Gestão de Alojamentos Turísticos (Testes)

Generated by Doxygen 1.10.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 SmartStay Namespace Reference

Namespaces

- namespace Tests

4.2 SmartStay.Tests Namespace Reference

Data Structures

- class ClientTests
- class ValidatorTests

  Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Chapter 5

Data Structure Documentation

5.1 SmartStay.Tests.ClientTests Class Reference

Public Member Functions

- void Client_ValidData_CreatesClient ()

  Tests the constructor of the Client class when valid data is provided. Ensures the client is created successfully with the given information.
- void Client_FullData_CreatesClient ()

  Tests the constructor of the Client class when all optional parameters are provided. Ensures the client is created successfully with full details.
- void Client_InvalidEmail_ThrowsValidationException ()

  Tests the constructor of the Client class when invalid email is provided. Ensures a ValidationException is thrown.
- void Client_InvalidPhoneNumber_ThrowsValidationException ()

  Tests the constructor of the Client class when invalid phone number is provided. Ensures a ValidationException is thrown.
- void Client_InvalidAddress_ThrowsValidationException ()

  Tests the constructor of the Client class when invalid address is provided. Ensures a ValidationException is thrown.
- void Client_InvalidPaymentMethod_ThrowsValidationException ()

  Tests the constructor of the Client class when an invalid payment method is provided. Ensures a ValidationException is thrown.
- void Client_SetValidFirstName_UpdatesFirstName ()

  Tests the property setter and getter for FirstName. Ensures that a valid name can be set and retrieved correctly.
- void Client_SetValidLastName_UpdatesLastName ()

  Tests the property setter and getter for LastName. Ensures that a valid last name can be set and retrieved correctly.
- void Client_SetValidEmail_UpdatesEmail ()

  Tests the property setter and getter for Email. Ensures that a valid email can be set and retrieved correctly.
- void Client_SetValidPhoneNumber_UpdatesPhoneNumber ()

  Tests the property setter and getter for PhoneNumber. Ensures that a valid phone number can be set and retrieved correctly.
- void Client_SetValidAddress_UpdatesAddress ()

  Tests the property setter and getter for Address. Ensures that a valid address can be set and retrieved correctly.
- void Client_SetValidPaymentMethod_UpdatesPaymentMethod ()

  Tests the property setter and getter for PreferredPaymentMethod. Ensures that a valid payment method can be set and retrieved correctly.
- void Client_GenerateUniqueClientId_CreatesUniqueIds ()

  Tests the client ID generation to ensure it increments properly for multiple clients. Ensures that each client has a unique ID.
- void Client_ToString_ReturnsValidJson ()

  Tests the ToString method of the Client class. Ensures the client object is serialized to a JSON string with proper formatting.

5.1.1 Detailed Description

Definition at line 11 of file ClientTests.cs.

5.1.2 Member Function Documentation

5.1.2.1 Client_FullData_CreatesClient()

```
void SmartStay.Tests.ClientTests.Client_FullData_CreatesClient ( )  [inline]
```

Tests the constructor of the Client class when all optional parameters are provided. Ensures the client is created successfully with full details.

Definition at line 36 of file ClientTests.cs.

5.1.2.2 Client_GenerateUniqueClientId_CreatesUniqueIds()

```
void SmartStay.Tests.ClientTests.Client_GenerateUniqueClientId_CreatesUniqueIds ( )  [inline]
```

Tests the client ID generation to ensure it increments properly for multiple clients. Ensures that each client has a unique ID.

Definition at line 246 of file ClientTests.cs.

5.1.2.3 Client_InvalidAddress_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidAddress_ThrowsValidationException ( )  [inline]
```

Tests the constructor of the Client class when invalid address is provided. Ensures a ValidationException is thrown.

Definition at line 94 of file ClientTests.cs.

5.1.2.4 Client_InvalidEmail_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidEmail_ThrowsValidationException ( )  [inline]
```

Tests the constructor of the Client class when invalid email is provided. Ensures a ValidationException is thrown.

Definition at line 61 of file ClientTests.cs.

5.1.2.5 Client_InvalidPaymentMethod_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidPaymentMethod_ThrowsValidationException ( )
[inline]
```

Tests the constructor of the Client class when an invalid payment method is provided. Ensures a ValidationException is thrown.

Definition at line 112 of file ClientTests.cs.

5.1.2.6 Client_InvalidPhoneNumber_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidPhoneNumber_ThrowsValidationException ( )  [inline]
```

Tests the constructor of the Client class when invalid phone number is provided. Ensures a ValidationException is thrown.

Definition at line 77 of file ClientTests.cs.

5.1.2.7 Client_SetValidAddress_UpdatesAddress()

```
void SmartStay.Tests.ClientTests.Client_SetValidAddress_UpdatesAddress ( )  [inline]
```

Tests the property setter and getter for Address. Ensures that a valid address can be set and retrieved correctly.

Definition at line 204 of file ClientTests.cs.

5.1.2.8 Client_SetValidEmail_UpdatesEmail()

```
void SmartStay.Tests.ClientTests.Client_SetValidEmail_UpdatesEmail ( )  [inline]
```

Tests the property setter and getter for Email. Ensures that a valid email can be set and retrieved correctly.

Definition at line 167 of file ClientTests.cs.

5.1.2.9 Client_SetValidFirstName_UpdatesFirstName()

```
void SmartStay.Tests.ClientTests.Client_SetValidFirstName_UpdatesFirstName ( )  [inline]
```

Tests the property setter and getter for FirstName. Ensures that a valid name can be set and retrieved correctly.

Definition at line 131 of file ClientTests.cs.

5.1.2.10 Client_SetValidLastName_UpdatesLastName()

```
void SmartStay.Tests.ClientTests.Client_SetValidLastName_UpdatesLastName ( )  [inline]
```

Tests the property setter and getter for LastName. Ensures that a valid last name can be set and retrieved correctly.

Definition at line 149 of file ClientTests.cs.

5.1.2.11 Client_SetValidPaymentMethod_UpdatesPaymentMethod()

```
void SmartStay.Tests.ClientTests.Client_SetValidPaymentMethod_UpdatesPaymentMethod ( )  [inline]
```

Tests the property setter and getter for PreferredPaymentMethod. Ensures that a valid payment method can be set and retrieved correctly.

Definition at line 224 of file ClientTests.cs.

5.1.2.12 Client_SetValidPhoneNumber_UpdatesPhoneNumber()

```
void SmartStay.Tests.ClientTests.Client_SetValidPhoneNumber_UpdatesPhoneNumber ( )  [inline]
```

Tests the property setter and getter for PhoneNumber. Ensures that a valid phone number can be set and retrieved correctly.

Definition at line 185 of file ClientTests.cs.

5.1.2.13 Client_ToString_ReturnsValidJson()

```
void SmartStay.Tests.ClientTests.Client_ToString_ReturnsValidJson ( )  [inline]
```

Tests the ToString method of the Client class. Ensures the client object is serialized to a JSON string with proper formatting.

Definition at line 259 of file ClientTests.cs.

5.1.2.14 Client_ValidData_CreatesClient()

```
void SmartStay.Tests.ClientTests.Client_ValidData_CreatesClient ( )  [inline]
```

Tests the constructor of the Client class when valid data is provided. Ensures the client is created successfully with the given information.

Definition at line 18 of file ClientTests.cs.

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

- ClientTests.cs

5.2 SmartStay.Tests.ValidatorTests Class Reference

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Public Member Functions

- void ValidateName_ValidName_ReturnsName (string validName)

  The ValidateName_ValidName_ReturnsName Tests the ValidateName method with a valid name input and ensures it returns the same name.
- void ValidateName_InvalidName_ThrowsException (string invalidName)

  The ValidateName_InvalidName_ThrowsException Tests the ValidateName method with invalid name inputs (empty or null) and ensures a ValidationException is thrown.
- void ValidateAccommodationName_ValidName_ReturnsName (string validAccommodationName)

  The ValidateAccommodationName_ValidName_ReturnsName Tests the ValidateAccommodationName method with a valid accommodation name and ensures it returns the same name.
- void ValidateAccommodationName_InvalidName_ThrowsException (string invalidAccommodationName)

  The ValidateAccommodationName_InvalidName_ThrowsException Tests the ValidateAccommodationName method with invalid accommodation names (empty or null) and ensures a ValidationException is thrown.
- void ValidateEmail_ValidEmail_ReturnsEmail (string validEmail)

  The ValidateEmail_ValidEmail_ReturnsEmail Tests the ValidateEmail method with a valid email input and ensures it returns the same email.
- void ValidateEmail_InvalidEmail_ThrowsException (string invalidEmail)

  The ValidateEmail_InvalidEmail_ThrowsException Tests the ValidateEmail method with invalid email inputs (e.g., an email missing '@') and ensures a ValidationException is thrown.
- void ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber (string validPhoneNumber)

  The ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber Tests the ValidatePhoneNumber method with a valid phone number input and ensures it returns the same phone number.
- void ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException (string invalidPhoneNumber)

  The ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException Tests the ValidatePhoneNumber method with an invalid phone number input and ensures a ValidationException is thrown.
- void ValidateAddress_ValidAddress_ReturnsAddress (string validAddress)

  The ValidateAddress_ValidAddress_ReturnsAddress Tests the ValidateAddress method with a valid address and ensures it returns the same address.
- void ValidateAddress_InvalidAddress_ThrowsException (string invalidAddress)

  The ValidateAddress_InvalidAddress_ThrowsException Tests the ValidateAddress method with an invalid address (e.g., empty) and ensures a ValidationException is thrown.
- void ValidatePrice_ValidPrice_ReturnsPrice ()

  The ValidatePrice_ValidPrice_ReturnsPrice Tests the ValidatePrice method with a valid price input and ensures it returns the same price.
- void ValidatePrice_InvalidPrice_ThrowsException ()

  The ValidatePrice_InvalidPrice_ThrowsException Tests the ValidatePrice method with an invalid price input (e.g., zero or negative) and ensures a ValidationException is thrown.
- void ValidatePaymentAmount_ValidAmount_ReturnsAmount ()

  The ValidatePaymentAmount_ValidAmount_ReturnsAmount Tests the ValidatePaymentAmount method with a valid payment amount input and ensures it returns the same amount.
- void ValidatePaymentAmount_InvalidAmount_ThrowsException ()

  The ValidatePaymentAmount_InvalidAmount_ThrowsException Tests the ValidatePaymentAmount method with an invalid payment amount input (e.g., negative amount) and ensures a ValidationException is thrown.
- void ValidatePaymentStatus_ValidStatus_ReturnsStatus ()

  Validates that the payment status is correctly processed when a valid status is provided.
- void ValidatePaymentStatus_InvalidStatus_ThrowsException ()

  Validates that an exception is thrown when an invalid payment status (out of the predefined enum) is provided.
- void ValidateAccommodationType_ValidType_ReturnsType ()

  Validates that the accommodation type is correctly processed when a valid type is provided.
- void ValidateAccommodationType_InvalidType_ThrowsException ()

  Validates that an exception is thrown when an invalid accommodation type (out of the predefined enum) is provided.
- void ValidateCheckInDate_ValidFutureDate_ReturnsDate ()

  Validates that the check-in date is processed correctly when a future date is provided.

- void ValidateCheckInDate_PastDate_ThrowsException ()

  Validates that an exception is thrown when a past date is provided as the check-in date.
- void ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate ()

  Validates that the check-out date is processed correctly when it is after the check-in date.
- void ValidateCheckOutDate_InvalidDateRange_ThrowsException ()

  Validates that an exception is thrown when the check-out date is before the check-in date.
- void ValidateTotalCost_ValidCost_ReturnsTotalCost ()

  Validates that the total cost is correctly processed when a valid cost is provided.
- void ValidateTotalCost_InvalidCost_ThrowsException ()

  Validates that an exception is thrown when a negative cost is provided.
- void ValidatePayment_ValidPayment_ReturnsPayment ()

  Validates that the payment amount is correctly processed when a valid payment is provided.
- void ValidatePayment_InvalidPayment_ThrowsException ()

  Validates that an exception is thrown when a negative payment value is provided.
- void ValidateReservationStatus_ValidStatus_ReturnsStatus ()

  Validates that the reservation status is correctly processed when a valid status is provided.
- void ValidateReservationStatus_InvalidStatus_ThrowsException ()

  Validates that an exception is thrown when an invalid reservation status (out of the predefined enum) is provided.

5.2.1 Detailed Description

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Definition at line 32 of file ValidationTests.cs.

5.2.2 Member Function Documentation

5.2.2.1 ValidateAccommodationName_InvalidName_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationName_InvalidName_ThrowsException (
             string invalidAccommodationName )  [inline]
```

The ValidateAccommodationName_InvalidName_ThrowsException Tests the ValidateAccommodationName method with invalid accommodation names (empty or null) and ensures a ValidationException is thrown.

Parameters

invalidAccommodationName	The invalidAccommodationNamestring

Definition at line 90 of file ValidationTests.cs.

5.2.2.2 ValidateAccommodationName_ValidName_ReturnsName()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationName_ValidName_ReturnsName (
             string validAccommodationName )  [inline]
```

The ValidateAccommodationName_ValidName_ReturnsName Tests the ValidateAccommodationName method with a valid accommodation name and ensures it returns the same name.

Parameters

validAccommodationName	The validAccommodationNamestring

Definition at line 75 of file ValidationTests.cs.

5.2.2.3 ValidateAccommodationType_InvalidType_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationType_InvalidType_ThrowsException ( )
[inline]
```

Validates that an exception is thrown when an invalid accommodation type (out of the predefined enum) is provided.

Definition at line 298 of file ValidationTests.cs.

5.2.2.4 ValidateAccommodationType_ValidType_ReturnsType()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationType_ValidType_ReturnsType ( )  [inline]
```

Validates that the accommodation type is correctly processed when a valid type is provided.

Definition at line 286 of file ValidationTests.cs.

5.2.2.5 ValidateAddress_InvalidAddress_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAddress_InvalidAddress_ThrowsException (
             string invalidAddress )  [inline]
```

The ValidateAddress_InvalidAddress_ThrowsException Tests the ValidateAddress method with an invalid address (e.g., empty) and ensures a ValidationException is thrown.

Parameters

invalidAddress	The invalidAddressstring

Definition at line 186 of file ValidationTests.cs.

5.2.2.6 ValidateAddress_ValidAddress_ReturnsAddress()

```
void SmartStay.Tests.ValidatorTests.ValidateAddress_ValidAddress_ReturnsAddress (
             string validAddress )  [inline]
```

The ValidateAddress_ValidAddress_ReturnsAddress Tests the ValidateAddress method with a valid address and ensures it returns the same address.

Parameters

validAddress	The validAddressstring

Definition at line 172 of file ValidationTests.cs.

5.2.2.7 ValidateCheckInDate_PastDate_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckInDate_PastDate_ThrowsException ( )  [inline]
```

Validates that an exception is thrown when a past date is provided as the check-in date.

Definition at line 325 of file ValidationTests.cs.

5.2.2.8 ValidateCheckInDate_ValidFutureDate_ReturnsDate()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckInDate_ValidFutureDate_ReturnsDate ( )  [inline]
```

Validates that the check-in date is processed correctly when a future date is provided.

Definition at line 314 of file ValidationTests.cs.

5.2.2.9 ValidateCheckOutDate_InvalidDateRange_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckOutDate_InvalidDateRange_ThrowsException ( )
[inline]
```

Validates that an exception is thrown when the check-out date is before the check-in date.

Definition at line 352 of file ValidationTests.cs.

5.2.2.10 ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate (
)  [inline]
```

Validates that the check-out date is processed correctly when it is after the check-in date.

Definition at line 340 of file ValidationTests.cs.

5.2.2.11 ValidateEmail_InvalidEmail_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateEmail_InvalidEmail_ThrowsException (
             string invalidEmail )  [inline]
```

The ValidateEmail_InvalidEmail_ThrowsException Tests the ValidateEmail method with invalid email inputs (e.g., an email missing '@') and ensures a ValidationException is thrown.

Parameters

invalidEmail	The invalidEmailstring

Definition at line 122 of file ValidationTests.cs.

5.2.2.12 ValidateEmail_ValidEmail_ReturnsEmail()

```
void SmartStay.Tests.ValidatorTests.ValidateEmail_ValidEmail_ReturnsEmail (
             string validEmail )  [inline]
```

The ValidateEmail_ValidEmail_ReturnsEmail Tests the ValidateEmail method with a valid email input and ensures it returns the same email.

Parameters

validEmail	The validEmailstring

Definition at line 108 of file ValidationTests.cs.

5.2.2.13 ValidateName_InvalidName_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateName_InvalidName_ThrowsException (
             string invalidName )  [inline]
```

The ValidateName_InvalidName_ThrowsException Tests the ValidateName method with invalid name inputs (empty or null) and ensures a ValidationException is thrown.

Parameters

invalidName	The invalidNamestring

Definition at line 58 of file ValidationTests.cs.

5.2.2.14 ValidateName_ValidName_ReturnsName()

```
void SmartStay.Tests.ValidatorTests.ValidateName_ValidName_ReturnsName (
             string validName )  [inline]
```

The ValidateName_ValidName_ReturnsName Tests the ValidateName method with a valid name input and ensures it returns the same name.

Parameters

validName	The validNamestring

Definition at line 43 of file ValidationTests.cs.

5.2.2.15 ValidatePayment_InvalidPayment_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePayment_InvalidPayment_ThrowsException ( )  [inline]
```

Validates that an exception is thrown when a negative payment value is provided.

Definition at line 406 of file ValidationTests.cs.

5.2.2.16 ValidatePayment_ValidPayment_ReturnsPayment()

```
void SmartStay.Tests.ValidatorTests.ValidatePayment_ValidPayment_ReturnsPayment ( )  [inline]
```

Validates that the payment amount is correctly processed when a valid payment is provided.

Definition at line 395 of file ValidationTests.cs.

5.2.2.17 ValidatePaymentAmount_InvalidAmount_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentAmount_InvalidAmount_ThrowsException ( )
[inline]
```

The ValidatePaymentAmount_InvalidAmount_ThrowsException Tests the ValidatePaymentAmount method with an invalid payment amount input (e.g., negative amount) and ensures a ValidationException is thrown.

Definition at line 243 of file ValidationTests.cs.

5.2.2.18 ValidatePaymentAmount_ValidAmount_ReturnsAmount()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentAmount_ValidAmount_ReturnsAmount ( )  [inline]
```

The ValidatePaymentAmount_ValidAmount_ReturnsAmount Tests the ValidatePaymentAmount method with a valid payment amount input and ensures it returns the same amount.

Definition at line 230 of file ValidationTests.cs.

5.2.2.19 ValidatePaymentStatus_InvalidStatus_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentStatus_InvalidStatus_ThrowsException ( )
[inline]
```

Validates that an exception is thrown when an invalid payment status (out of the predefined enum) is provided.

Definition at line 270 of file ValidationTests.cs.

5.2.2.20 ValidatePaymentStatus_ValidStatus_ReturnsStatus()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentStatus_ValidStatus_ReturnsStatus ( )  [inline]
```

Validates that the payment status is correctly processed when a valid status is provided.

Definition at line 259 of file ValidationTests.cs.

5.2.2.21 ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException (
             string invalidPhoneNumber )  [inline]
```

The ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException Tests the ValidatePhoneNumber method with an invalid phone number input and ensures a ValidationException is thrown.

Parameters

invalidPhoneNumber	The invalidPhoneNumberstring

Definition at line 154 of file ValidationTests.cs.

5.2.2.22 ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber()

```
void SmartStay.Tests.ValidatorTests.ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber (
             string validPhoneNumber )  [inline]
```

The ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber Tests the ValidatePhoneNumber method with a valid phone number input and ensures it returns the same phone number.

Parameters

validPhoneNumber	The validPhoneNumberstring

Definition at line 140 of file ValidationTests.cs.

5.2.2.23 ValidatePrice_InvalidPrice_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePrice_InvalidPrice_ThrowsException ( )  [inline]
```

The ValidatePrice_InvalidPrice_ThrowsException Tests the ValidatePrice method with an invalid price input (e.g., zero or negative) and ensures a ValidationException is thrown.

Definition at line 214 of file ValidationTests.cs.

5.2.2.24 ValidatePrice_ValidPrice_ReturnsPrice()

```
void SmartStay.Tests.ValidatorTests.ValidatePrice_ValidPrice_ReturnsPrice ( )  [inline]
```

The ValidatePrice_ValidPrice_ReturnsPrice Tests the ValidatePrice method with a valid price input and ensures it returns the same price.

Definition at line 201 of file ValidationTests.cs.

5.2.2.25 ValidateReservationStatus_InvalidStatus_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateReservationStatus_InvalidStatus_ThrowsException (
)  [inline]
```

Validates that an exception is thrown when an invalid reservation status (out of the predefined enum) is provided.

Definition at line 433 of file ValidationTests.cs.

5.2.2.26 ValidateReservationStatus_ValidStatus_ReturnsStatus()

```
void SmartStay.Tests.ValidatorTests.ValidateReservationStatus_ValidStatus_ReturnsStatus ( )
[inline]
```

Validates that the reservation status is correctly processed when a valid status is provided.

Definition at line 421 of file ValidationTests.cs.

5.2.2.27 ValidateTotalCost_InvalidCost_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateTotalCost_InvalidCost_ThrowsException ( )  [inline]
```

Validates that an exception is thrown when a negative cost is provided.

Definition at line 380 of file ValidationTests.cs.

5.2.2.28 ValidateTotalCost_ValidCost_ReturnsTotalCost()

```
void SmartStay.Tests.ValidatorTests.ValidateTotalCost_ValidCost_ReturnsTotalCost ( )  [inline]
```

Validates that the total cost is correctly processed when a valid cost is provided.

Definition at line 369 of file ValidationTests.cs.

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

- ValidationTests.cs

Chapter 6

File Documentation

6.1 AccommodationTests.cs File Reference

6.2 AccommodationTests.cs

Go to the documentation of this file.

```
```

6.3 ClientTests.cs File Reference

Data Structures

- class SmartStay.Tests.ClientTests

Namespaces

- namespace SmartStay
- namespace SmartStay.Tests

6.4 ClientTests.cs

Go to the documentation of this file.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using SmartStay.Models;
using SmartStay.Models.Enums;
using SmartStay.Services;
using SmartStay.Validation;
using System;

namespace SmartStay.Tests
{
[TestClass]
public class ClientTests
{
    [TestMethod]
    public void Client_ValidData_CreatesClient()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var client = new Client(firstName, lastName, email);

        Assert.IsNotNull(client);
        Assert.AreEqual(firstName, client.FirstName);
        Assert.AreEqual(lastName, client.LastName);
        Assert.AreEqual(email, client.Email);
    }

    [TestMethod]
    public void Client_FullData_CreatesClient()
    {
        var firstName = "Jane";
        var lastName = "Smith";
        var email = "jane.smith@example.com";
        var phoneNumber = "+351999999999";
        var address = "123 Elm St, Springfield";
        var preferredPaymentMethod = PaymentMethod.BankTransfer;

        var client = new Client(firstName, lastName, email, phoneNumber, address, preferredPaymentMethod);

        Assert.IsNotNull(client);
        Assert.AreEqual(firstName, client.FirstName);
        Assert.AreEqual(lastName, client.LastName);
        Assert.AreEqual(email, client.Email);
        Assert.AreEqual(phoneNumber, client.PhoneNumber);
        Assert.AreEqual(address, client.Address);
        Assert.AreEqual(preferredPaymentMethod, client.PreferredPaymentMethod);
    }

    [TestMethod]
    public void Client_InvalidEmail_ThrowsValidationException()
    {
        var firstName = "John";
        var lastName = "Doe";
        var invalidEmail = "invalid-email";

        var exception =
            Assert.ThrowsException<ValidationException>(() => new Client(firstName, lastName, invalidEmail));
        Assert.AreEqual(ValidationErrorCode.InvalidEmail, exception.ErrorCode);
    }

    [TestMethod]
    public void Client_InvalidPhoneNumber_ThrowsValidationException()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var invalidPhoneNumber = "invalid-phone";

        var exception = Assert.ThrowsException<ValidationException>(
            () => new Client(firstName, lastName, email, invalidPhoneNumber, "123 Elm St"));
        Assert.AreEqual(ValidationErrorCode.InvalidPhoneNumber, exception.ErrorCode);
    }

    [TestMethod]
    public void Client_InvalidAddress_ThrowsValidationException()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var phoneNumber = "+351999999999";
        var invalidAddress = "";

        var exception = Assert.ThrowsException<ValidationException>(
            () => new Client(firstName, lastName, email, phoneNumber, invalidAddress));
        Assert.AreEqual(ValidationErrorCode.InvalidAddress, exception.ErrorCode);
    }

    [TestMethod]
    public void Client_InvalidPaymentMethod_ThrowsValidationException()
    {
        var firstName = "Jane";
        var lastName = "Smith";
        var email = "jane.smith@example.com";
        var phoneNumber = "+351999999999";
        var address = "123 Elm St, Springfield";
        var invalidPaymentMethod = (PaymentMethod)999; // Invalid payment method

        var exception = Assert.ThrowsException<ValidationException>(
            () => new Client(firstName, lastName, email, phoneNumber, address, invalidPaymentMethod));
        Assert.AreEqual(ValidationErrorCode.InvalidPaymentMethod, exception.ErrorCode);
    }

    [TestMethod]
    public void Client_SetValidFirstName_UpdatesFirstName()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var client = new Client(firstName, lastName, email);

        var newFirstName = "Johnny";
        client.FirstName = newFirstName;

        Assert.AreEqual(newFirstName, client.FirstName);
    }

    [TestMethod]
    public void Client_SetValidLastName_UpdatesLastName()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var client = new Client(firstName, lastName, email);

        var newLastName = "Smith";
        client.LastName = newLastName;

        Assert.AreEqual(newLastName, client.LastName);
    }

    [TestMethod]
    public void Client_SetValidEmail_UpdatesEmail()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var client = new Client(firstName, lastName, email);

        var newEmail = "johnny.doe@example.com";
        client.Email = newEmail;

        Assert.AreEqual(newEmail, client.Email);
    }

    [TestMethod]
    public void Client_SetValidPhoneNumber_UpdatesPhoneNumber()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var phoneNumber = "+351999999999";
        var client = new Client(firstName, lastName, email, phoneNumber, "123 Elm St");

        var newPhoneNumber = "+351888888888";
        client.PhoneNumber = newPhoneNumber;

        Assert.AreEqual(newPhoneNumber, client.PhoneNumber);
    }

    [TestMethod]
    public void Client_SetValidAddress_UpdatesAddress()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var phoneNumber = "+351999999999";
        var address = "123 Elm St, Springfield";
        var client = new Client(firstName, lastName, email, phoneNumber, address);

        var newAddress = "456 Oak St, Springfield";
        client.Address = newAddress;

        Assert.AreEqual(newAddress, client.Address);
    }

    [TestMethod]
    public void Client_SetValidPaymentMethod_UpdatesPaymentMethod()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var phoneNumber = "+351999999999";
        var address = "123 Elm St, Springfield";
        var preferredPaymentMethod = PaymentMethod.PayPal;

        var client = new Client(firstName, lastName, email, phoneNumber, address,
preferredPaymentMethod);

        var newPaymentMethod = PaymentMethod.BankTransfer;
        client.PreferredPaymentMethod = newPaymentMethod;

        Assert.AreEqual(newPaymentMethod, client.PreferredPaymentMethod);
    }

    [TestMethod]
    public void Client_GenerateUniqueClientId_CreatesUniqueIds()
    {
        var firstClient = new Client("John", "Doe", "john.doe@example.com");
        var secondClient = new Client("Jane", "Smith", "jane.smith@example.com");

        Assert.AreNotEqual(firstClient.Id, secondClient.Id);
    }

    [TestMethod]
    public void Client_ToString_ReturnsValidJson()
    {
        var firstName = "John";
        var lastName = "Doe";
        var email = "john.doe@example.com";
        var client = new Client(firstName, lastName, email);

        var json = client.ToString();

        Assert.IsTrue(json.Contains("\"FirstName\": \"John\""));
        Assert.IsTrue(json.Contains("\"LastName\": \"Doe\""));
        Assert.IsTrue(json.Contains("\"Email\": \"john.doe@example.com\""));
    }
}
}
```

6.5 ReservationTests.cs File Reference

6.6 ReservationTests.cs

Go to the documentation of this file.

```

```

6.7 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs File Reference

6.8 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs

Go to the documentation of this file.

```
// <autogenerated />
using System;
using System.Reflection;
[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
FrameworkDisplayName = ".NET 8.0")]
```

6.9 SmartStay.Tests.AssemblyInfo.cs File Reference

6.10 SmartStay.Tests.AssemblyInfo.cs

Go to the documentation of this file.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Tests")]
[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
[assembly:
System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+547a43e26c0e6800e3b3c5899fba97a9e932c12b")]
[assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Tests")]
[assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Tests")]
[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Generated by the MSBuild WriteCodeFragment class.

```

6.11 SmartStay.Tests.GlobalUsings.g.cs File Reference

6.12 SmartStay.Tests.GlobalUsings.g.cs

Go to the documentation of this file.

```
// <auto-generated/>
global using global::Microsoft.VisualStudio.TestTools.UnitTesting;
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
global using global::System.Threading.Tasks;
```

6.13 ValidationTests.cs File Reference

Data Structures

- class SmartStay.Tests.ValidatorTests

  Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Namespaces

- namespace SmartStay
- namespace SmartStay.Tests

6.14 ValidationTests.cs

Go to the documentation of this file.

```

using SmartStay.Models.Enums;
using SmartStay.Validation;

namespace SmartStay.Tests
{
[TestClass]
public class ValidatorTests
{
#region Name Validation Tests
```

```

    [TestMethod]
    [DataRow("John Doe")]
    public void ValidateName_ValidName_ReturnsName(string validName)
    {
        var result = Validator.ValidateName(validName);
        Assert.AreEqual(validName, result);
    }

    [TestMethod]
    [DataRow("")]
    [DataRow(null)]
    public void ValidateName_InvalidName_ThrowsException(string invalidName)
    {
        var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateName(invalidName));
        Assert.AreEqual(ValidationErrorCode.InvalidName, exception.ErrorCode);
    }

#endregion

#region Accommodation Name Validation Tests

    [TestMethod]
    [DataRow("Ocean View Resort")]
    public void ValidateAccommodationName_ValidName_ReturnsName(string validAccommodationName)
    {
        var result = Validator.ValidateAccommodationName(validAccommodationName);
        Assert.AreEqual(validAccommodationName, result);
    }

    [TestMethod]
    [DataRow("")]
    [DataRow(null)]
    public void ValidateAccommodationName_InvalidName_ThrowsException(string invalidAccommodationName)
    {
        var exception = Assert.ThrowsException<ValidationException>(
            () => Validator.ValidateAccommodationName(invalidAccommodationName));
        Assert.AreEqual(ValidationErrorCode.InvalidAccommodationName, exception.ErrorCode);
    }

#endregion

#region Email Validation Tests

    [TestMethod]
    [DataRow("user@example.com")]
    public void ValidateEmail_ValidEmail_ReturnsEmail(string validEmail)
    {
        var result = Validator.ValidateEmail(validEmail);
        Assert.AreEqual(validEmail, result);
    }

    [TestMethod]
    [DataRow("invalid-email")]
    public void ValidateEmail_InvalidEmail_ThrowsException(string invalidEmail)
    {
        var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateEmail(invalidEmail));
        Assert.AreEqual(ValidationErrorCode.InvalidEmail, exception.ErrorCode);
    }

#endregion

#region Phone Number Validation Tests

    [TestMethod]
    [DataRow("+351234567890")]
    public void ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber(string validPhoneNumber)
    {
        var result = Validator.ValidatePhoneNumber(validPhoneNumber);
        Assert.AreEqual(validPhoneNumber, result);
    }

    [TestMethod]
    [DataRow("12345")]
    public void ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException(string invalidPhoneNumber)
    {
        var exception =
            Assert.ThrowsException<ValidationException>(() =>
Validator.ValidatePhoneNumber(invalidPhoneNumber));
        Assert.AreEqual(ValidationErrorCode.InvalidPhoneNumber, exception.ErrorCode);
    }

#endregion

#region Address Validation Tests
```

```
    [TestMethod]
    [DataRow("123 Main St")]
    public void ValidateAddress_ValidAddress_ReturnsAddress(string validAddress)
    {
        var result = Validator.ValidateAddress(validAddress);
        Assert.AreEqual(validAddress, result);
    }

    [TestMethod]
    [DataRow("")]
    public void ValidateAddress_InvalidAddress_ThrowsException(string invalidAddress)
    {
        var exception = Assert.ThrowsException<ValidationException>(() => Validator.ValidateAddress(invalidAddress));
        Assert.AreEqual(ValidationErrorCode.InvalidAddress, exception.ErrorCode);
    }

#endregion

#region Price Validation Tests

    [TestMethod]
    public void ValidatePrice_ValidPrice_ReturnsPrice()
    {
        decimal validPrice = 100.0m;
        var result = Validator.ValidatePrice(validPrice);
        Assert.AreEqual(validPrice, result);
    }

    [TestMethod]
    public void ValidatePrice_InvalidPrice_ThrowsException()
    {
        decimal invalidPrice = 0.0m;
        var exception = Assert.ThrowsException<ValidationException>(() => Validator.ValidatePrice(invalidPrice));
        Assert.AreEqual(ValidationErrorCode.InvalidPrice, exception.ErrorCode);
    }

#endregion

#region Payment Amounts Validation Tests

    [TestMethod]
    public void ValidatePaymentAmount_ValidAmount_ReturnsAmount()
    {
        decimal validAmount = 150.0m;
        var result = Validator.ValidatePaymentAmount(validAmount);
        Assert.AreEqual(validAmount, result);
    }

    [TestMethod]
    public void ValidatePaymentAmount_InvalidAmount_ThrowsException()
    {
        decimal invalidAmount = -10.0m;
        var exception =
            Assert.ThrowsException<ValidationException>(() => Validator.ValidatePaymentAmount(invalidAmount));
        Assert.AreEqual(ValidationErrorCode.InvalidPaymentValue, exception.ErrorCode);
    }

#endregion

#region Payment Status Validation Tests

    [TestMethod]
    public void ValidatePaymentStatus_ValidStatus_ReturnsStatus()
    {
        var validStatus = PaymentStatus.Completed;
        var result = Validator.ValidatePaymentStatus(validStatus);
        Assert.AreEqual(validStatus, result);
    }

    [TestMethod]
    public void ValidatePaymentStatus_InvalidStatus_ThrowsException()
    {
        var invalidStatus = (PaymentStatus)999;
        var exception =
            Assert.ThrowsException<ValidationException>(() => Validator.ValidatePaymentStatus(invalidStatus));
        Assert.AreEqual(ValidationErrorCode.InvalidPaymentStatus, exception.ErrorCode);
    }

#endregion

#region Accommodation Type Validation Tests

```

```
    [TestMethod]
    public void ValidateAccommodationType_ValidType_ReturnsType()
    {
        var validType = AccommodationType.Hotel;
        var result = Validator.ValidateAccommodationType(validType);
        Assert.AreEqual(validType, result);
    }

    [TestMethod]
    public void ValidateAccommodationType_InvalidType_ThrowsException()
    {
        var invalidType = (AccommodationType)999;
        var exception =
            Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateAccommodationType(invalidType));
        Assert.AreEqual(ValidationErrorCode.InvalidAccommodationType, exception.ErrorCode);
    }

#endregion

#region Check In Date Validation Tests

    [TestMethod]
    public void ValidateCheckInDate_ValidFutureDate_ReturnsDate()
    {
        var futureDate = DateTime.Now.AddDays(1);
        var result = Validator.ValidateCheckInDate(futureDate);
        Assert.AreEqual(futureDate, result);
    }

    [TestMethod]
    public void ValidateCheckInDate_PastDate_ThrowsException()
    {
        var pastDate = DateTime.Now.AddDays(-1);
        var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateCheckInDate(pastDate));
        Assert.AreEqual(ValidationErrorCode.InvalidDate, exception.ErrorCode);
    }

#endregion

#region Check Out Date Validation Tests

    [TestMethod]
    public void ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()
    {
        var checkInDate = DateTime.Now.AddDays(1);
        var checkOutDate = checkInDate.AddDays(1);
        var result = Validator.ValidateCheckOutDate(checkOutDate, checkInDate);
        Assert.AreEqual(checkOutDate, result);
    }

    [TestMethod]
    public void ValidateCheckOutDate_InvalidDateRange_ThrowsException()
    {
        var checkInDate = DateTime.Now.AddDays(1);
        var checkOutDate = checkInDate.AddDays(-1);
        var exception = Assert.ThrowsException<ValidationException>(
            () => Validator.ValidateCheckOutDate(checkOutDate, checkInDate));
        Assert.AreEqual(ValidationErrorCode.InvalidDateRange, exception.ErrorCode);
    }

#endregion

#region Total Cost Validation Tests

    [TestMethod]
    public void ValidateTotalCost_ValidCost_ReturnsTotalCost()
    {
        decimal validCost = 300.0m;
        var result = Validator.ValidateTotalCost(validCost);
        Assert.AreEqual(validCost, result);
    }

    [TestMethod]
    public void ValidateTotalCost_InvalidCost_ThrowsException()
    {
        decimal invalidCost = -5.0m;
        var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateTotalCost(invalidCost));
        Assert.AreEqual(ValidationErrorCode.InvalidTotalCost, exception.ErrorCode);
    }

#endregion

#region Payment Validation Tests

```

```
    [TestMethod]
    public void ValidatePayment_ValidPayment_ReturnsPayment()
    {
        decimal validPayment = 10.0m;
        var result = Validator.ValidatePayment(validPayment);
        Assert.AreEqual(validPayment, result);
    }

    [TestMethod]
    public void ValidatePayment_InvalidPayment_ThrowsException()
    {
        decimal invalidPayment = -100.0m;
        var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidatePayment(invalidPayment));
        Assert.AreEqual(ValidationErrorCode.InvalidPaymentValue, exception.ErrorCode);
    }

#endregion

#region Reservation Status Validation Tests

    [TestMethod]
    public void ValidateReservationStatus_ValidStatus_ReturnsStatus()
    {
        var validStatus = ReservationStatus.Confirmed;
        var result = Validator.ValidateReservationStatus(validStatus);
        Assert.AreEqual(validStatus, result);
    }

    [TestMethod]
    public void ValidateReservationStatus_InvalidStatus_ThrowsException()
    {
        var invalidStatus = (ReservationStatus)999;
        var exception =
            Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateReservationStatus(invalidStatus));
        Assert.AreEqual(ValidationErrorCode.InvalidReservationStatus, exception.ErrorCode);
    }

#endregion
}
}
```

Index