Gestão de Alojamentos Turísticos

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

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 SmartStay Namespace Reference

Namespaces

- namespace Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.
- namespace Repositories

  The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.
- namespace Services

  The `SmartStay.Services` namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.
- namespace Utilities

  The `SmartStay.Utilities` namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.
- namespace Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

Data Structures

- class **Program**

  The Program class is the main entry point for the SmartStay application. This application is designed for managing tourist accommodations, including functionalities for client management, reservations, and check-ins.

5.2 SmartStay.Models Namespace Reference

The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

Namespaces

- namespace Enums

  This namespace contains enumerations related to accommodation types used within the SmartStay application.
- namespace Interfaces

  This namespace contains interfaces used within the SmartStay application.

Data Structures

- class Accommodation

  Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.
- class Client

  Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.
- class Payment

  Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.
- class Reservation

  Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

5.2.1 Detailed Description

The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

5.3 SmartStay.Models.Enums Namespace Reference

This namespace contains enumerations related to accommodation types used within the SmartStay application.

Enumerations

- enum AccommodationType {
  Hotel , House , Apartment , Villa ,
  BedAndBreakfast , Hostel , Resort , Cottage ,
  Cabin , Guesthouse , Chalet , Lodge }

  Enumeration representing different types of accommodations available for booking.
- enum PaymentMethod { None , PayPal , MultiBanco , BankTransfer }

  Enumeration representing the possible payment methods available for transactions.
- enum PaymentStatus {
  Unpaid , Pending , Completed , PartiallyPaid ,
  Rejected , Refunded , Cancelled }

  Enumerator representing payment status.
- enum ReservationStatus {
  Pending , CheckedIn , CheckedOut , Cancelled ,
  NoShow , Confirmed , Declined }

  Enumeration representing the current status of a reservation.

5.3.1 Detailed Description

This namespace contains enumerations related to accommodation types used within the SmartStay application.

<copyright file="AccommodationType.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the AccommodationType enumeration used in the SmartStay application, representing different accommodation types available for booking. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="PaymentMethod.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the PaymentMethod enumeration used in the SmartStay application, representing different payment methods available for bookings and transactions. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="PaymentStatus.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the PaymentStatus enumeration used in the SmartStay application representing various payment status. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="ReservationStatus.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the ReservationStatus enumeration used in the SmartStay application, representing the different statuses a reservation can have. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

5.3.2 Enumeration Type Documentation

5.3.2.1 AccommodationType

`enum SmartStay.Models.Enums.AccommodationType`

Enumeration representing different types of accommodations available for booking.

Enumerator

Hotel	Represents a traditional hotel accommodation, typically offering private rooms and common amenities.
House	Represents a standalone house accommodation, ideal for private stays and larger groups.
Apartment	Represents an apartment accommodation, typically part of a larger building, offering self-contained living space.
Villa	Represents a villa accommodation, usually a larger, luxury residence often with a private pool and garden.
BedAndBreakfast	Represents a bed and breakfast accommodation, providing a private room with breakfast included, often in a home setting.
Hostel	Represents a hostel accommodation, often offering dormitory-style rooms and shared facilities, popular among budget travelers.
Resort	Represents a resort accommodation, typically offering all-inclusive services and multiple leisure amenities on-site.
Cottage	Represents a cottage accommodation, usually a small, cozy house in a rural or nature setting.
Cabin	Represents a cabin accommodation, typically a small, rustic structure often located in remote or forested areas.
Guesthouse	Represents a guesthouse accommodation, which offers a private room within a larger property, usually with shared amenities.
Chalet	Represents a chalet accommodation, usually a wooden house located in mountain regions, popular for ski vacations.
Lodge	Represents a lodge accommodation, typically found in nature destinations, offering basic to luxurious amenities.

Definition at line 19 of file AccommodationType.cs.

5.3.2.2 PaymentMethod

enum SmartStay.Models.Enums.PaymentMethod

Enumeration representing the possible payment methods available for transactions.

Enumerator

None	No specific payment method selected; used as a default or placeholder value.
PayPal	Payment method through PayPal, allowing secure online payments.
MultiBanco	Payment method using MultiBanco, a popular Portuguese banking payment system.
BankTransfer	Payment method via bank transfer, where funds are transferred directly between bank accounts.

Definition at line 19 of file PaymentMethod.cs.

5.3.2.3 PaymentStatus

enum SmartStay.Models.Enums.PaymentStatus

Enumerator representing payment status.

Enumerator

Unpaid	Payment has not been made yet.
Pending	Payment has been initiated but not yet completed (e.g., pending in processing).
Completed	Payment has been completed successfully.
PartiallyPaid	Payment was partially completed; more payments are expected.
Rejected	Payment was rejected, usually by the payment processor.
Refunded	Payment was refunded to the client.
Cancelled	Payment has been cancelled, typically by the client or system.

Definition at line 19 of file PaymentStatus.cs.

5.3.2.4 ReservationStatus

enum SmartStay.Models.Enums.ReservationStatus

Enumeration representing the current status of a reservation.

Enumerator

Pending	Reservation has been made but the client has not yet checked in.
CheckedIn	Client has checked in to the accommodation.
CheckedOut	Client has checked out from the accommodation.
Cancelled	Reservation was cancelled before the client checked in.
NoShow	Client did not show up for the reservation.
Confirmed	Reservation has been confirmed, but the client has not yet checked in.
Declined	Reservation was declined or denied due to some issue (e.g., payment failure, overbooked, etc.).

Definition at line 19 of file ReservationStatus.cs.

5.4 SmartStay.Models.Interfaces Namespace Reference

This namespace contains interfaces used within the SmartStay application.

Data Structures

- interface IManageableEntity

  Defines the IManageableEntity<T> interface for managing a collection of entities of type T . This interface standardizes methods for adding, removing, importing, and exporting entities.

5.4.1 Detailed Description

This namespace contains interfaces used within the SmartStay application.

<copyright file="ManageableEntity.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the definition of the IManageableEntity interface, which provides a standard structure for managing collections of entities within the SmartStay application.

This interface can be implemented by any collection class to provide a consistent API for managing entities, facilitating code reuse and standardization across different types of entity collections (e.g., Clients, Reservations, Accommodations). </file> <author>Enrique Rodrigues</author> <date>11/11/2024</date>

5.5 SmartStay.Repositories Namespace Reference

The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

Data Structures

- class Accommodations

  Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.
- class Clients

  Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.
- class Reservations

  Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

5.5.1 Detailed Description

The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

5.6 SmartStay.Services Namespace Reference

The SmartStay.Services namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.

Data Structures

- class **BookingManager**

  Provides a static facade for managing clients, reservations, and accommodations in the booking system. This class centralizes all operations for adding, removing, importing, and exporting data for these entities. It interacts with internal repositories to simplify the main API and ensure a standardized approach.

5.6.1 Detailed Description

The SmartStay.Services namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.

5.7 SmartStay.Utilities Namespace Reference

The SmartStay.Utilities namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

Data Structures

- class AccommodationConverter

  Custom JSON converter for Accommodation objects, used to serialize and deserialize accommodations to and from JSON format. It provides custom handling for the reserved dates and accommodation type.
- class DateRangeComparer

  Implements IComparer<T> to provide comparison logic for tuples of DateTime values representing date ranges. The comparison is done based on the Start date of each range.
- class **JsonHelper**

  Provides static methods to serialize and deserialize objects to and from JSON format.

5.7.1 Detailed Description

The SmartStay.Utilities namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

<copyright file="DateRangeComparer.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the DateRangeComparer class, which implements the IComparer<T> interface to provide custom comparison logic for date ranges (represented as tuples of DateTime values). The comparison is based on the start date of each range, used primarily for sorting or ordering date ranges. </file> <author>Enrique Rodrigues</author> <date>10/11/2024</date>

5.8 SmartStay.Validation Namespace Reference

The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

Data Structures

- class **ValidationErrorMessages**

  Provides error messages corresponding to each ValidationErrorCode value used in client data validation.
- class **Validator**

  Provides a set of static methods for validating input data in the SmartStay application, including names, emails, phone numbers, addresses, and payment methods.

Enumerations

- enum ValidationErrorCode {
  InvalidName = 1001 , InvalidEmail = 1002 , InvalidPhoneNumber = 1003 , InvalidAddress = 1004 ,
  InvalidPaymentMethod = 1005 , InvalidAccommodationType = 1006 , InvalidId = 1007 , InvalidDateRange = 1008 ,
  InvalidDate = 1009 , InvalidTotalCost = 1010 , InvalidPaymentValue = 1011 , InvalidReservationStatus = 1012 ,
  InvalidAccommodationName = 1013 , InvalidPrice = 1014 , InvalidPaymentStatus = 1015 }

  Defines error codes for validation failures within the SmartStay application.

Functions

- class ValidationException (ValidationErrorCode errorCode)

  Initializes a new instance of the ValidationException class with a specified validation error code. The error message is derived from ValidationErrorMessages.GetErrorMessage based on the provided error code.

5.8.1 Detailed Description

The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

<copyright file="ValidationErrorCodes.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the definition of the ValidationErrorCode enum, which represents specific error codes related to validation failures within the SmartStay application.

Contains the ValidationErrorCode enum, which defines error codes for validation errors that can occur when processing client data, such as invalid names, emails, phone numbers, addresses, and payment methods. </file> <author>Enrique Rodrigues</author> <date>09/11/2024</date>

<copyright file="ValidationErrorMessages.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the ValidationErrorMessages static class, which provides human-readable error messages for validation error codes used in the SmartStay application. </file> <author>Enrique Rodrigues</author> <date>09/11/2024</date>

<copyright file="ValidationException.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the ValidationException class, which represents a custom exception used for handling validation errors within the SmartStay application. The exception includes a specific error code and a descriptive message for easy identification of validation issues. </file> <author>Enrique Rodrigues</author> <date>09/11/2024</date>

5.8.2 Enumeration Type Documentation

5.8.2.1 ValidationErrorCode

```
enum SmartStay.Validation.ValidationErrorCode
```

Defines error codes for validation failures within the SmartStay application.

Enumerator

InvalidName	Error code indicating that the provided name is invalid.
InvalidEmail	Error code indicating that the provided email address is invalid.
InvalidPhoneNumber	Error code indicating that the provided phone number is invalid.
InvalidAddress	Error code indicating that the provided address is invalid.
InvalidPaymentMethod	Error code indicating that the provided payment method is invalid.
InvalidAccommodationType	Error code indicating that the provided accommodation type is invalid.
InvalidId	Error code indicating that the provided ID is invalid.
InvalidDateRange	Error code indicating that the provided date range is invalid, typically when the check-in date is later than or equal to the check-out date.
InvalidDate	Error code indicating that the provided date is invalid, typically when the date is in the past or does not meet the expected criteria.
InvalidTotalCost	Error code indicating that the total cost provided is invalid, usually if it is a negative value.
InvalidPaymentValue	Error code indicating that the provided payment value is invalid, such as when it is negative or exceeds the total cost.
InvalidReservationStatus	Error code indicating that the provided reservation status is invalid, typically if it does not match any defined status in the ReservationStatus enumeration.
InvalidAccommodationName	Error code indicating that the provided accommodation name is invalid.
InvalidPrice	Error code indicating that the provided price is invalid.
InvalidPaymentStatus	Error code indicating that the provided status is invalid.

Definition at line 25 of file ValidationErrorCodes.cs.

5.8.3 Function Documentation

5.8.3.1 ValidationException()

```
class SmartStay.Validation.ValidationException (
            ValidationErrorCode errorCode )
```

Initializes a new instance of the ValidationException class with a specified validation error code. The error message is derived from ValidationErrorMessages.GetErrorMessage based on the provided error code.

Parameters

errorCode	The ValidationErrorCode indicating the type of validation error.

Gets the specific ValidationErrorCode associated with this validation exception, providing context for the validation failure.

Definition at line 25 of file ValidationException.cs.

Chapter 6

Data Structure Documentation

6.1 SmartStay.Models.Accommodation Class Reference

Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.

Public Member Functions

- Accommodation (AccommodationType type, string name, string address, decimal pricePerNight)

  Initializes a new instance of the Accommodation class with the specified details: type, name, address, and price per night.
- bool IsAvailable (DateTime startDate, DateTime endDate)

  Checks if the accommodation is available for the specified date range.
- bool AddReservation (DateTime startDate, DateTime endDate)

  Attempts to add a reservation for the specified date range if the accommodation is available.
- decimal CalculateTotalCost (DateTime startDate, DateTime endDate)

  Calculates the total cost for a given stay duration.
- override string ToString ()

  Overridden ToString method to provide accommodation information in a readable JSON format.

Properties

- int Id `[get]`

  Public getter for the accommodation ID.
- AccommodationType Type `[get, set]`

  Public getter and setter for the Type.
- string Name `[get, set]`

  Public getter and setter for the Name.
- string Address `[get, set]`

  Public getter and setter for the Address.
- decimal PricePerNight `[get, set]`

  Public getter and setter for the PricePerNight.
- IReadOnlyList<(DateTime Start, DateTime End)> ReservedDates `[get]`

  Public getter for a read-only list of reserved dates.

6.1.1 Detailed Description

Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.

Definition at line 29 of file Accommodation.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Accommodation()

```
SmartStay.Models.Accommodation.Accommodation (
            AccommodationType type,
            string name,
            string address,
            decimal pricePerNight )  [inline]
```

Initializes a new instance of the Accommodation class with the specified details: type, name, address, and price per night.

Parameters

type	The type of the accommodation (e.g., Hotel, House).
name	The name of the accommodation.
address	The address of the accommodation.
pricePerNight	The nightly price of the accommodation.

Exceptions

ValidationException	Thrown if any of the provided parameters fail validation:
ValidationException	Thrown if the accommodation type is invalid.
ValidationException	Thrown if the accommodation name is invalid.
ValidationException	Thrown if the address is invalid.
ValidationException	Thrown if the price per night is invalid.

The constructor validates the provided parameters using the Validator class before initializing the properties. If any validation fails, a ValidationException is thrown with the appropriate error code.

Definition at line 59 of file Accommodation.cs.

6.1.3 Member Function Documentation

6.1.3.1 AddReservation()

```
bool SmartStay.Models.Accommodation.AddReservation (
            DateTime startDate,
            DateTime endDate )  [inline]
```

Attempts to add a reservation for the specified date range if the accommodation is available.

Parameters

startDate	The start date of the booking.
endDate	The end date of the booking.

Returns

True if the reservation was successfully added; otherwise, false.

This method first checks if the accommodation is available for the given date range using IsAvailable. If available, it inserts the booking in the sorted list at the correct position to maintain the list's order.

Definition at line 164 of file Accommodation.cs.

6.1.3.2 CalculateTotalCost()

```
decimal SmartStay.Models.Accommodation.CalculateTotalCost (
            DateTime startDate,
            DateTime endDate )  [inline]
```

Calculates the total cost for a given stay duration.

Parameters

startDate	The start date of the stay.
endDate	The end date of the stay.

Returns

The total cost for the stay based on the price per night.

Exceptions

ArgumentException	Thrown when the end date is before the start date.

Definition at line 187 of file Accommodation.cs.

6.1.3.3 IsAvailable()

```
bool SmartStay.Models.Accommodation.IsAvailable (
            DateTime startDate,
            DateTime endDate )  [inline]
```

Checks if the accommodation is available for the specified date range.

Parameters

startDate	The start date of the requested booking period.
endDate	The end date of the requested booking period.

Returns

True if the accommodation is available for the entire date range; otherwise, false.

Exceptions

ArgumentException	Thrown if the end date is not after the start date.

Definition at line 130 of file Accommodation.cs.

6.1.3.4 ToString()

```
override string SmartStay.Models.Accommodation.ToString ( )  [inline]
```

Overridden ToString method to provide accommodation information in a readable JSON format.

Returns

A JSON string representation of the accommodation object.

Definition at line 218 of file Accommodation.cs.

6.1.4 Property Documentation

6.1.4.1 Address

```
string SmartStay.Models.Accommodation.Address  [get], [set], [add]
```

Public getter and setter for the Address.

Definition at line 103 of file Accommodation.cs.

6.1.4.2 Id

```
int SmartStay.Models.Accommodation.Id  [get]
```

Public getter for the accommodation ID.

Definition at line 80 of file Accommodation.cs.

6.1.4.3 Name

```
string SmartStay.Models.Accommodation.Name  [get], [set]
```

Public getter and setter for the Name.

Definition at line 94 of file Accommodation.cs.

6.1.4.4 PricePerNight

```
decimal SmartStay.Models.Accommodation.PricePerNight  [get], [set]
```

Public getter and setter for the PricePerNight.

Definition at line 112 of file Accommodation.cs.

6.1.4.5 ReservedDates

```
IReadOnlyList<(DateTime Start, DateTime End)> SmartStay.Models.Accommodation.ReservedDates
[get]
```

Public getter for a read-only list of reserved dates.

Definition at line 121 of file Accommodation.cs.

6.1.4.6 Type

```
AccommodationType SmartStay.Models.Accommodation.Type  [get], [set]
```

Public getter and setter for the Type.

Definition at line 85 of file Accommodation.cs.

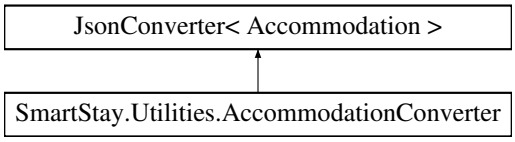
The documentation for this class was generated from the following file:

- Accommodation.cs

6.2 SmartStay.Utilities.AccommodationConverter Class Reference

Custom JSON converter for Accommodation objects, used to serialize and deserialize accommodations to and from JSON format. It provides custom handling for the reserved dates and accommodation type.

Inheritance diagram for SmartStay.Utilities.AccommodationConverter:



Public Member Functions

- override Accommodation Read (ref Utf8JsonReader reader, Type typeToConvert, JsonSerializerOptions options)

  Reads and deserializes an Accommodation object from JSON.
- override void Write (Utf8JsonWriter writer, Accommodation value, JsonSerializerOptions options)

  Writes an Accommodation object as JSON.

6.2.1 Detailed Description

Custom JSON converter for Accommodation objects, used to serialize and deserialize accommodations to and from JSON format. It provides custom handling for the reserved dates and accommodation type.

Definition at line 25 of file AccommodationConverter.cs.

6.2.2 Member Function Documentation

6.2.2.1 Read()

```
override Accommodation SmartStay.Utilities.AccommodationConverter.Read (
            ref Utf8JsonReader reader,
            Type typeToConvert,
            JsonSerializerOptions options )  [inline]
```

Reads and deserializes an Accommodation object from JSON.

Parameters

reader	The JSON reader containing the JSON data.
typeToConvert	The type of object to convert.
options	The options to use for deserialization.

Returns

The deserialized Accommodation object.

Exceptions

JsonException	Thrown if the deserialization fails.

Definition at line 35 of file AccommodationConverter.cs.

6.2.2.2 Write()

```
override void SmartStay.Utilities.AccommodationConverter.Write (
            Utf8JsonWriter writer,
            Accommodation value,
            JsonSerializerOptions options )  [inline]
```

Writes an Accommodation object as JSON.

Parameters

writer	The JSON writer to write the serialized object to.
value	The Accommodation object to serialize.
options	The options to use for serialization.

Definition at line 50 of file AccommodationConverter.cs.

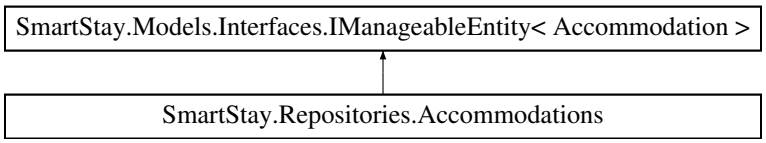
The documentation for this class was generated from the following file:

- AccommodationConverter.cs

6.3 SmartStay.Repositories.Accommodations Class Reference

Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.

Inheritance diagram for SmartStay.Repositories.Accommodations:



Public Member Functions

- bool Add (Accommodation accommodation)

  Attempts to add a new accommodation to the collection.
- bool Remove (Accommodation accommodation)

  Removes an accommodation from the collection.
- void Import (string data)

  Imports accommodations from a JSON string into the collection. Existing accommodations with the same ID are replaced.
- string Export ()

  Exports the current list of accommodations to a JSON string.
- Accommodation? FindAccommodationById (int accommodationId)

  Finds an accommodation by its unique ID.
- IReadOnlyCollection< Accommodation > GetAllAccommodations ()

  Retrieves all the accommodations in the collection.
- int CountAccommodations ()

  Counts the number of accommodations in the collection.

Public Member Functions inherited from SmartStay.Models.Interfaces.IManageableEntity< Accommodation >

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- void Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.

6.3.1 Detailed Description

Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.

Definition at line 28 of file Accommodations.cs.

6.3.2 Member Function Documentation

6.3.2.1 Add()

```
bool SmartStay.Repositories.Accommodations.Add (
            Accommodation accommodation )  [inline]
```

Attempts to add a new accommodation to the collection.

Parameters

accommodation	The Accommodation to add to the collection.

Returns

`true` if the accommodation was successfully added to the collection; `false` if an accommodation with the same ID already exists in the collection.

Exceptions

ArgumentNullException	Thrown if *accommodation* is `null`.

Definition at line 44 of file Accommodations.cs.

6.3.2.2 CountAccommodations()

```
int SmartStay.Repositories.Accommodations.CountAccommodations ( )  [inline]
```

Counts the number of accommodations in the collection.

Returns

The number of accommodations in the collection.

Definition at line 142 of file Accommodations.cs.

6.3.2.3 Export()

```
string SmartStay.Repositories.Accommodations.Export ( )  [inline]
```

Exports the current list of accommodations to a JSON string.

Returns

A JSON string representation of the accommodations in the collection.</returns

Definition at line 107 of file Accommodations.cs.

6.3.2.4 FindAccommodationById()

```
Accommodation?  SmartStay.Repositories.Accommodations.FindAccommodationById (
            int accommodationId )  [inline]
```

Finds an accommodation by its unique ID.

Parameters

accommodationId	The unique ID of the accommodation to find.

Returns

Returns the Accommodation object if found; otherwise, `null`.

Definition at line 119 of file Accommodations.cs.

6.3.2.5 GetAllAccommodations()

```
IReadOnlyCollection< Accommodation > SmartStay.Repositories.Accommodations.GetAllAccommodations
( )  [inline]
```

Retrieves all the accommodations in the collection.

Returns

A read-only collection of Accommodation objects.

Definition at line 131 of file Accommodations.cs.

6.3.2.6 Import()

```
void SmartStay.Repositories.Accommodations.Import (
            string data )  [inline]
```

Imports accommodations from a JSON string into the collection. Existing accommodations with the same ID are replaced.

Parameters

data	The JSON string containing the list of accommodations.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 86 of file Accommodations.cs.

6.3.2.7 Remove()

```
bool SmartStay.Repositories.Accommodations.Remove (
            Accommodation accommodation )  [inline]
```

Removes an accommodation from the collection.

Parameters

accommodation	The Accommodation object to remove from the collection.

Returns

`true` if the accommodation was successfully removed from the collection; `false` if the accommodation was not found.

Exceptions

ArgumentNullException	Thrown if *accommodation* is `null`.</exception

Definition at line 69 of file Accommodations.cs.

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

- Accommodations.cs

6.4 SmartStay.Models.Client Class Reference

Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Public Member Functions

- Client (string firstName, string lastName, string email)

  Constructor to initialize a new client with basic details: first name, last name, and email. Validates the input parameters.
- Client (string firstName, string lastName, string email, string phoneNumber, string address)

  Constructor to initialize a new client with basic details (first name, last name, email) and additional details (phone number and address).
- Client (string firstName, string lastName, string email, string phoneNumber, string address, PaymentMethod preferredPaymentMethod)

  Constructor to initialize a new client with all details including the preferred payment method.
- override string ToString ()

  Overridden ToString method to provide client information in a readable JSON format.

Properties

- int Id `[get]`

  Public getter for the user Id.
- string FirstName `[get, set]`

  Public getter and setter for the FirstName. Sets the value after validating it.
- string LastName `[get, set]`

  Public getter and setter for the LastName. Sets the value after validating it.
- string Email `[get, set]`

  Public getter and setter for the Email. Sets the value after validating it.
- string PhoneNumber `[get, set]`

  Public getter and setter for the PhoneNumber. Sets the value after validating it.
- string Address `[get, set]`

  Public getter and setter for the Address. Sets the value after validating it.
- PaymentMethod PreferredPaymentMethod `[get, set]`

  Public getter and setter for the PreferredPaymentMethod. Sets the value after validating it.

6.4.1 Detailed Description

Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Definition at line 29 of file Client.cs.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Client() [1/3]

```
SmartStay.Models.Client.Client (
            string firstName,
            string lastName,
            string email )  [inline]
```

Constructor to initialize a new client with basic details: first name, last name, and email. Validates the input parameters.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid.

Definition at line 52 of file Client.cs.

6.4.2.2 Client() [2/3]

```
SmartStay.Models.Client.Client (
            string firstName,
            string lastName,
            string email,
            string phoneNumber,
            string address )  [inline]
```

Constructor to initialize a new client with basic details (first name, last name, email) and additional details (phone number and address).

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.
phoneNumber	The phone number of the client.
address	The residential address of the client.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid.

Definition at line 77 of file Client.cs.

6.4.2.3 Client() [3/3]

```
SmartStay.Models.Client.Client (
            string firstName,
            string lastName,
            string email,
            string phoneNumber,
            string address,
            PaymentMethod preferredPaymentMethod )  [inline]
```

Constructor to initialize a new client with all details including the preferred payment method.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.
phoneNumber	The phone number of the client.
address	The residential address of the client.
preferredPaymentMethod	The preferred payment method of the client.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid.

Definition at line 99 of file Client.cs.

6.4.3 Member Function Documentation

6.4.3.1 ToString()

```
override string SmartStay.Models.Client.ToString ( )  [inline]
```

Overridden ToString method to provide client information in a readable JSON format.

Returns

A JSON string representation of the client object.

Definition at line 194 of file Client.cs.

6.4.4 Property Documentation

6.4.4.1 Address

```
string SmartStay.Models.Client.Address  [get], [set], [add]
```

Public getter and setter for the Address. Sets the value after validating it.

Definition at line 158 of file Client.cs.

6.4.4.2 Email

```
string SmartStay.Models.Client.Email  [get], [set]
```

Public getter and setter for the Email. Sets the value after validating it.

Definition at line 138 of file Client.cs.

6.4.4.3 FirstName

```
string SmartStay.Models.Client.FirstName  [get], [set]
```

Public getter and setter for the FirstName. Sets the value after validating it.

Definition at line 118 of file Client.cs.

6.4.4.4 Id

```
int SmartStay.Models.Client.Id  [get]
```

Public getter for the user Id.

Definition at line 112 of file Client.cs.

6.4.4.5 LastName

```
string SmartStay.Models.Client.LastName  [get], [set]
```

Public getter and setter for the LastName. Sets the value after validating it.

Definition at line 128 of file Client.cs.

6.4.4.6 PhoneNumber

```
string SmartStay.Models.Client.PhoneNumber  [get], [set]
```

Public getter and setter for the PhoneNumber. Sets the value after validating it.

Definition at line 148 of file Client.cs.

6.4.4.7 PreferredPaymentMethod

```
PaymentMethod SmartStay.Models.Client.PreferredPaymentMethod  [get], [set]
```

Public getter and setter for the PreferredPaymentMethod. Sets the value after validating it.

Definition at line 168 of file Client.cs.

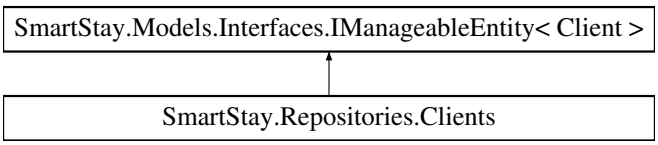
The documentation for this class was generated from the following file:

- Client.cs

6.5 SmartStay.Repositories.Clients Class Reference

Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.

Inheritance diagram for SmartStay.Repositories.Clients:



Public Member Functions

- bool Add (Client client)

  Attempts to add a new client to the collection.
- bool Remove (Client client)

  Removes a client from the collection.
- void Import (string data)

  Imports a list of clients from a JSON string. Replaces any existing clients with the same ID in the collection.
- string Export ()

  Exports the current list of clients to a JSON string.
- Client? FindClientById (int id)

  Finds a client by their unique ID.
- IReadOnlyCollection< Client > GetAllClients ()

  Retrieves all the clients in the collection.
- int CountClients ()

  Counts the number of clients in the collection.

Public Member Functions inherited from SmartStay.Models.Interfaces.IManageableEntity< Client >

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- void Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.

6.5.1 Detailed Description

Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.

Definition at line 25 of file Clients.cs.

6.5.2 Member Function Documentation

6.5.2.1 Add()

```
bool SmartStay.Repositories.Clients.Add (
            Client client )  [inline]
```

Attempts to add a new client to the collection.

Parameters

client	The Client to add to the collection.

Returns

`true` if the client was successfully added to the collection; `false` if a client with the same ID already exists in the collection.

Exceptions

ArgumentNullException	Thrown if *client* is `null`.

Definition at line 43 of file Clients.cs.

6.5.2.2 CountClients()

```
int SmartStay.Repositories.Clients.CountClients ( )  [inline]
```

Counts the number of clients in the collection.

Returns

The number of clients in the collection.

Definition at line 142 of file Clients.cs.

6.5.2.3 Export()

```
string SmartStay.Repositories.Clients.Export ( )  [inline]
```

Exports the current list of clients to a JSON string.

Returns

A JSON string representation of the clients in the collection.

Definition at line 104 of file Clients.cs.

6.5.2.4 FindClientById()

```
Client?  SmartStay.Repositories.Clients.FindClientById (
            int id )  [inline]
```

Finds a client by their unique ID.

Parameters

id	The unique ID of the client to find.

Returns

Returns the Client object if found; otherwise, `null`.

Definition at line 116 of file Clients.cs.

6.5.2.5 GetAllClients()

```
IReadOnlyCollection< Client > SmartStay.Repositories.Clients.GetAllClients ( )  [inline]
```

Retrieves all the clients in the collection.

Returns

A read-only collection of Client objects.

Returns a copy of the internal dictionary's values to prevent external modification.

Definition at line 131 of file Clients.cs.

6.5.2.6 Import()

```
void SmartStay.Repositories.Clients.Import (
            string data )  [inline]
```

Imports a list of clients from a JSON string. Replaces any existing clients with the same ID in the collection.

Parameters

data	The JSON string containing the list of clients.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 84 of file Clients.cs.

6.5.2.7 Remove()

```
bool SmartStay.Repositories.Clients.Remove (
            Client client )  [inline]
```

Removes a client from the collection.

Parameters

client	The Client object to remove from the collection.

Returns

`true` if the client was successfully removed from the collection; `false` if the client was not found.

Exceptions

ArgumentNullException	Thrown if *client* is `null`.

Definition at line 68 of file Clients.cs.

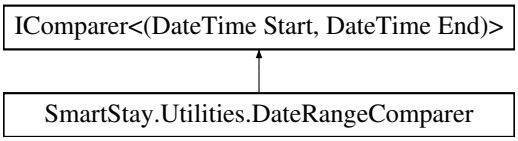
The documentation for this class was generated from the following file:

- Clients.cs

6.6 SmartStay.Utilities.DateRangeComparer Class Reference

Implements IComparer<T> to provide comparison logic for tuples of DateTime values representing date ranges. The comparison is done based on the Start date of each range.

Inheritance diagram for SmartStay.Utilities.DateRangeComparer:



Public Member Functions

- int Compare ((DateTime Start, DateTime End) x,(DateTime Start, DateTime End) y)
  Compares two date ranges based on their start date.

6.6.1 Detailed Description

Implements IComparer<T> to provide comparison logic for tuples of DateTime values representing date ranges. The comparison is done based on the Start date of each range.

Definition at line 24 of file DateRangeComparer.cs.

6.6.2 Member Function Documentation

6.6.2.1 Compare()

```
int SmartStay.Utilities.DateRangeComparer.Compare (
            (DateTime Start, DateTime End) x,
            (DateTime Start, DateTime End) y )  [inline]
```

Compares two date ranges based on their start date.

Parameters

x	The first DateTime tuple representing a date range with a Start and End date.
y	The second DateTime tuple representing a date range with a Start and End date.

Returns

A value indicating the relative order of the two date ranges:

- A negative value if *x* starts before *y* .
- Zero if both date ranges start at the same time.
- A positive value if *x* starts after *y* .

Definition at line 37 of file DateRangeComparer.cs.

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

- DateRangeComparer.cs

6.7 SmartStay.Models.Interfaces.IManageableEntity< in T > Interface Template Reference

Defines the IManageableEntity<T> interface for managing a collection of entities of type *T* . This interface standardizes methods for adding, removing, importing, and exporting entities.

Public Member Functions

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- void Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.

6.7.1 Detailed Description

Defines the IManageableEntity<T> interface for managing a collection of entities of type *T* . This interface standardizes methods for adding, removing, importing, and exporting entities.

Template Parameters

T	The type of entities managed by the implementing collection class.

Definition at line 26 of file ManageableEntity.cs.

6.7.2 Member Function Documentation

6.7.2.1 Add()

```
bool SmartStay.Models.Interfaces.IManageableEntity< in T >.Add (
            T item )
```

Adds a single entity of type *T* to the collection.

Parameters

item	The entity to add to the collection.

Returns

Returns `true` if the entity was successfully added; otherwise, `false`.

6.7.2.2 Export()

```
string SmartStay.Models.Interfaces.IManageableEntity< in T >.Export ( )
```

Exports the current list of items as a serialized string.

Returns

A serialized string representing the collection of items.

6.7.2.3 Import()

```
void SmartStay.Models.Interfaces.IManageableEntity< in T >.Import (
            string data )
```

Imports a list of items from a serialized string.

Parameters

data	The serialized string representing a collection of items.

6.7.2.4 Remove()

```
bool SmartStay.Models.Interfaces.IManageableEntity< in T >.Remove (
            T item )
```

Removes a specified entity of type *T* from the collection.

Parameters

item	The entity to remove from the collection.

Returns

Returns `true` if the entity was successfully removed; otherwise, `false`.

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

- ManageableEntity.cs

6.8 SmartStay.Models.Payment Class Reference

Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

Public Member Functions

- Payment (int reservationId, decimal amount, DateTime paymentDate, PaymentMethod paymentMethod, PaymentStatus paymentStatus)

  Initializes a new instance of the Payment class with specified details.
- override string ToString ()

  Overridden ToString method to provide payment information in a readable JSON format.

Properties

- int Id `[get]`

  Public getter for the payment Id.
- int ReservationId `[get]`

  Public getter for the reservation Id being paid.
- decimal Amount `[get]`

  Public getter for the Amount.
- DateTime Date `[get]`

  Gets the date the payment was made.
- PaymentMethod Method `[get]`

  Gets the method used for the payment (e.g., Credit Card, Bank Transfer).
- PaymentStatus Status `[get, set]`

  Gets or sets the status of the payment. When setting, validates the new status using Validator.ValidatePaymentStatus.

6.8.1 Detailed Description

Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

Definition at line 24 of file Payment.cs.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Payment()

```
SmartStay.Models.Payment.Payment (
            int reservationId,
            decimal amount,
            DateTime paymentDate,
            PaymentMethod paymentMethod,
            PaymentStatus paymentStatus )  [inline]
```

Initializes a new instance of the Payment class with specified details.

Parameters

amount	The amount for the payment.
paymentDate	The date when the payment was made.
paymentMethod	The method used for the payment.
paymentStatus	The status of the payment.

Exceptions

ValidationException	Thrown when the provided amount, payment method, or payment status is invalid.

Definition at line 46 of file Payment.cs.

6.8.3 Member Function Documentation

6.8.3.1 ToString()

```
override string SmartStay.Models.Payment.ToString ( )  [inline]
```

Overridden ToString method to provide payment information in a readable JSON format.

Returns

A JSON string representation of the payment object.

Definition at line 123 of file Payment.cs.

6.8.4 Property Documentation

6.8.4.1 Amount

```
decimal SmartStay.Models.Payment.Amount  [get]
```

Public getter for the Amount.

Definition at line 77 of file Payment.cs.

6.8.4.2 Date

```
DateTime SmartStay.Models.Payment.Date [get]
```

Gets the date the payment was made.

Definition at line 82 of file Payment.cs.

6.8.4.3 Id

```
int SmartStay.Models.Payment.Id [get]
```

Public getter for the payment Id.

Definition at line 67 of file Payment.cs.

6.8.4.4 Method

```
PaymentMethod SmartStay.Models.Payment.Method [get]
```

Gets the method used for the payment (e.g., Credit Card, Bank Transfer).

Definition at line 87 of file Payment.cs.

6.8.4.5 ReservationId

```
int SmartStay.Models.Payment.ReservationId [get]
```

Public getter for the reservation Id being paid.

Definition at line 72 of file Payment.cs.

6.8.4.6 Status

```
PaymentStatus SmartStay.Models.Payment.Status [get], [set]
```

Gets or sets the status of the payment. When setting, validates the new status using Validator.ValidatePayment↩ Status.

Exceptions

ValidationException	Thrown when the provided status is invalid.

Definition at line 96 of file Payment.cs.

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

- Payment.cs

6.9 SmartStay.Models.Reservation Class Reference

Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

Public Member Functions

- Reservation (int clientId, int accommodationId, AccommodationType accommodationType, DateTime checkInDate, DateTime checkOutDate, decimal totalCost)

  Constructor to initialize a new reservation with essential details. Validates the input parameters.
- void CheckIn ()

  Marks the reservation as checked in and updates the status to CheckedIn.
- void CheckOut ()

  Marks the reservation as checked out and updates the status to CheckedOut.
- void MakePayment (decimal paymentAmount, PaymentMethod paymentMethod)

  Makes a payment towards the reservation and adds a new Payment object to the payment list.
- bool IsFullyPaid ()

  Checks if the reservation is fully paid.
- override string ToString ()

  Overridden ToString method to provide reservation information in a readable JSON format.

Properties

- IReadOnlyList< Payment > Payments `[get]`

  Gets the list of payments made towards the reservation.
- int ReservationId `[get]`

  Gets the Reservation ID.
- int ClientId `[get]`

  Gets the Client ID associated with the reservation.
- int AccommodationId `[get]`

  Gets the Accommodation ID associated with the reservation.
- AccommodationType AccommodationType `[get, set]`

  Gets or sets the Accommodation Type.
- DateTime CheckInDate `[get, set]`

  Gets or sets the Check-In Date.
- DateTime CheckOutDate `[get, set]`

  Gets or sets the Check-Out Date.
- decimal TotalCost `[get, set]`

  Gets or sets the Total Cost.
- ReservationStatus Status `[get, set]`

  Gets or sets the Reservation Status.
- decimal AmountPaid `[get, set]`

  Gets or sets the Amount Paid towards the reservation.

6.9.1 Detailed Description

Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

Definition at line 26 of file Reservation.cs.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Reservation()

```
SmartStay.Models.Reservation.Reservation (
            int clientId,
            int accommodationId,
            AccommodationType accommodationType,
            DateTime checkInDate,
            DateTime checkOutDate,
            decimal totalCost )  [inline]
```

Constructor to initialize a new reservation with essential details. Validates the input parameters.

Parameters

clientId	The ID of the client.
accommodationId	The ID of the accommodation.
accommodationType	The type of accommodation.
checkInDate	The check-in date.
checkOutDate	The check-out date.
totalCost	The total cost of the reservation.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid.

Definition at line 53 of file Reservation.cs.

6.9.3 Member Function Documentation

6.9.3.1 CheckIn()

```
void SmartStay.Models.Reservation.CheckIn ( )  [inline]
```

Marks the reservation as checked in and updates the status to CheckedIn.

Exceptions

InvalidOperationException	Thrown if the reservation status is not Pending.

Definition at line 166 of file Reservation.cs.

6.9.3.2 CheckOut()

```
void SmartStay.Models.Reservation.CheckOut ( )  [inline]
```

Marks the reservation as checked out and updates the status to CheckedOut.

Exceptions

InvalidOperationException	Thrown if the reservation status is not CheckedIn.

Definition at line 179 of file Reservation.cs.

6.9.3.3 IsFullyPaid()

```
bool SmartStay.Models.Reservation.IsFullyPaid ( )  [inline]
```

Checks if the reservation is fully paid.

Returns

True if the amount paid equals the total cost, otherwise false.

Definition at line 218 of file Reservation.cs.

6.9.3.4 MakePayment()

```
void SmartStay.Models.Reservation.MakePayment (
            decimal paymentAmount,
            PaymentMethod paymentMethod )  [inline]
```

Makes a payment towards the reservation and adds a new Payment object to the payment list.

Definition at line 191 of file Reservation.cs.

6.9.3.5 ToString()

```
override string SmartStay.Models.Reservation.ToString ( )  [inline]
```

Overridden ToString method to provide reservation information in a readable JSON format.

Returns

A JSON string representation of the reservation object.

Definition at line 227 of file Reservation.cs.

6.9.4 Property Documentation

6.9.4.1 AccommodationId

```
int SmartStay.Models.Reservation.AccommodationId  [get]
```

Gets the Accommodation ID associated with the reservation.

Definition at line 92 of file Reservation.cs.

6.9.4.2 AccommodationType

```
AccommodationType SmartStay.Models.Reservation.AccommodationType [get], [set]
```

Gets or sets the Accommodation Type.

Definition at line 97 of file Reservation.cs.

6.9.4.3 AmountPaid

```
decimal SmartStay.Models.Reservation.AmountPaid [get], [set]
```

Gets or sets the Amount Paid towards the reservation.

Definition at line 142 of file Reservation.cs.

6.9.4.4 CheckInDate

```
DateTime SmartStay.Models.Reservation.CheckInDate [get], [set]
```

Gets or sets the Check-In Date.

Definition at line 106 of file Reservation.cs.

6.9.4.5 CheckOutDate

```
DateTime SmartStay.Models.Reservation.CheckOutDate [get], [set]
```

Gets or sets the Check-Out Date.

Definition at line 115 of file Reservation.cs.

6.9.4.6 ClientId

```
int SmartStay.Models.Reservation.ClientId [get]
```

Gets the Client ID associated with the reservation.

Definition at line 87 of file Reservation.cs.

6.9.4.7 Payments

```
IReadOnlyList<Payment> SmartStay.Models.Reservation.Payments [get]
```

Gets the list of payments made towards the reservation.

Definition at line 77 of file Reservation.cs.

6.9.4.8 ReservationId

```
int SmartStay.Models.Reservation.ReservationId [get]
```

Gets the Reservation ID.

Definition at line 82 of file Reservation.cs.

6.9.4.9 Status

```
ReservationStatus SmartStay.Models.Reservation.Status [get], [set]
```

Gets or sets the Reservation Status.

Definition at line 133 of file Reservation.cs.

6.9.4.10 TotalCost

```
decimal SmartStay.Models.Reservation.TotalCost [get], [set]
```

Gets or sets the Total Cost.

Definition at line 124 of file Reservation.cs.

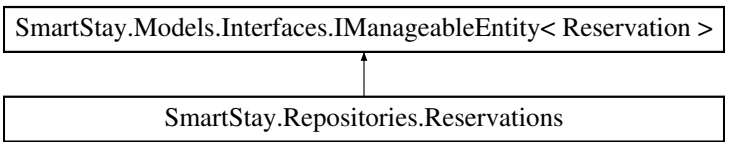
The documentation for this class was generated from the following file:

- Reservation.cs

6.10 SmartStay.Repositories.Reservations Class Reference

Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

Inheritance diagram for SmartStay.Repositories.Reservations:



Public Member Functions

- bool Add (Reservation reservation)

  Attempts to add a new reservation to the collection.
- bool Remove (Reservation reservation)

  Removes a reservation from the collection.
- void Import (string data)

  Imports reservations from a JSON string into the collection, replacing any existing reservations with the same ID.
- string Export ()

  Exports the current list of reservations to a JSON string.
- Reservation? FindReservationById (int reservationId)

  Finds a reservation by its unique ID.
- IEnumerable< Reservation > FindReservationsByClientId (int clientId)

  Finds all reservations associated with a client by their unique client ID.
- IEnumerable< Reservation > FindReservationsByAccommodationId (int accommodationId)

  Finds all reservations associated with an accommodation by its unique accommodation ID.
- IReadOnlyCollection< Reservation > GetAllReservations ()

  Retrieves all the reservations in the collection.
- int CountReservations ()

  Counts the number of reservations in the collection.

Public Member Functions inherited from SmartStay.Models.Interfaces.IManageableEntity< Reservation >

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- void Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.

6.10.1 Detailed Description

Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

Definition at line 25 of file Reservations.cs.

6.10.2 Member Function Documentation

6.10.2.1 Add()

```
bool SmartStay.Repositories.Reservations.Add (
            Reservation reservation )  [inline]
```

Attempts to add a new reservation to the collection.

Parameters

reservation	The Reservation to add to the collection.

Returns

`true` if the reservation was successfully added to the collection; `false` if a reservation with the same ID already exists in the collection.

Exceptions

ArgumentNullException	Thrown if *reservation* is `null`.

Definition at line 43 of file Reservations.cs.

6.10.2.2 CountReservations()

```
int SmartStay.Repositories.Reservations.CountReservations ( )  [inline]
```

Counts the number of reservations in the collection.

Returns

The number of reservations in the collection.

Definition at line 167 of file Reservations.cs.

6.10.2.3 Export()

```
string SmartStay.Repositories.Reservations.Export ( )  [inline]
```

Exports the current list of reservations to a JSON string.

Returns

A JSON string representation of the reservations in the collection.

Definition at line 106 of file Reservations.cs.

6.10.2.4 FindReservationById()

```
Reservation?  SmartStay.Repositories.Reservations.FindReservationById (
            int reservationId )  [inline]
```

Finds a reservation by its unique ID.

Parameters

reservationId	The unique ID of the reservation to find.

Returns

Returns the Reservation object if found; otherwise, `null`.

Definition at line 118 of file Reservations.cs.

6.10.2.5 FindReservationsByAccommodationId()

```
IEnumerable< Reservation > SmartStay.Repositories.Reservations.FindReservationsByAccommodationId (
            int accommodationId )  [inline]
```

Finds all reservations associated with an accommodation by its unique accommodation ID.

Parameters

accommodationId	The unique ID of the accommodation whose reservations to find.

Returns

A list of Reservation objects for the given accommodation. Returns an empty list if no reservations are found.

Definition at line 142 of file Reservations.cs.

6.10.2.6 FindReservationsByClientId()

```
IEnumerable< Reservation > SmartStay.Repositories.Reservations.FindReservationsByClientId (
            int clientId )  [inline]
```

Finds all reservations associated with a client by their unique client ID.

Parameters

clientId	The unique ID of the client whose reservations to find.

Returns

A list of Reservation objects for the given client.

Definition at line 129 of file Reservations.cs.

6.10.2.7 GetAllReservations()

```
IReadOnlyCollection< Reservation > SmartStay.Repositories.Reservations.GetAllReservations ( )
[inline]
```

Retrieves all the reservations in the collection.

Returns

A read-only collection of Reservation objects.

Returns a copy of the internal dictionary's values as a list to prevent external modification.

Definition at line 156 of file Reservations.cs.

6.10.2.8 Import()

```
void SmartStay.Repositories.Reservations.Import (
            string data )  [inline]
```

Imports reservations from a JSON string into the collection, replacing any existing reservations with the same ID.

Parameters

data	The JSON string containing the list of reservations.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 86 of file Reservations.cs.

6.10.2.9 Remove()

```
bool SmartStay.Repositories.Reservations.Remove (
            Reservation reservation )  [inline]
```

Removes a reservation from the collection.

Parameters

reservation	The Reservation to remove from the collection.

Returns

`true` if the reservation was successfully removed from the collection; `false` if the reservation was not found.

Exceptions

ArgumentNullException	Thrown if *reservation* is `null`.

Definition at line 68 of file Reservations.cs.

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

- Reservations.cs

Chapter 7

File Documentation

7.1 Accommodation.cs File Reference

Data Structures

- class SmartStay.Models.Accommodation

  Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

7.2 Accommodation.cs

Go to the documentation of this file.

```
00001
00011 using System.Text.Json;
00012 using System.Text.Json.Serialization;
00013 using SmartStay.Models.Enums;
00014 using SmartStay.Utilities;
00015 using SmartStay.Validation;
00016
00021 namespace SmartStay.Models
00022 {
00028 [JsonConverter(typeof(AccommodationConverter))]
00029 public class Accommodation
00030 {
00031     static int _lastAccommodationId = 0; // Last assigned accommodation ID
00032     static readonly JsonSerializerOptions _jsonOptions = new() { WriteIndented = true }; // JSON
      Serializer options
00033
00034     readonly int _id;                                          // ID of the accommodation
00035     AccommodationType _type;                                   // Type of accommodation
      (Hotel, House, etc.)
00036     string _name;                                              // Name of the accommodation
00037     string _address;                                           // Address of the accommodation
00038     decimal _pricePerNight;                                    // Price per night for the
      accommodation
00039     readonly List<(DateTime Start, DateTime End)> _reservedDates = []; // Booked date ranges
00040
```

```
00059     public Accommodation(AccommodationType type, string name, string address, decimal pricePerNight)
00060     {
00061         if (!Validator.IsValidAccommodationType(type))
00062             throw new ValidationException(ValidationErrorCode.InvalidAccommodationType);
00063         if (!Validator.IsValidAccommodationName(name))
00064             throw new ValidationException(ValidationErrorCode.InvalidAccommodationName);
00065         if (!Validator.IsValidAddress(address))
00066             throw new ValidationException(ValidationErrorCode.InvalidAddress);
00067         if (!Validator.IsValidPrice(pricePerNight))
00068             throw new ValidationException(ValidationErrorCode.InvalidPrice);
00069 
00070         _id = GenerateAccommodationId();
00071         _type = type;
00072         _name = name;
00073         _address = address;
00074         _pricePerNight = pricePerNight;
00075     }
00076 
00080     public int Id => _id;
00081 
00085     public AccommodationType Type
00086     {
00087         get => _type;
00088         set => _type = Validator.ValidateAccommodationType(value);
00089     }
00090 
00094     public string Name
00095     {
00096         get => _name;
00097         set => _name = Validator.ValidateAccommodationName(value);
00098     }
00099 
00103     public string Address
00104     {
00105         get => _address;
00106         set => _address = Validator.ValidateAddress(value);
00107     }
00108 
00112     public decimal PricePerNight
00113     {
00114         get => _pricePerNight;
00115         set => _pricePerNight = Validator.ValidatePrice(value);
00116     }
00117 
00121     public IReadOnlyList<(DateTime Start, DateTime End)> ReservedDates => _reservedDates.AsReadOnly();
00122 
00130     public bool IsAvailable(DateTime startDate, DateTime endDate)
00131     {
00132         if (endDate <= startDate)
00133             throw new ArgumentException("End date must be after the start date.");
00134 
00135         // Perform binary search to find the nearest potential conflict
00136         int index = _reservedDates.BinarySearch((startDate, endDate), new DateRangeComparer());
00137         if (index < 0)
00138             index = ~index; // If not found, BinarySearch returns bitwise complement of the insertion
      index
00139 
00140         // Check previous and next entries for potential overlaps
00141         if (index > 0 && _reservedDates[index - 1].End > startDate)
00142         {
00143             return false; // Overlaps with previous booking
00144         }
00145         if (index < _reservedDates.Count && _reservedDates[index].Start < endDate)
00146         {
00147             return false; // Overlaps with next booking
00148         }
00149 
00150         return true; // No overlap found, accommodation is available
00151     }
00152 
00164     public bool AddReservation(DateTime startDate, DateTime endDate)
00165     {
00166         if (!IsAvailable(startDate, endDate))
00167         {
00168             return false; // Not available, booking cannot be added
00169         }
00170 
00171         // Find the correct position to insert the new booking using binary search
00172         int index = _reservedDates.BinarySearch((startDate, endDate), new DateRangeComparer());
00173         if (index < 0)
00174             index = ~index; // If not found, BinarySearch returns the bitwise complement of the index
00175 
00176         _reservedDates.Insert(index, (startDate, endDate)); // Insert at the correct position
00177         return true;                                        // Booking added successfully
00178     }
00179 
00187     public decimal CalculateTotalCost(DateTime startDate, DateTime endDate)
```

```
00188     {
00189         if (endDate <= startDate)
00190         {
00191             throw new ArgumentException("End date must be after the start date.");
00192         }
00193 
00194         int nights = (endDate - startDate).Days;
00195         return nights * _pricePerNight;
00196     }
00197 
00203     private static int GenerateAccommodationId()
00204     {
00205         // Check if the current value exceeds the max limit of int (2,147,483,647)
00206         if (_lastAccommodationId >= int.MaxValue)
00207         {
00208             throw new InvalidOperationException("Accommodation ID limit exceeded.");
00209         }
00210 
00211         return Interlocked.Increment(ref _lastAccommodationId);
00212     }
00213 
00218     public override string ToString()
00219     {
00220         return JsonSerializer.Serialize(this, _jsonOptions);
00221     }
00222 }
00223 }
```

7.3 Client.cs File Reference

Data Structures

- class SmartStay.Models.Client

  Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

7.4 Client.cs

Go to the documentation of this file.

```
00001 
00011 using System.Text.Encodings.Web;
00012 using System.Text.Json;
00013 using SmartStay.Models.Enums;
00014 using SmartStay.Validation;
00015 
00020 namespace SmartStay.Models
00021 {
00029 public class Client
00030 {
00031     static int _lastClientId = 0; // Last assigned client ID
00032     static readonly JsonSerializerOptions _jsonOptions =
00033         new() { WriteIndented = true,
00034                 Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping }; // JSON Serializer options
00035 
00036     readonly int _id;                                          // ID of the client
00037     string _firstName;                                         // First name of the client
00038     string _lastName;                                          // Last name of the client
00039     string _email;                                             // Email address of the client
00040     string _phoneNumber = string.Empty;                        // Phone number of the client
```

```
00041     string _address = string.Empty;                              // Address of the client
00042     PaymentMethod _preferredPaymentMethod = PaymentMethod.None; // Preferred payment method of the
      client
00043
00052     public Client(string firstName, string lastName, string email)
00053     {
00054         if (!Validator.IsValidName(firstName))
00055             throw new ValidationException(ValidationErrorCode.InvalidName);
00056         if (!Validator.IsValidName(lastName))
00057             throw new ValidationException(ValidationErrorCode.InvalidName);
00058         if (!Validator.IsValidEmail(email))
00059             throw new ValidationException(ValidationErrorCode.InvalidEmail);
00060
00061         _id = GenerateClientId();
00062         _firstName = firstName;
00063         _lastName = lastName;
00064         _email = email;
00065     }
00066
00077     public Client(string firstName, string lastName, string email, string phoneNumber, string address)
00078         : this(firstName, lastName, email)
00079     {
00080         if (!Validator.IsValidPhoneNumber(phoneNumber))
00081             throw new ValidationException(ValidationErrorCode.InvalidPhoneNumber);
00082         if (!Validator.IsValidAddress(address))
00083             throw new ValidationException(ValidationErrorCode.InvalidAddress);
00084
00085         _phoneNumber = phoneNumber;
00086         _address = address;
00087     }
00088
00099     public Client(string firstName, string lastName, string email, string phoneNumber, string address,
00100                   PaymentMethod preferredPaymentMethod)
00101         : this(firstName, lastName, email, phoneNumber, address)
00102     {
00103         if (!Validator.IsValidPaymentMethod(preferredPaymentMethod))
00104             throw new ValidationException(ValidationErrorCode.InvalidPaymentMethod);
00105
00106         _preferredPaymentMethod = preferredPaymentMethod;
00107     }
00108
00112     public int Id => _id;
00113
00118     public string FirstName
00119     {
00120         get => _firstName;
00121         set => _firstName = Validator.ValidateName(value);
00122     }
00123
00128     public string LastName
00129     {
00130         get => _lastName;
00131         set => _lastName = Validator.ValidateName(value);
00132     }
00133
00138     public string Email
00139     {
00140         get => _email;
00141         set => _email = Validator.ValidateEmail(value);
00142     }
00143
00148     public string PhoneNumber
00149     {
00150         get => _phoneNumber;
00151         set => _phoneNumber = Validator.ValidatePhoneNumber(value);
00152     }
00153
00158     public string Address
00159     {
00160         get => _address;
00161         set => _address = Validator.ValidateAddress(value);
00162     }
00163
00168     public PaymentMethod PreferredPaymentMethod
00169     {
00170         get => _preferredPaymentMethod;
00171         set => _preferredPaymentMethod = Validator.ValidatePaymentMethod(value);
00172     }
00173
00179     private static int GenerateClientId()
00180     {
00181         // Check if the current value exceeds the max limit of int (2,147,483,647)
00182         if (_lastClientId >= int.MaxValue)
00183         {
00184             throw new InvalidOperationException("Client ID limit exceeded.");
00185         }
00186
```

```
00187         return Interlocked.Increment(ref _lastClientId);
00188     }
00189
00194     public override string ToString()
00195     {
00196         return JsonSerializer.Serialize(this, _jsonOptions);
00197     }
00198 }
00199 }
```

7.5 AccommodationType.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.
- namespace SmartStay.Models.Enums

  This namespace contains enumerations related to accommodation types used within the SmartStay application.

Enumerations

- enum SmartStay.Models.Enums.AccommodationType { SmartStay.Models.Enums.Hotel , SmartStay.Models.Enums.House , SmartStay.Models.Enums.Apartment , SmartStay.Models.Enums.Villa , SmartStay.Models.Enums.BedAndBreakfast , SmartStay.Models.Enums.Hostel , SmartStay.Models.Enums.Resort , SmartStay.Models.Enums.Cottage , SmartStay.Models.Enums.Cabin , SmartStay.Models.Enums.Guesthouse , SmartStay.Models.Enums.Chalet , SmartStay.Models.Enums.Lodge }

  Enumeration representing different types of accommodations available for booking.

7.6 AccommodationType.cs

Go to the documentation of this file.

```
00001
00010
00014 namespace SmartStay.Models.Enums
00015 {
00019 public enum AccommodationType
00020 {
00024     Hotel,
00025
00029     House,
00030
00035     Apartment,
00036
00040     Villa,
00041
00046     BedAndBreakfast,
00047
00052     Hostel,
00053
00058     Resort,
00059
00063     Cottage,
00064
00068     Cabin,
00069
00074     Guesthouse,
00075
00080     Chalet,
00081
00085     Lodge
00086 }
00087 }
```

7.7 PaymentMethod.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.
- namespace SmartStay.Models.Enums

  This namespace contains enumerations related to accommodation types used within the SmartStay application.

Enumerations

- enum SmartStay.Models.Enums.PaymentMethod { SmartStay.Models.Enums.None , SmartStay.Models.Enums.PayPal , SmartStay.Models.Enums.MultiBanco , SmartStay.Models.Enums.BankTransfer }

  Enumeration representing the possible payment methods available for transactions.

7.8 PaymentMethod.cs

Go to the documentation of this file.

```
00001
00010
00014 namespace SmartStay.Models.Enums
00015 {
00019 public enum PaymentMethod
00020 {
00024     None,
00025
00029     PayPal,
00030
00034     MultiBanco,
00035
00039     BankTransfer
00040 }
00041 }
```

7.9 PaymentStatus.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.
- namespace SmartStay.Models.Enums

  This namespace contains enumerations related to accommodation types used within the SmartStay application.

Enumerations

- enum SmartStay.Models.Enums.PaymentStatus { SmartStay.Models.Enums.Unpaid , SmartStay.Models.Enums.Pending , SmartStay.Models.Enums.Completed , SmartStay.Models.Enums.PartiallyPaid , SmartStay.Models.Enums.Rejected , SmartStay.Models.Enums.Refunded , SmartStay.Models.Enums.Cancelled }

  Enumerator representing payment status.

7.10 PaymentStatus.cs

Go to the documentation of this file.

```
00001
00010
00014 namespace SmartStay.Models.Enums
00015 {
00019 public enum PaymentStatus
00020 {
00024     Unpaid,
00025
00029     Pending,
00030
00034     Completed,
00035
00039     PartiallyPaid,
00040
00044     Rejected,
00045
00049     Refunded,
00050
00054     Cancelled
00055 }
00056 }
```

7.11 ReservationStatus.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.
- namespace SmartStay.Models.Enums

  This namespace contains enumerations related to accommodation types used within the SmartStay application.

Enumerations

- enum SmartStay.Models.Enums.ReservationStatus { SmartStay.Models.Enums.Pending , SmartStay.Models.Enums.CheckedIn , SmartStay.Models.Enums.CheckedOut , SmartStay.Models.Enums.Cancelled , SmartStay.Models.Enums.NoShow , SmartStay.Models.Enums.Confirmed , SmartStay.Models.Enums.Declined }

  Enumeration representing the current status of a reservation.

7.12 ReservationStatus.cs

Go to the documentation of this file.

```
00001
00010
00014 namespace SmartStay.Models.Enums
00015 {
00019 public enum ReservationStatus
00020 {
00024     Pending,
00025
00029     CheckedIn,
00030
00034     CheckedOut,
00035
00039     Cancelled,
00040
00044     NoShow,
00045
00049     Confirmed,
00050
00054     Declined
00055 }
00056 }
```

7.13 ManageableEntity.cs File Reference

Data Structures

- interface SmartStay.Models.Interfaces.IManageableEntity< in T >

  Defines the IManageableEntity<T> interface for managing a collection of entities of type T . This interface standardizes methods for adding, removing, importing, and exporting entities.

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.
- namespace SmartStay.Models.Interfaces

  This namespace contains interfaces used within the SmartStay application.

7.14 ManageableEntity.cs

Go to the documentation of this file.

```
00001
00014
00018 namespace SmartStay.Models.Interfaces
00019 {
00026 public interface IManageableEntity<in T>
00027 {
00033     bool Add(T item);
00034
00040     bool Remove(T item);
00041
00046     void Import(string data);
00047
00052     string Export();
00053 }
00054 }
```

7.15 Payment.cs File Reference

Data Structures

- class SmartStay.Models.Payment

  Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

7.16 Payment.cs

Go to the documentation of this file.

```
00001
00011 using System.Text.Json;
00012 using SmartStay.Models.Enums;
00013 using SmartStay.Validation;
00014
00019 namespace SmartStay.Models
00020 {
00024 public class Payment
00025 {
00026     static int _lastPaymentId = 0;                                                    // Last
      assigned payment ID
00027     static readonly JsonSerializerOptions _jsonOptions = new() { WriteIndented = true }; // JSON
      Serializer options
00028
00029     readonly int _id;               // ID of the payment
00030     readonly int _reservationId;    // ID of the reservation being paid
00031     readonly decimal _amount;       // Amount of the payment
00032     readonly DateTime _date;        // Date the payment was made
00033     readonly PaymentMethod _method; // Payment Method used
00034     PaymentStatus _status;          // Status of the payment
00035
00046     public Payment(int reservationId, decimal amount, DateTime paymentDate, PaymentMethod
      paymentMethod,
00047                    PaymentStatus paymentStatus)
00048     {
00049         if (!Validator.IsValidPaymentAmount(amount))
00050             throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
00051         if (!Validator.IsValidPaymentMethod(paymentMethod))
00052             throw new ValidationException(ValidationErrorCode.InvalidPaymentMethod);
00053         if (!Validator.IsValidPaymentStatus(paymentStatus))
00054             throw new ValidationException(ValidationErrorCode.InvalidPaymentStatus);
00055
00056         _id = GeneratePaymentId();
00057         _reservationId = reservationId;
00058         _amount = amount;
00059         _date = paymentDate;
00060         _method = paymentMethod;
00061         _status = paymentStatus;
00062     }
00063
00067     public int Id => _id;
00068
00072     public int ReservationId => _reservationId;
00073
00077     public decimal Amount => _amount;
00078
00082     public DateTime Date => _date;
00083
00087     public PaymentMethod Method => _method;
00088
00096     public PaymentStatus Status
00097     {
00098
00099         get => _status;
00100         set => _status = Validator.ValidatePaymentStatus(value);
00101     }
00102
00108     private static int GeneratePaymentId()
00109     {
00110         // Check if the current value exceeds the max limit of int (2,147,483,647)
00111         if (_lastPaymentId >= int.MaxValue)
00112         {
00113             throw new InvalidOperationException("Client ID limit exceeded.");
00114         }
00115
00116         return Interlocked.Increment(ref _lastPaymentId);
00117     }
00118
00123     public override string ToString()
00124     {
00125         return JsonSerializer.Serialize(this, _jsonOptions);
00126     }
00127 }
00128 }
```

7.17 Reservation.cs File Reference

Data Structures

- class SmartStay.Models.Reservation

  Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

Namespaces

- namespace SmartStay
- namespace SmartStay.Models

  The `SmartStay.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

7.18 Reservation.cs

Go to the documentation of this file.

```
00001
00011 using System.Text.Json;
00012 using SmartStay.Models.Enums;
00013 using SmartStay.Validation;
00014
00019 namespace SmartStay.Models
00020 {
00026 public class Reservation
00027 {
00028     static int _lastReservationId = 0;                                                    // Last
      assigned reservation ID
00029     static readonly JsonSerializerOptions _jsonOptions = new() { WriteIndented = true }; // JSON
      Serializer options
00030
00031     readonly int _reservationId;                          // ID of the reservation
00032     readonly int _clientId;                               // ID of the client making the reservation
00033     readonly int _accommodationId;                        // ID of the accommodation
00034     AccommodationType _accommodationType;                 // Type of accommodation (e.g., Room,
      Suite, etc.)
00035     DateTime _checkInDate;                                // Check-in date for the reservation
00036     DateTime _checkOutDate;                               // Check-out date for the reservation
00037     ReservationStatus _status = ReservationStatus.Pending; // Current reservation status
00038     decimal _totalCost;                                   // Total cost of the reservation
00039     decimal _amountPaid = 0;                              // Amount paid towards the reservation
00040     readonly List<Payment> _payments = [];                // List of payments made for the
      reservation
00041
00053     public Reservation(int clientId, int accommodationId, AccommodationType accommodationType,
      DateTime checkInDate,
00054                        DateTime checkOutDate, decimal totalCost)
00055     {
00056         if (clientId <= 0)
00057             throw new ValidationException(ValidationErrorCode.InvalidId);
00058         if (accommodationId <= 0)
00059             throw new ValidationException(ValidationErrorCode.InvalidId);
00060         if (!Validator.IsValidDateRange(checkInDate, checkOutDate))
00061             throw new ValidationException(ValidationErrorCode.InvalidDateRange);
00062         if (totalCost < 0)
00063             throw new ValidationException(ValidationErrorCode.InvalidTotalCost);
00064
00065         _reservationId = GenerateReservationId();
00066         _clientId = clientId;
00067         _accommodationId = accommodationId;
00068         _accommodationType = accommodationType;
00069         _checkInDate = checkInDate;
00070         _checkOutDate = checkOutDate;
00071         _totalCost = totalCost;
00072     }
00073
00077     public IReadOnlyList<Payment> Payments => _payments.AsReadOnly();
00078
00082     public int ReservationId => _reservationId;
```

```
00083
00087     public int ClientId => _clientId;
00088
00092     public int AccommodationId => _accommodationId;
00093
00097     public AccommodationType AccommodationType
00098     {
00099         get => _accommodationType;
00100         set => _accommodationType = Validator.ValidateAccommodationType(value);
00101     }
00102
00106     public DateTime CheckInDate
00107     {
00108         get => _checkInDate;
00109         set => _checkInDate = Validator.ValidateCheckInDate(value);
00110     }
00111
00115     public DateTime CheckOutDate
00116     {
00117         get => _checkOutDate;
00118         set => _checkOutDate = Validator.ValidateCheckOutDate(value, _checkInDate);
00119     }
00120
00124     public decimal TotalCost
00125     {
00126         get => _totalCost;
00127         set => _totalCost = Validator.ValidateTotalCost(value);
00128     }
00129
00133     public ReservationStatus Status
00134     {
00135         get => _status;
00136         set => _status = Validator.ValidateReservationStatus(value);
00137     }
00138
00142     public decimal AmountPaid
00143     {
00144         get => _amountPaid;
00145         set => _amountPaid = Validator.ValidatePayment(value);
00146     }
00147
00153     private static int GenerateReservationId()
00154     {
00155         if (_lastReservationId >= int.MaxValue)
00156         {
00157             throw new InvalidOperationException("Reservation ID limit exceeded.");
00158         }
00159         return Interlocked.Increment(ref _lastReservationId);
00160     }
00161
00166     public void CheckIn()
00167     {
00168         if (_status != ReservationStatus.Pending)
00169         {
00170             throw new InvalidOperationException("Reservation must be in Pending status to check in.");
00171         }
00172         _status = ReservationStatus.CheckedIn;
00173     }
00174
00179     public void CheckOut()
00180     {
00181         if (_status != ReservationStatus.CheckedIn)
00182         {
00183             throw new InvalidOperationException("Reservation must be in CheckedIn status to check
      out.");
00184         }
00185         _status = ReservationStatus.CheckedOut;
00186     }
00187
00191     public void MakePayment(decimal paymentAmount, PaymentMethod paymentMethod)
00192     {
00193         if (paymentAmount <= 0)
00194             throw new InvalidOperationException("Payment amount must be greater than zero.");
00195         if (IsFullyPaid())
00196             throw new InvalidOperationException("Reservation is already fully paid.");
00197         if (paymentAmount > _totalCost - _amountPaid)
00198             throw new InvalidOperationException("Payment is more than total required.");
00199
00200         // Validate the payment method
00201         if (!Validator.IsValidPaymentMethod(paymentMethod))
00202         {
00203             throw new ValidationException(ValidationErrorCode.InvalidPaymentMethod);
00204         }
00205
00206         // Create a new Payment instance and add it to the list
00207         var payment = new Payment(_reservationId, paymentAmount, DateTime.Now, paymentMethod,
      PaymentStatus.Completed);
```

```
00208         _payments.Add(payment);
00209
00210         // Update the amount paid
00211         _amountPaid += paymentAmount;
00212     }
00213
00218     public bool IsFullyPaid()
00219     {
00220         return _amountPaid >= _totalCost;
00221     }
00222
00227     public override string ToString()
00228     {
00229         return JsonSerializer.Serialize(this, _jsonOptions);
00230     }
00231 }
00232 }
```

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

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

Go to the documentation of this file.

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
      FrameworkDisplayName = ".NET 8.0")]
```

7.21 SmartStay.AssemblyInfo.cs File Reference

7.22 SmartStay.AssemblyInfo.cs

Go to the documentation of this file.

```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
      System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+a0e24edd77c90e5c9d4886deff3bd54f973cb67d")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023
```

7.23 SmartStay.GlobalUsings.g.cs File Reference

7.24 SmartStay.GlobalUsings.g.cs

Go to the documentation of this file.

```
00001 // <auto-generated/>
00002 global using global::System;
00003 global using global::System.Collections.Generic;
00004 global using global::System.IO;
00005 global using global::System.Linq;
00006 global using global::System.Net.Http;
00007 global using global::System.Threading;
00008 global using global::System.Threading.Tasks;
```

7.25 Program.cs File Reference

Data Structures

- class **SmartStay.Program**

  The Program class is the main entry point for the SmartStay application. This application is designed for managing tourist accommodations, including functionalities for client management, reservations, and check-ins.

Namespaces

- namespace SmartStay

7.26 Program.cs

Go to the documentation of this file.

```
00001
00020 using SmartStay.Models.Enums;
00021 using SmartStay.Models;
00022 using SmartStay.Services;
00023 using SmartStay.Validation;
00024 using SmartStay.Repositories;
00025
00026 namespace SmartStay
00027 {
00033 internal static class Program
00034 {
00035     internal static void Main()
00036     {
00037         // Create and add clients with exception handling
00038         try
00039         {
00040             var client1 = new Client("John", "Doe", "john.doe@example.com");
00041             var client2 = new Client("Jane", "Smith", "jane.smith@example.com", "+3515556482097", "Foo
      Address");
00042
00043             Console.WriteLine($"Client 1 added: {BookingManager.AddClient(client1)}");
00044             Console.WriteLine($"Client 2 added: {BookingManager.AddClient(client2)}");
00045         }
00046         catch (ValidationException ex)
00047         {
00048             Console.WriteLine($"Failed to create client: {ex.Message} (Code: {ex.ErrorCode})");
00049         }
00050
00051         // Retrieve and display client details
00052         var foundClient1 = BookingManager.FindClientById(1);
00053         var foundClient2 = BookingManager.FindClientById(2);
00054         Console.WriteLine($"Found Client 1: {foundClient1}");
00055         Console.WriteLine($"Found Client 2: {foundClient2}");
00056
00057         // Create and add accommodations with exception handling
```

```
00058         try
00059         {
00060             var accommodation1 = new Accommodation(AccommodationType.Hotel, "Grand Hotel", "123 Main
      Street", 100.00m);
00061             var accommodation2 = new Accommodation(AccommodationType.House, "Cozy Cottage", "456 Elm
      Street", 80.00m);
00062 
00063             Console.WriteLine($"Accommodation 1 added:
      {BookingManager.AddAccommodation(accommodation1)}");
00064             Console.WriteLine($"Accommodation 2 added:
      {BookingManager.AddAccommodation(accommodation2)}");
00065         }
00066         catch (ValidationException ex)
00067         {
00068             Console.WriteLine($"Failed to create accommodation: {ex.Message} (Code: {ex.ErrorCode})");
00069         }
00070 
00071         // Retrieve and display accommodation details
00072         var foundAccommodation1 = BookingManager.FindAccommodationById(1);
00073         var foundAccommodation2 = BookingManager.FindAccommodationById(2);
00074         Console.WriteLine($"Found Accommodation 1: {foundAccommodation1}");
00075         Console.WriteLine($"Found Accommodation 2: {foundAccommodation2}");
00076 
00077         // Create reservations and add to accommodations
00078         try
00079         {
00080             if (foundClient1 != null && foundAccommodation1 != null)
00081             {
00082                 var reservation1 = new Reservation(foundClient1.Id, foundAccommodation1.Id,
      AccommodationType.Hotel,
00083                                                    DateTime.Now.AddDays(1), DateTime.Now.AddDays(3),
      200.00m);
00084                 BookingManager.AddReservation(reservation1);
00085 
00086                 // Add reservation to accommodation1
00087                 bool added = foundAccommodation1.AddReservation(reservation1.CheckInDate,
      reservation1.CheckOutDate);
00088                 Console.WriteLine($"Reservation 1 added to Accommodation 1: {added}");
00089                 if (added)
00090                 {
00091                     Console.WriteLine($"Accommodation 1: {foundAccommodation1}");
00092                 }
00093             }
00094 
00095             if (foundClient2 != null && foundAccommodation2 != null)
00096             {
00097                 var reservation2 = new Reservation(foundClient2.Id, foundAccommodation2.Id,
      AccommodationType.House,
00098                                                    DateTime.Now.AddDays(2), DateTime.Now.AddDays(5),
      240.00m);
00099                 BookingManager.AddReservation(reservation2);
00100 
00101                 // Add reservation to accommodation2
00102                 bool added = foundAccommodation2.AddReservation(reservation2.CheckInDate,
      reservation2.CheckOutDate);
00103                 Console.WriteLine($"Reservation 2 added to Accommodation 2: {added}");
00104                 if (added)
00105                 {
00106                     Console.WriteLine($"Accommodation 2: {foundAccommodation2}");
00107                 }
00108             }
00109         }
00110         catch (ValidationException ex)
00111         {
00112             Console.WriteLine($"Failed to create reservation: {ex.Message} (Code: {ex.ErrorCode})");
00113         }
00114 
00115         // Check accommodation availability after adding reservations
00116         if (foundAccommodation1 != null)
00117         {
00118             Console.WriteLine("\nChecking availability for Accommodation 1:");
00119 
00120             // Check availability on dates outside reserved range
00121             var checkDate1Start = DateTime.Now.AddDays(4);
00122             var checkDate1End = DateTime.Now.AddDays(6);
00123             var isAvailable1 = foundAccommodation1.IsAvailable(checkDate1Start, checkDate1End);
00124             Console.WriteLine($"Availability from {checkDate1Start} to {checkDate1End}:
      {isAvailable1}");
00125 
00126             // Check availability on dates overlapping with a reservation
00127             var checkDate2Start = DateTime.Now.AddDays(2);
00128             var checkDate2End = DateTime.Now.AddDays(4);
00129             var isAvailable2 = foundAccommodation1.IsAvailable(checkDate2Start, checkDate2End);
00130             Console.WriteLine($"Availability from {checkDate2Start} to {checkDate2End}:
      {isAvailable2}");
00131         }
00132 
```

```
00133         if (foundAccommodation2 != null)
00134         {
00135             Console.WriteLine("\nChecking availability for Accommodation 2:");
00136 
00137             // Check availability on dates outside reserved range
00138             var checkDate3Start = DateTime.Now.AddDays(6);
00139             var checkDate3End = DateTime.Now.AddDays(8);
00140             var isAvailable3 = foundAccommodation2.IsAvailable(checkDate3Start, checkDate3End);
00141             Console.WriteLine($"Availability from {checkDate3Start} to {checkDate3End}:
      {isAvailable3}");
00142 
00143             // Check availability on dates overlapping with a reservation
00144             var checkDate4Start = DateTime.Now.AddDays(3);
00145             var checkDate4End = DateTime.Now.AddDays(5);
00146             var isAvailable4 = foundAccommodation2.IsAvailable(checkDate4Start, checkDate4End);
00147             Console.WriteLine($"Availability from {checkDate4Start} to {checkDate4End}:
      {isAvailable4}");
00148         }
00149     }
00150 }
00151 }
```

7.27 Accommodations.cs File Reference

Data Structures

- class SmartStay.Repositories.Accommodations

  Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.

Namespaces

- namespace SmartStay
- namespace SmartStay.Repositories

  The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

7.28 Accommodations.cs

Go to the documentation of this file.

```
00001 
00013 using SmartStay.Models;
00014 using SmartStay.Models.Interfaces;
00015 using SmartStay.Utilities;
00016 
00021 namespace SmartStay.Repositories
00022 
00023 {
00028 public class Accommodations : IManageableEntity<Accommodation>
00029 {
00033     readonly Dictionary<int, Accommodation> _accommodationDictionary = new();
00034 
00044     public bool Add(Accommodation accommodation)
00045     {
00046         if (accommodation == null)
00047         {
00048             throw new ArgumentNullException(nameof(accommodation), "Accommodation cannot be null");
00049         }
00050 
00051         if (_accommodationDictionary.ContainsKey(accommodation.Id))
00052         {
00053             return false; // Accommodation already exists
00054         }
00055 
00056         _accommodationDictionary[accommodation.Id] = accommodation;
00057         return true; // Accommodation added successfully
00058     }
00059 
00069     public bool Remove(Accommodation accommodation)
```

```
00070     {
00071         if (accommodation == null)
00072         {
00073             throw new ArgumentNullException(nameof(accommodation), "Accommodation cannot be null");
00074         }
00075 
00076         return _accommodationDictionary.Remove(accommodation.Id); // Remove using accommodation ID
00077     }
00078 
00086     public void Import(string data)
00087     {
00088         if (string.IsNullOrEmpty(data))
00089         {
00090             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00091         }
00092 
00093         var accommodations =
00094             JsonHelper.DeserializeFromJson<Accommodation>(data) ??
00095             throw new ArgumentException("Deserialized accommodation data cannot be null",
      nameof(data));
00096 
00097         foreach (var accommodation in accommodations)
00098         {
00099             _accommodationDictionary[accommodation.Id] = accommodation;
00100         }
00101     }
00102 
00107     public string Export()
00108     {
00109         return JsonHelper.SerializeToJson(_accommodationDictionary.Values);
00110     }
00111 
00119     public Accommodation? FindAccommodationById(int accommodationId)
00120     {
00121       _accommodationDictionary.TryGetValue(accommodationId, out Accommodation? accommodation);
00122       return accommodation;
00123     }
00124 
00131     public IReadOnlyCollection<Accommodation> GetAllAccommodations()
00132     {
00133         return _accommodationDictionary.Values.ToList(); // Returns a copy of the accommodation
      collection.
00134     }
00135 
00142     public int CountAccommodations()
00143     {
00144         return _accommodationDictionary.Count;
00145     }
00146 }
00147 }
```

7.29 Clients.cs File Reference

Data Structures

- class SmartStay.Repositories.Clients

  Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.

Namespaces

- namespace SmartStay
- namespace SmartStay.Repositories

  The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

7.30 Clients.cs

Go to the documentation of this file.

```
00001
00011 using SmartStay.Models;
00012 using SmartStay.Models.Interfaces;
00013 using SmartStay.Utilities;
00014
00019 namespace SmartStay.Repositories
00020 {
00025 public class Clients : IManageableEntity<Client>
00026 {
00030     readonly Dictionary<int, Client> _clientDictionary = [];
00031
00043     public bool Add(Client client)
00044     {
00045         if (client == null)
00046         {
00047             throw new ArgumentNullException(nameof(client), "Client cannot be null");
00048         }
00049
00050         if (_clientDictionary.ContainsKey(client.Id))
00051         {
00052             return false; // Client already exists
00053         }
00054
00055         _clientDictionary[client.Id] = client;
00056         return true; // Client added successfully
00057     }
00058
00068     public bool Remove(Client client)
00069     {
00070         if (client == null)
00071         {
00072             throw new ArgumentNullException(nameof(client), "Client cannot be null");
00073         }
00074
00075         return _clientDictionary.Remove(client.Id); // Remove by client ID
00076     }
00077
00084     public void Import(string data)
00085     {
00086         if (string.IsNullOrEmpty(data))
00087         {
00088             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00089         }
00090
00091         var clients = JsonHelper.DeserializeFromJson<Client>(data) ??
00092                       throw new ArgumentException("Deserialized client data cannot be null",
      nameof(data));
00093
00094         foreach (var client in clients)
00095         {
00096             _clientDictionary[client.Id] = client; // Direct insertion for efficiency
00097         }
00098     }
00099
00104     public string Export()
00105     {
00106         return JsonHelper.SerializeToJson(_clientDictionary.Values ?? Enumerable.Empty<Client>());
00107     }
00108
00116     public Client? FindClientById(int id)
00117     {
00118         _clientDictionary.TryGetValue(id, out Client? client);
00119         return client;
00120     }
00121
00131     public IReadOnlyCollection<Client> GetAllClients()
00132     {
00133         return _clientDictionary.Values.ToList(); // Returns a copy of the client collection.
00134     }
00135
00142     public int CountClients()
00143     {
00144         return _clientDictionary.Count;
00145     }
00146 }
00147 }
```

7.31 Reservations.cs File Reference

Data Structures

- class SmartStay.Repositories.Reservations

  Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

Namespaces

- namespace SmartStay
- namespace SmartStay.Repositories

  The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

7.32 Reservations.cs

Go to the documentation of this file.

```
00001
00011 using SmartStay.Models;
00012 using SmartStay.Models.Interfaces;
00013 using SmartStay.Utilities;
00014
00019 namespace SmartStay.Repositories
00020 {
00025 public class Reservations : IManageableEntity<Reservation>
00026 {
00030     readonly Dictionary<int, Reservation> _reservationDictionary = [];
00031
00043     public bool Add(Reservation reservation)
00044     {
00045         if (reservation == null)
00046         {
00047             throw new ArgumentNullException(nameof(reservation), "Reservation cannot be null");
00048         }
00049
00050         if (_reservationDictionary.ContainsKey(reservation.ReservationId))
00051         {
00052             return false; // Reservation already exists
00053         }
00054
00055         _reservationDictionary[reservation.ReservationId] = reservation;
00056         return true; // Reservation added successfully
00057     }
00058
00068     public bool Remove(Reservation reservation)
00069     {
00070         if (reservation == null)
00071         {
00072             throw new ArgumentNullException(nameof(reservation), "Reservation cannot be null");
00073         }
00074
00075         // Remove the reservation using its ID
00076         return _reservationDictionary.Remove(reservation.ReservationId);
00077     }
00078
00086     public void Import(string data)
00087     {
00088         if (string.IsNullOrEmpty(data))
00089         {
00090             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00091         }
00092
00093         var reservations = JsonHelper.DeserializeFromJson<Reservation>(data) ??
00094                            throw new ArgumentException("Deserialized reservation data cannot be null",
      nameof(data));
00095
00096         foreach (var reservation in reservations)
00097         {
00098             _reservationDictionary[reservation.ReservationId] = reservation;
00099         }
00100     }
```

```
00101
00106     public string Export()
00107     {
00108         return JsonHelper.SerializeToJson(_reservationDictionary.Values);
00109     }
00110
00118     public Reservation? FindReservationById(int reservationId)
00119     {
00120       _reservationDictionary.TryGetValue(reservationId, out Reservation? reservation);
00121       return reservation;
00122     }
00123
00129     public IEnumerable<Reservation> FindReservationsByClientId(int clientId)
00130     {
00131         return _reservationDictionary.Values.Where(r => r.ClientId == clientId);
00132     }
00133
00142     public IEnumerable<Reservation> FindReservationsByAccommodationId(int accommodationId)
00143     {
00144         return _reservationDictionary.Values.Where(r => r.AccommodationId == accommodationId);
00145     }
00146
00156     public IReadOnlyCollection<Reservation> GetAllReservations()
00157     {
00158         return _reservationDictionary.Values.ToList(); // Returns a copy of the reservation
      collection.
00159     }
00160
00167     public int CountReservations()
00168     {
00169         return _reservationDictionary.Count;
00170     }
00171 }
00172 }
```

7.33 BookingManager.cs File Reference

Data Structures

- class **SmartStay.Services.BookingManager**

  Provides a static facade for managing clients, reservations, and accommodations in the booking system. This class centralizes all operations for adding, removing, importing, and exporting data for these entities. It interacts with internal repositories to simplify the main API and ensure a standardized approach.

Namespaces

- namespace SmartStay
- namespace SmartStay.Services

  The `SmartStay.Services` namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.

7.34 BookingManager.cs

Go to the documentation of this file.

```
00001
00011 using SmartStay.Models;
00012 using SmartStay.Repositories;
00013
00018 namespace SmartStay.Services
00019 {
00029 public static class BookingManager
00030 {
00031 #region Collections
00032
00036     internal static readonly Clients _clients = new();
00037
00041     internal static readonly Reservations _reservations = new();
```

```
00042
00046     internal static readonly Accommodations _accommodations = new();
00047
00048 #endregion
00049
00050 #region Client Operations
00051
00062     public static bool AddClient(Client client) => _clients.Add(client);
00063
00074     public static bool RemoveClient(Client client) => _clients.Remove(client);
00075
00083     public static void ImportClients(string data) => _clients.Import(data);
00084
00095     public static void ExportClients(string filePath) => File.WriteAllText(filePath,
      _clients.Export());
00096
00101     public static IReadOnlyCollection<Client> GetAllClients() => _clients.GetAllClients();
00102
00113     public static Client? FindClientById(int id) => _clients.FindClientById(id);
00114
00119     public static int CountClients() => _clients.CountClients();
00120
00121 #endregion
00122
00123 #region Reservation Operations
00124
00135     public static bool AddReservation(Reservation reservation) => _reservations.Add(reservation);
00136
00147     public static bool RemoveReservation(Reservation reservation) =>
      _reservations.Remove(reservation);
00148
00156     public static void ImportReservations(string data) => _reservations.Import(data);
00157
00168     public static void ExportReservations(string filePath) => File.WriteAllText(filePath,
      _reservations.Export());
00169
00174     public static IReadOnlyCollection<Reservation> GetAllReservations() =>
      _reservations.GetAllReservations();
00175
00186     public static Reservation? FindReservationById(int id) => _reservations.FindReservationById(id);
00187
00196     public static IEnumerable<Reservation> FindReservationsByClientId(int id) =>
00197         _reservations.FindReservationsByClientId(id);
00198
00207     public static IEnumerable<Reservation> FindReservationsByAccommodationId(int id) =>
00208         _reservations.FindReservationsByAccommodationId(id);
00209
00214     public static int CountReservations() => _reservations.CountReservations();
00215
00216 #endregion
00217
00218 #region Accommodation Operations
00219
00230     public static bool AddAccommodation(Accommodation accommodation) =>
      _accommodations.Add(accommodation);
00231
00242     public static bool RemoveAccommodation(Accommodation accommodation) =>
      _accommodations.Remove(accommodation);
00243
00251     public static void ImportAccommodations(string data) => _accommodations.Import(data);
00252
00263     public static void ExportAccommodations(string filePath) => File.WriteAllText(filePath,
      _accommodations.Export());
00264
00269     public static IReadOnlyCollection<Accommodation> GetAllAccommodations() =>
      _accommodations.GetAllAccommodations();
00270
00281     public static Accommodation? FindAccommodationById(int id) =>
      _accommodations.FindAccommodationById(id);
00282
00287     public static int CountAccommodations() => _accommodations.CountAccommodations();
00288
00289 #endregion
00290 }
00291 }
```

7.35 AccommodationConverter.cs File Reference

Data Structures

- class SmartStay.Utilities.AccommodationConverter

Custom JSON converter for Accommodation objects, used to serialize and deserialize accommodations to and from JSON format. It provides custom handling for the reserved dates and accommodation type.

Namespaces

- namespace SmartStay
- namespace SmartStay.Utilities

  The `SmartStay.Utilities` namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

7.36 AccommodationConverter.cs

Go to the documentation of this file.

```
00001
00010 using System.Text.Json;
00011 using System.Text.Json.Serialization;
00012 using SmartStay.Models;
00013
00019 namespace SmartStay.Utilities
00020 {
00025 public class AccommodationConverter : JsonConverter<Accommodation>
00026 {
00035     public override Accommodation Read(ref Utf8JsonReader reader, Type typeToConvert,
      JsonSerializerOptions options)
00036     {
00037         // Attempt to deserialize the object
00038         var accommodation = JsonSerializer.Deserialize<Accommodation>(ref reader, options);
00039
00040         // If deserialization fails, throw an exception
00041         return accommodation ?? throw new JsonException("Failed to deserialize the Accommodation
      object.");
00042     }
00043
00050     public override void Write(Utf8JsonWriter writer, Accommodation value, JsonSerializerOptions
      options)
00051     {
00052         writer.WriteStartObject();
00053
00054         // Serialize properties of the accommodation
00055         writer.WriteNumber("Id", value.Id);
00056         writer.WriteString("Type", value.Type.ToString());
00057         writer.WriteString("Name", value.Name);
00058         writer.WriteString("Address", value.Address);
00059         writer.WriteNumber("PricePerNight", value.PricePerNight);
00060
00061         // Serialize reserved dates as an array
00062         writer.WriteStartArray("ReservedDates");
00063         foreach (var (Start, End) in value.ReservedDates)
00064         {
00065             writer.WriteStartObject();
00066             writer.WriteString("Start", Start.ToString("yyyy-MM-dd"));
00067             writer.WriteString("End", End.ToString("yyyy-MM-dd"));
00068             writer.WriteEndObject();
00069         }
00070         writer.WriteEndArray();
00071
00072         writer.WriteEndObject();
00073     }
00074 }
00075 }
```

7.37 DateRangeComparer.cs File Reference

Data Structures

- class SmartStay.Utilities.DateRangeComparer

  Implements IComparer<T> to provide comparison logic for tuples of DateTime values representing date ranges. The comparison is done based on the Start date of each range.

Namespaces

- namespace SmartStay
- namespace SmartStay.Utilities

  The `SmartStay.Utilities` namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

7.38 DateRangeComparer.cs

Go to the documentation of this file.

```
00001
00012
00018 namespace SmartStay.Utilities
00019 {
00024 public class DateRangeComparer : IComparer<(DateTime Start, DateTime End)>
00025 {
00037     public int Compare((DateTime Start, DateTime End)x, (DateTime Start, DateTime End)y)
00038     {
00039         return x.Start.CompareTo(y.Start);
00040     }
00041 }
00042 }
```

7.39 JsonHelper.cs File Reference

Data Structures

- class **SmartStay.Utilities.JsonHelper**

  Provides static methods to serialize and deserialize objects to and from JSON format.

Namespaces

- namespace SmartStay
- namespace SmartStay.Utilities

  The `SmartStay.Utilities` namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

7.40 JsonHelper.cs

Go to the documentation of this file.

```
00001
00009 using System.Text.Json;
00010
00016 namespace SmartStay.Utilities
00017 {
00021 public static class JsonHelper
00022 {
00026     private static readonly JsonSerializerOptions _jsonOptions = new() { WriteIndented = true };
00027
00038     public static string SerializeToJson<T>(IEnumerable<T> items)
00039     {
00040         return JsonSerializer.Serialize(items, _jsonOptions);
00041     }
00042
00053     public static List<T> DeserializeFromJson<T>(string json)
00054     {
00055         return JsonSerializer.Deserialize<List<T»(json) ?? new List<T>();
00056     }
00057 }
00058 }
```

7.41 ValidationErrorCodes.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

Enumerations

- enum SmartStay.Validation.ValidationErrorCode {
  SmartStay.Validation.InvalidName = 1001 , SmartStay.Validation.InvalidEmail = 1002 , SmartStay.Validation.InvalidPhoneNumbe
  = 1003 , SmartStay.Validation.InvalidAddress = 1004 ,
  SmartStay.Validation.InvalidPaymentMethod = 1005 , SmartStay.Validation.InvalidAccommodationType =
  1006 , SmartStay.Validation.InvalidId = 1007 , SmartStay.Validation.InvalidDateRange = 1008 ,
  SmartStay.Validation.InvalidDate = 1009 , SmartStay.Validation.InvalidTotalCost = 1010 , SmartStay.Validation.InvalidPaymentVa
  = 1011 , SmartStay.Validation.InvalidReservationStatus = 1012 ,
  SmartStay.Validation.InvalidAccommodationName = 1013 , SmartStay.Validation.InvalidPrice = 1014 ,
  SmartStay.Validation.InvalidPaymentStatus = 1015 }

  Defines error codes for validation failures within the SmartStay application.

7.42 ValidationErrorCodes.cs

Go to the documentation of this file.

```
00001
00014
00020 namespace SmartStay.Validation
00021 {
00025 public enum ValidationErrorCode
00026 {
00030     InvalidName = 1001,
00031
00035     InvalidEmail = 1002,
00036
00040     InvalidPhoneNumber = 1003,
00041
00045     InvalidAddress = 1004,
00046
00050     InvalidPaymentMethod = 1005,
00051
00055     InvalidAccommodationType = 1006,
00056
00060     InvalidId = 1007,
00061
00066     InvalidDateRange = 1008,
00067
00072     InvalidDate = 1009,
00073
00077     InvalidTotalCost = 1010,
00078
00083     InvalidPaymentValue = 1011,
00084
00089     InvalidReservationStatus = 1012,
00090
00094     InvalidAccommodationName = 1013,
00095
00099     InvalidPrice = 1014,
00100
00104     InvalidPaymentStatus = 1015,
00105 }
00106 }
```

7.43 ValidationErrorMessages.cs File Reference

Data Structures

- class **SmartStay.Validation.ValidationErrorMessages**

  Provides error messages corresponding to each ValidationErrorCode value used in client data validation.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

7.44 ValidationErrorMessages.cs

Go to the documentation of this file.

```
00001
00010
00016 namespace SmartStay.Validation
00017 {
00022 public static class ValidationErrorMessages
00023 {
00027     private static readonly Dictionary<ValidationErrorCode, string> ErrorMessages = new() {
00028         { ValidationErrorCode.InvalidName, "The name must be a non-empty string and not exceed 50
      characters." },
00029         { ValidationErrorCode.InvalidEmail, "The email address is invalid or does not match the
      required format." },
00030         { ValidationErrorCode.InvalidPhoneNumber, "The phone number is invalid or empty." },
00031         { ValidationErrorCode.InvalidAddress, "The address is invalid or empty." },
00032         { ValidationErrorCode.InvalidPaymentMethod, "The selected payment method is not valid." },
00033         { ValidationErrorCode.InvalidAccommodationType, "The accommodation type is invalid or not
      recognized." },
00034         { ValidationErrorCode.InvalidId, "The provided ID is invalid or does not exist." },
00035         { ValidationErrorCode.InvalidDateRange,
00036           "The date range is invalid. Ensure the check-out date is later than the check-in date." },
00037         { ValidationErrorCode.InvalidDate, "The date provided is invalid. It must be a valid date in
      the future." },
00038         { ValidationErrorCode.InvalidTotalCost, "The total cost is invalid. It cannot be negative." },
00039         { ValidationErrorCode.InvalidPaymentValue,
00040           "The payment value is invalid. It cannot be negative or greater than the total cost." },
00041         { ValidationErrorCode.InvalidReservationStatus, "The reservation status is invalid or
      unrecognized." },
00042         { ValidationErrorCode.InvalidAccommodationName, "The accommodation name is invalid or empty."
      },
00043         { ValidationErrorCode.InvalidPrice, "The price is invalid. It must be a positive value." }
00044     };
00045
00054     public static string GetErrorMessage(ValidationErrorCode errorCode)
00055     {
00056         return ErrorMessages.TryGetValue(errorCode, out var message) ? message : "Unknown validation
      error.";
00057     }
00058 }
00059 }
```

7.45 ValidationException.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

Functions

- class SmartStay.Validation.ValidationException (ValidationErrorCode errorCode)

  Initializes a new instance of the ValidationException class with a specified validation error code. The error message is derived from ValidationErrorMessages.GetErrorMessage based on the provided error code.

7.46 ValidationException.cs

Go to the documentation of this file.

```
00001
00011
00017 namespace SmartStay.Validation
00018 {
00025 public class ValidationException
00026 (ValidationErrorCode errorCode) : Exception(ValidationErrorMessages.GetErrorMessage(errorCode))
00027 {
00032     public ValidationErrorCode ErrorCode { get; } = errorCode;
00033 }
00034 }
```

7.47 Validator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validator**

  Provides a set of static methods for validating input data in the SmartStay application, including names, emails, phone numbers, addresses, and payment methods.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

7.48 Validator.cs

Go to the documentation of this file.

```
00001
00012 using System.Text.RegularExpressions;
00013 using SmartStay.Models.Enums;
00014
00020 namespace SmartStay.Validation
00021 {
00026 public static class Validator
00027 {
00031     private static readonly string EmailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
00032
00036     private static readonly string PhoneNumberPattern = @"^\+(\d{1,3})\d{7,15}$";
00037
00044     public static string ValidateName(string name)
00045     {
00046         if (!IsValidName(name))
00047         {
00048             throw new ValidationException(ValidationErrorCode.InvalidName);
00049         }
00050         return name;
00051     }
```

```
00052
00059     public static string ValidateAccommodationName(string name)
00060     {
00061         if (!IsValidName(name))
00062         {
00063             throw new ValidationException(ValidationErrorCode.InvalidAccommodationName);
00064         }
00065         return name;
00066     }
00067
00074     public static string ValidateEmail(string email)
00075     {
00076         if (!IsValidEmail(email))
00077         {
00078             throw new ValidationException(ValidationErrorCode.InvalidEmail);
00079         }
00080         return email;
00081     }
00082
00089     public static string ValidatePhoneNumber(string phoneNumber)
00090     {
00091         if (!IsValidPhoneNumber(phoneNumber))
00092         {
00093             throw new ValidationException(ValidationErrorCode.InvalidPhoneNumber);
00094         }
00095         return phoneNumber;
00096     }
00097
00104     public static string ValidateAddress(string address)
00105     {
00106         if (!IsValidAddress(address))
00107         {
00108             throw new ValidationException(ValidationErrorCode.InvalidAddress);
00109         }
00110         return address;
00111     }
00112
00119     public static decimal ValidatePrice(decimal price)
00120     {
00121         if (!IsValidPrice(price))
00122         {
00123             throw new ValidationException(ValidationErrorCode.InvalidPrice);
00124         }
00125         return price;
00126     }
00127
00134     public static decimal ValidatePaymentAmount(decimal amount)
00135     {
00136         if (!IsValidPaymentAmount(amount))
00137         {
00138             throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
00139         }
00140         return amount;
00141     }
00142
00149     public static PaymentStatus ValidatePaymentStatus(PaymentStatus status)
00150     {
00151         if (!IsValidPaymentStatus(status))
00152         {
00153             throw new ValidationException(ValidationErrorCode.InvalidPaymentStatus);
00154         }
00155         return status;
00156     }
00157
00164     public static PaymentMethod ValidatePaymentMethod(PaymentMethod paymentMethod)
00165     {
00166         if (!IsValidPaymentMethod(paymentMethod))
00167         {
00168             throw new ValidationException(ValidationErrorCode.InvalidPaymentMethod);
00169         }
00170         return paymentMethod;
00171     }
00172
00179     public static AccommodationType ValidateAccommodationType(AccommodationType accommodationType)
00180     {
00181         if (!IsValidAccommodationType(accommodationType))
00182         {
00183             throw new ValidationException(ValidationErrorCode.InvalidAccommodationType);
00184         }
00185         return accommodationType;
00186     }
00187
00194     public static DateTime ValidateCheckInDate(DateTime checkInDate)
00195     {
00196         if (!IsValidFutureDate(checkInDate))
00197         {
00198             throw new ValidationException(ValidationErrorCode.InvalidDate);
```

```
00199         }
00200         return checkInDate;
00201     }
00202
00210     public static DateTime ValidateCheckOutDate(DateTime checkOutDate, DateTime checkInDate)
00211     {
00212         if (!IsValidDateRange(checkInDate, checkOutDate))
00213         {
00214             throw new ValidationException(ValidationErrorCode.InvalidDateRange);
00215         }
00216         return checkOutDate;
00217     }
00218
00225     public static decimal ValidateTotalCost(decimal totalCost)
00226     {
00227         if (totalCost < 0)
00228         {
00229             throw new ValidationException(ValidationErrorCode.InvalidTotalCost);
00230         }
00231         return totalCost;
00232     }
00233
00240     public static decimal ValidatePayment(decimal paymentValue)
00241     {
00242         if (paymentValue < 0)
00243         {
00244             throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
00245         }
00246         return paymentValue;
00247     }
00248
00255     public static ReservationStatus ValidateReservationStatus(ReservationStatus status)
00256     {
00257         if (!IsValidReservationStatus(status))
00258         {
00259             throw new ValidationException(ValidationErrorCode.InvalidReservationStatus);
00260         }
00261         return status;
00262     }
00263
00269     public static bool IsValidName(string name) => !string.IsNullOrWhiteSpace(name) && name.Length <=
      50;
00270
00276     public static bool IsValidAccommodationName(string name) => !string.IsNullOrWhiteSpace(name) &&
      name.Length <= 100;
00277
00283     public static bool IsValidEmail(string email) => !string.IsNullOrWhiteSpace(email) &&
      Regex.IsMatch(email,
00284
      EmailPattern);
00285
00291     public static bool IsValidPhoneNumber(string phoneNumber) =>
00292         !string.IsNullOrWhiteSpace(phoneNumber) && Regex.IsMatch(phoneNumber, PhoneNumberPattern);
00293
00299     public static bool IsValidPrice(decimal price) => price > 0;
00300
00306     public static bool IsValidPaymentAmount(decimal amount) => amount > 0;
00307
00313     public static bool IsValidAddress(string address) => !string.IsNullOrWhiteSpace(address);
00314
00320     public static bool IsValidPaymentMethod(PaymentMethod paymentMethod) =>
      Enum.IsDefined(typeof(PaymentMethod),
00321
      paymentMethod);
00322
00328     public static bool IsValidPaymentStatus(PaymentStatus paymentStatus) =>
      Enum.IsDefined(typeof(PaymentStatus),
00329
      paymentStatus);
00330
00337     public static bool IsValidAccommodationType(AccommodationType accommodationType)
00338     {
00339         return Enum.IsDefined(typeof(AccommodationType), accommodationType);
00340     }
00341
00348     public static bool IsValidDateRange(DateTime checkInDate, DateTime checkOutDate)
00349     {
00350         return checkInDate < checkOutDate;
00351     }
00352
00358     public static bool IsValidFutureDate(DateTime date)
00359     {
00360         return date >= DateTime.Today;
00361     }
00362
00369     public static bool IsValidReservationStatus(ReservationStatus status)
00370     {
```

```
00371         return Enum.IsDefined(typeof(ReservationStatus), status);
00372     }
00373 }
00374 }
```

Index