }
00255
00256 void TUEmail::tearDown(){
00257     delete email;
00258 }
00259
00260 void TUEmail::testarCenarioSucesso(){
00261     try{
00262         email->setValor(VALOR_VALIDO);
00263         if (email->getValor() != VALOR_VALIDO)
00264             estado = FALHA;
00265     }
00266     catch(invalid_argument &excecao){
00267         estado = FALHA;
00268     }
00269 }
00270
00271 void TUEmail::testarCenarioFalha(){
00272     try{
00273         email->setValor(VALOR_INVALIDO);
00274         estado = FALHA;
00275     }
00276     catch(invalid_argument &excecao){
00277         if (email->getValor() == VALOR_INVALIDO)
00278             estado = FALHA;
00279     }
00280 }
00281
00282 int TUEmail::run(){
00283     setUp();
00284     testarCenarioSucesso();
00285     testarCenarioFalha();
00286     tearDown();
00287     return estado;
00288 }
00289
00290
00291 void TUEndereco::setUp(){
00292     endereco = new Endereco();
00293     estado = SUCESSO;
00294 }
00295
```

```
00296 void TUEndereco::tearDown(){
00297     delete endereco;
00298 }
00299
00300 void TUEndereco::testarCenarioSucesso(){
00301     try{
00302         endereco->setValor(VALOR_VALIDO);
00303         if (endereco->getValor() != VALOR_VALIDO)
00304             estado = FALHA;
00305     }
00306     catch(invalid_argument &excecao){
00307         estado = FALHA;
00308     }
00309 }
00310
00311 void TUEndereco::testarCenarioFalha(){
00312     try{
00313         endereco->setValor(VALOR_INVALIDO);
00314         estado = FALHA;
00315     }
00316     catch(invalid_argument &excecao){
00317         if (endereco->getValor() == VALOR_INVALIDO)
00318             estado = FALHA;
00319     }
00320 }
00321
00322 int TUEndereco::run(){
00323     setUp();
00324     testarCenarioSucesso();
00325     testarCenarioFalha();
00326     tearDown();
00327     return estado;
00328 }
00329
00330
00331 void TUHorario::setUp(){
00332     horario = new Horario();
00333     estado = SUCESSO;
00334 }
00335
00336 void TUHorario::tearDown(){
00337     delete horario;
00338 }
00339
00340 void TUHorario::testarCenarioSucesso(){
00341     try{
00342         horario->setValor(VALOR_VALIDO);
00343         if (horario->getValor() != VALOR_VALIDO)
00344             estado = FALHA;
00345     }
00346     catch(invalid_argument &excecao){
00347         estado = FALHA;
00348     }
00349 }
00350
00351 void TUHorario::testarCenarioFalha(){
00352     try{
00353         horario->setValor(VALOR_INVALIDO);
00354         estado = FALHA;
00355     }
00356     catch(invalid_argument &excecao){
00357         if (horario->getValor() == VALOR_INVALIDO)
00358             estado = FALHA;
00359     }
00360 }
00361
00362 int TUHorario::run(){
00363     setUp();
00364     testarCenarioSucesso();
00365     testarCenarioFalha();
00366     tearDown();
00367     return estado;
00368 }
00369
00370
00371 void TUIdioma::setUp(){
00372     idioma = new Idioma();
00373     estado = SUCESSO;
00374 }
00375
00376 void TUIdioma::tearDown(){
00377     delete idioma;
00378 }
00379
00380 void TUIdioma::testarCenarioSucesso(){
00381     try{
00382         idioma->setValor(VALOR_VALIDO);
```

```
00383         if (idioma->getValor() != VALOR_VALIDO)
00384             estado = FALHA;
00385     }
00386     catch(invalid_argument &excecao){
00387         estado = FALHA;
00388     }
00389 }
00390
00391 void TUIdioma::testarCenarioFalha(){
00392     try{
00393         idioma->setValor(VALOR_INVALIDO);
00394         estado = FALHA;
00395     }
00396     catch(invalid_argument &excecao){
00397         if (idioma->getValor() == VALOR_INVALIDO)
00398             estado = FALHA;
00399     }
00400 }
00401
00402 int TUIdioma::run(){
00403     setUp();
00404     testarCenarioSucesso();
00405     testarCenarioFalha();
00406     tearDown();
00407     return estado;
00408 }
00409
00410
00411 void TUNome::setUp(){
00412     nome = new Nome();
00413     estado = SUCESSO;
00414 }
00415
00416 void TUNome::tearDown(){
00417     delete nome;
00418 }
00419
00420 void TUNome::testarCenarioSucesso(){
00421     try{
00422         nome->setValor(VALOR_VALIDO);
00423         if (nome->getValor() != VALOR_VALIDO)
00424             estado = FALHA;
00425     }
00426     catch(invalid_argument &excecao){
00427         estado = FALHA;
00428     }
00429 }
00430
00431 void TUNome::testarCenarioFalha(){
00432     try{
00433         nome->setValor(VALOR_INVALIDO);
00434         estado = FALHA;
00435     }
00436     catch(invalid_argument &excecao){
00437         if (nome->getValor() == VALOR_INVALIDO)
00438             estado = FALHA;
00439     }
00440 }
00441
00442 int TUNome::run(){
00443     setUp();
00444     testarCenarioSucesso();
00445     testarCenarioFalha();
00446     tearDown();
00447     return estado;
00448 }
00449
00450
00451 void TUNota::setUp(){
00452     nota = new Nota();
00453     estado = SUCESSO;
00454 }
00455
00456 void TUNota::tearDown(){
00457     delete nota;
00458 }
00459
00460 void TUNota::testarCenarioSucesso(){
00461     try{
00462         nota->setValor(VALOR_VALIDO);
00463         if (nota->getValor() != VALOR_VALIDO)
00464             estado = FALHA;
00465     }
00466     catch(invalid_argument &excecao){
00467         estado = FALHA;
00468     }
00469 }
```

```
00470
00471 void TUNota::testarCenarioFalha(){
00472     try{
00473         nota->setValor(VALOR_INVALIDO);
00474         estado = FALHA;
00475     }
00476     catch(invalid_argument &excecao){
00477         if (nota->getValor() == VALOR_INVALIDO)
00478             estado = FALHA;
00479     }
00480 }
00481
00482 int TUNota::run(){
00483     setUp();
00484     testarCenarioSucesso();
00485     testarCenarioFalha();
00486     tearDown();
00487     return estado;
00488 }
00489
00490
00491 void TUSenha::setUp(){
00492     senha = new Senha();
00493     estado = SUCESSO;
00494 }
00495
00496 void TUSenha::tearDown(){
00497     delete senha;
00498 }
00499
00500 void TUSenha::testarCenarioSucesso(){
00501     try{
00502         senha->setValor(VALOR_VALIDO);
00503         if (senha->getValor() != VALOR_VALIDO)
00504             estado = FALHA;
00505     }
00506     catch(invalid_argument &excecao){
00507         estado = FALHA;
00508     }
00509 }
00510
00511 void TUSenha::testarCenarioFalha(){
00512     try{
00513         senha->setValor(VALOR_INVALIDO);
00514         estado = FALHA;
00515     }
00516     catch(invalid_argument &excecao){
00517         if (senha->getValor() == VALOR_INVALIDO)
00518             estado = FALHA;
00519     }
00520 }
00521
00522 int TUSenha::run(){
00523     setUp();
00524     testarCenarioSucesso();
00525     testarCenarioFalha();
00526     tearDown();
00527     return estado;
00528 }
00529
00530
00531 void TUTitulo::setUp(){
00532     titulo = new Titulo();
00533     estado = SUCESSO;
00534 }
00535
00536 void TUTitulo::tearDown(){
00537     delete titulo;
00538 }
00539
00540 void TUTitulo::testarCenarioSucesso(){
00541     try{
00542         titulo->setValor(VALOR_VALIDO);
00543         if (titulo->getValor() != VALOR_VALIDO)
00544             estado = FALHA;
00545     }
00546     catch(invalid_argument &excecao){
00547         estado = FALHA;
00548     }
00549 }
00550
00551 void TUTitulo::testarCenarioFalha(){
00552     try{
00553         titulo->setValor(VALOR_INVALIDO);
00554         estado = FALHA;
00555     }
00556     catch(invalid_argument &excecao){
```

```
00557         if (titulo->getValor() == VALOR_INVALIDO)
00558             estado = FALHA;
00559     }
00560 }
00561
00562 int TUTitulo::run(){
00563     setUp();
00564     testarCenarioSucesso();
00565     testarCenarioFalha();
00566     tearDown();
00567     return estado;
00568 }
```

5.9 testes_entidades.cpp

```
00001 #include "../headers/testes_entidades.h"
00002
00003
00004 /*-------------------------------------------------------------------------------------------------
00005
00006 // ------------------------------------------------------------------------
00007 // Implementa��es de m�todos de classe de teste de unidade.
00008 // No método TUProjeto::testarCenarioSucesso(), precisa implementar os trechos de código de dentro
       dele para cada atributo da entidade
00009
00010 void TUProjeto::setUp(){
00011     projeto = new Projeto();
00012     estado = SUCESSO;
00013 }
00014
00015 void TUProjeto::tearDown(){
00016     delete projeto;
00017 }
00018
00019 void TUProjeto::testarCenarioSucesso(){
00020     Codigo codigo;
00021     codigo.setValor(VALOR_VALIDO);
00022     projeto->setCodigo(codigo);
00023     if(projeto->getCodigo().getValor() != VALOR_VALIDO)
00024         estado = FALHA;
00025
00026     Prioridade prioridade;
00027     prioridade.setValor(VALOR_VALIDO);
00028     projeto->setPrioridade(prioridade);
00029     if(projeto->getPrioridade().getValor() != VALOR_VALIDO)
00030         estado = FALHA;
00031 }
00032
00033 int TUProjeto::run(){
00034     setUp();
00035     testarCenarioSucesso();
00036     tearDown();
00037     return estado;
00038 }
00039
00040 -------------------------------------------------------------------------------------------------*/
00041
00042 void TUUsuario::setUp(){
00043     usuario = new Usuario();
00044     estado = SUCESSO;
00045 }
00046
00047 void TUUsuario::tearDown(){
00048     delete usuario;
00049 }
00050
00051 void TUUsuario::testarCenarioSucesso(){
00052     Nome nome;
00053     nome.setValor(VALOR_VALIDO_NOME);
00054     usuario->setNome(nome);
00055     if(usuario->getNome().getValor() != VALOR_VALIDO_NOME)
00056         estado = FALHA;
00057
00058     Email email;
00059     email.setValor(VALOR_VALIDO_EMAIL);
00060     usuario->setEmail(email);
00061     if(usuario->getEmail().getValor() != VALOR_VALIDO_EMAIL)
00062         estado = FALHA;
00063
00064     Senha senha;
00065     senha.setValor(VALOR_VALIDO_SENHA);
00066     usuario->setSenha(senha);
00067     if(usuario->getSenha().getValor() != VALOR_VALIDO_SENHA)
```

```
00068         estado = FALHA;
00069 }
00070
00071 int TUUsuario::run(){
00072     setUp();
00073     testarCenarioSucesso();
00074     tearDown();
00075     return estado;
00076 }
00077
00078 void TUAvaliacao::setUp(){
00079     avaliacao = new Avaliacao();
00080     estado = SUCESSO;
00081 }
00082
00083 void TUAvaliacao::tearDown(){
00084     delete avaliacao;
00085 }
00086
00087 void TUAvaliacao::testarCenarioSucesso(){
00088     Codigo codigo;
00089     codigo.setValor(VALOR_VALIDO_CODIGO);
00090     avaliacao->setCodigo(codigo);
00091     if(avaliacao->getCodigo().getValor() != VALOR_VALIDO_CODIGO)
00092         estado = FALHA;
00093
00094     Nota nota;
00095     nota.setValor(VALOR_VALIDO_NOTA);
00096     avaliacao->setNota(nota);
00097     if(avaliacao->getNota().getValor() != VALOR_VALIDO_NOTA)
00098         estado = FALHA;
00099
00100     Descricao descricao;
00101     descricao.setValor(VALOR_VALIDO_DESCRICAO);
00102     avaliacao->setDescricao(descricao);
00103     if(avaliacao->getDescricao().getValor() != VALOR_VALIDO_DESCRICAO)
00104         estado = FALHA;
00105 }
00106
00107 int TUAvaliacao::run(){
00108     setUp();
00109     testarCenarioSucesso();
00110     tearDown();
00111     return estado;
00112 }
00113
00114 void TUSessao::setUp(){
00115     sessao = new Sessao();
00116     estado = SUCESSO;
00117 }
00118
00119 void TUSessao::tearDown(){
00120     delete sessao;
00121 }
00122
00123 void TUSessao::testarCenarioSucesso(){
00124     Codigo codigo;
00125     codigo.setValor(VALOR_VALIDO_CODIGO);
00126     sessao->setCodigo(codigo);
00127     if(sessao->getCodigo().getValor() != VALOR_VALIDO_CODIGO)
00128         estado = FALHA;
00129
00130     Data data;
00131     data.setValor(VALOR_VALIDO_DATA);
00132     sessao->setData(data);
00133     if(sessao->getData().getValor() != VALOR_VALIDO_DATA)
00134         estado = FALHA;
00135
00136     Horario horario;
00137     horario.setValor(VALOR_VALIDO_HORARIO);
00138     sessao->setHorario(horario);
00139     if(sessao->getHorario().getValor() != VALOR_VALIDO_HORARIO)
00140         estado = FALHA;
00141
00142     Idioma idioma;
00143     idioma.setValor(VALOR_VALIDO_IDIOMA);
00144     sessao->setIdioma(idioma);
00145     if(sessao->getIdioma().getValor() != VALOR_VALIDO_IDIOMA)
00146         estado = FALHA;
00147 }
00148
00149 int TUSessao::run(){
00150     setUp();
00151     testarCenarioSucesso();
00152     tearDown();
00153     return estado;
00154 }
```

```
00155
00156 void TUExcursao::setUp(){
00157     excursao = new Excursao();
00158     estado = SUCESSO;
00159 }
00160
00161 void TUExcursao::tearDown(){
00162     delete excursao;
00163 }
00164
00165 void TUExcursao::testarCenarioSucesso(){
00166     Codigo codigo;
00167     codigo.setValor(VALOR_VALIDO_CODIGO);
00168     excursao->setCodigo(codigo);
00169     if(excursao->getCodigo().getValor() != VALOR_VALIDO_CODIGO)
00170         estado = FALHA;
00171
00172     Titulo titulo;
00173     titulo.setValor(VALOR_VALIDO_TITULO);
00174     excursao->setTitulo(titulo);
00175     if(excursao->getTitulo().getValor() != VALOR_VALIDO_TITULO)
00176         estado = FALHA;
00177
00178     Nota nota;
00179     nota.setValor(VALOR_VALIDO_NOTA);
00180     excursao->setNota(nota);
00181     if(excursao->getNota().getValor() != VALOR_VALIDO_NOTA)
00182         estado = FALHA;
00183
00184     Cidade cidade;
00185     cidade.setValor(VALOR_VALIDO_CIDADE);
00186     excursao->setCidade(cidade);
00187     if(excursao->getCidade().getValor() != VALOR_VALIDO_CIDADE)
00188         estado = FALHA;
00189
00190     Duracao duracao;
00191     duracao.setValor(VALOR_VALIDO_DURACAO);
00192     excursao->setDuracao(duracao);
00193     if(excursao->getDuracao().getValor() != VALOR_VALIDO_DURACAO)
00194         estado = FALHA;
00195
00196     Descricao descricao;
00197     descricao.setValor(VALOR_VALIDO_DESCRICAO);
00198     excursao->setDescricao(descricao);
00199     if(excursao->getDescricao().getValor() != VALOR_VALIDO_DESCRICAO)
00200         estado = FALHA;
00201
00202     Endereco endereco;
00203     endereco.setValor(VALOR_VALIDO_ENDERECO);
00204     excursao->setEndereco(endereco);
00205     if(excursao->getEndereco().getValor() != VALOR_VALIDO_ENDERECO)
00206         estado = FALHA;
00207 }
00208
00209 int TUExcursao::run(){
00210     setUp();
00211     testarCenarioSucesso();
00212     tearDown();
00213     return estado;
00214 }
```

Index

Avaliacao, 7
  getCodigo, 7
  getDescricao, 8
  getNota, 8
  setCodigo, 8
  setDescricao, 9
  setNota, 9

Cidade, 9
  getValor, 10
  setValor, 10
Codigo, 11
  getValor, 11
  setValor, 11

Data, 12
  getValor, 12
  setValor, 13
Descricao, 13
  getValor, 14
  setValor, 14
Duracao, 15
  getValor, 15
  setValor, 15

Email, 16
  getValor, 16
  setValor, 17
Endereco, 17
  getValor, 18
  setValor, 18
Excursao, 19
  getCidade, 19
  getCodigo, 20
  getDescricao, 20
  getDuracao, 20
  getEndereco, 20
  getNota, 21
  getTitulo, 21
  setCidade, 21
  setCodigo, 22
  setDescricao, 22
  setDuracao, 22
  setEndereco, 23
  setNota, 23
  setTitulo, 23

FALHA
  TUAvaliacao, 34
  TUCidade, 36
  TUCodigo, 37
  TUData, 38
  TUDescricao, 39
  TUDuracao, 40
  TUEmail, 41
  TUEndereco, 42
  TUExcursao, 43
  TUHorario, 44
  TUIdioma, 45
  TUNome, 46
  TUNota, 47
  TUSenha, 48
  TUSessao, 49
  TUTitulo, 50
  TUUsuario, 51

getCidade
  Excursao, 19
getCodigo
  Avaliacao, 7
  Excursao, 20
  Sessao, 30
getData
  Sessao, 31
getDescricao
  Avaliacao, 8
  Excursao, 20
getDuracao
  Excursao, 20
getEmail
  Usuario, 52
getEndereco
  Excursao, 20
getHorario
  Sessao, 31
getIdioma
  Sessao, 31
getNome
  Usuario, 52
getNota
  Avaliacao, 8
  Excursao, 21
getSenha
  Usuario, 53
getTitulo
  Excursao, 21
getValor
  Cidade, 10
  Codigo, 11
  Data, 12

Descricao, 14
Duracao, 15
Email, 16
Endereco, 18
Horario, 24
Idioma, 25
Nome, 26
Nota, 28
Senha, 29
Titulo, 33

headers/dominios.h, 55
headers/entidades.h, 58
headers/testes_dominios.h, 61
headers/testes_entidades.h, 64
Horario, 24
getValor, 24
setValor, 24

Idioma, 25
getValor, 25
setValor, 25

Nome, 26
getValor, 26
setValor, 27
Nota, 27
getValor, 28
setValor, 28

run
TUAvaliacao, 34
TUCidade, 35
TUCodigo, 36
TUData, 38
TUDescricao, 39
TUDuracao, 40
TUEmail, 41
TUEndereco, 42
TUExcursao, 43
TUHorario, 44
TUIdioma, 45
TUNome, 46
TUNota, 47
TUSenha, 48
TUSessao, 49
TUTitulo, 50
TUUsuario, 51

Senha, 29
getValor, 29
setValor, 29
Sessao, 30
getCodigo, 30
getData, 31
getHorario, 31
getIdioma, 31
setCodigo, 31
setData, 32
setHorario, 32
setIdioma, 32
setCidade
Excursao, 21
setCodigo
Avaliacao, 8
Excursao, 22
Sessao, 31
setData
Sessao, 32
setDescricao
Avaliacao, 9
Excursao, 22
setDuracao
Excursao, 22
setEmail
Usuario, 53
setEndereco
Excursao, 23
setHorario
Sessao, 32
setIdioma
Sessao, 32
setNome
Usuario, 53
setNota
Avaliacao, 9
Excursao, 23
setSenha
Usuario, 54
setTitulo
Excursao, 23
setValor
Cidade, 10
Codigo, 11
Data, 13
Descricao, 14
Duracao, 15
Email, 17
Endereco, 18
Horario, 24
Idioma, 25
Nome, 27
Nota, 28
Senha, 29
Titulo, 33
source/dominios.cpp, 65
source/entidades.cpp, 71
source/main.cpp, 71
source/testes_dominios.cpp, 74
source/testes_entidades.cpp, 81
SUCESSO
TUAvaliacao, 35
TUCidade, 36
TUCodigo, 37
TUData, 38
TUDescricao, 39
TUDuracao, 40

TUEmail, 41
TUEndereco, 42
TUExcursao, 43
TUHorario, 44
TUIdioma, 45
TUNome, 46
TUNota, 47
TUSenha, 48
TUSessao, 49
TUTitulo, 50
TUUsuario, 51

Titulo, 33
getValor, 33
setValor, 33
TUAvaliacao, 34
FALHA, 34
run, 34
SUCESSO, 35
TUCidade, 35
FALHA, 36
run, 35
SUCESSO, 36
TUCodigo, 36
FALHA, 37
run, 36
SUCESSO, 37
TUData, 37
FALHA, 38
run, 38
SUCESSO, 38
TUDescricao, 38
FALHA, 39
run, 39
SUCESSO, 39
TUDuracao, 39
FALHA, 40
run, 40
SUCESSO, 40
TUEmail, 41
FALHA, 41
run, 41
SUCESSO, 41
TUEndereco, 42
FALHA, 42
run, 42
SUCESSO, 42
TUExcursao, 43
FALHA, 43
run, 43
SUCESSO, 43
TUHorario, 44
FALHA, 44
run, 44
SUCESSO, 44
TUIdioma, 45
FALHA, 45
run, 45
SUCESSO, 45
TUNome, 46
FALHA, 46
run, 46
SUCESSO, 46
TUNota, 47
FALHA, 47
run, 47
SUCESSO, 47
TUSenha, 48
FALHA, 48
run, 48
SUCESSO, 48
TUSessao, 49
FALHA, 49
run, 49
SUCESSO, 49
TUTitulo, 50
FALHA, 50
run, 50
SUCESSO, 50
TUUsuario, 51
FALHA, 51
run, 51
SUCESSO, 51

Usuario, 52
getEmail, 52
getNome, 52
getSenha, 53
setEmail, 53
setNome, 53
setSenha, 54