Manage constructions

Generated by Doxygen 1.12.0

Chapter 1

Manage constructions

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 Business_Tier Namespace Reference

Classes

- class Company

  The Enterprise class manages operations related to customers employees, materials and projects.It handles tasks such as registration, update and removal, manages project states and resource usage.Ensures proper validation and exception handling for each action.

5.2 CustomExceptions Namespace Reference

Classes

- class ConfigurationErrorException

  Custom exception used for handling configuration-related errors in the application.

5.3 Data_Layer Namespace Reference

Classes

- class MaterialService

  Class for managing materials used in a project.

5.4 Data_Tier Namespace Reference

Classes

- class Clients

  Singleton class that manages the list of clients. Allows adding, removing, updating and retrieving clients.
- class **ClientsService**

  Service class that manages clients associated with a project.
- class Data

  The `Data` class serves as a centralized data manager for the application. It collects data from various parts of the system, organizes it into lists, and provides methods for saving to and loading from a binary file.
- class Employees

  Singleton class that manages a list of employees. Allows adding, removing, updating and retrieving employees.
- class EmployeesService

  Class that manages employees associated with projects.
- class MaterialInventory

  Singleton class that manages the material inventory. Allows adding, removing, updating, and retrieving materials.
- class Materials

  Singleton class that manages the materials in the system. Allows adding, checking, updating, and retrieving materials.
- class ProjectData

  Class that represents project data, including operations related to clients, employees, and materials.
- class Projects

  Singleton class that manages the projects in the system. Allows adding, removing, updating, and retrieving projects.

5.5 Interface_Tier Namespace Reference

Classes

- interface IClients

  Methods to implement in the clients class.
- interface IEmployees

  Methods to implement in the employees class.
- interface IMaterialInventory

  Methods to implement in the inventory class.
- interface IMaterials

  Methods to implement in the materials class.
- interface IProjects

  Methods to implement in the projects class.

5.6 Object_Tier Namespace Reference

Classes

- class Client

  Represents a client in the system, inheriting from the Person class.
- class Employee

  Represents an employee in the system, inheriting from the Person class.
- class Material

Represents a material with ID, name, unit price, and registration date.

- class MaterialQuantity

  Represents the quantity of a material and the date it was added.

- class Person

  Represents a person with an ID and a name.

- class Project

  Represents a project with information about its status, start date, end date, and ID management.

Enumerations

- enum Status { **NotStart** = 1 , **InProgress** = 2 , **OnHold** = 3 , **Completed** = 4 }

  Enum that defines the possible status of a project.

5.6.1 Enumeration Type Documentation

5.6.1.1 Status

```
enum Object_Tier.Status
```

Enum that defines the possible status of a project.

```
00019     {
00020         NotStart = 1, //In wait list
00021         InProgress = 2, //In Progress
00022         OnHold = 3, //Stopped for a while
00023         Completed = 4  //Complet
00024     }
```

5.7 Presentation_Tier Namespace Reference

Classes

- class **Program**

5.8 Unit_Test Namespace Reference

Classes

- class ClientsTest
- class EmployeeTest

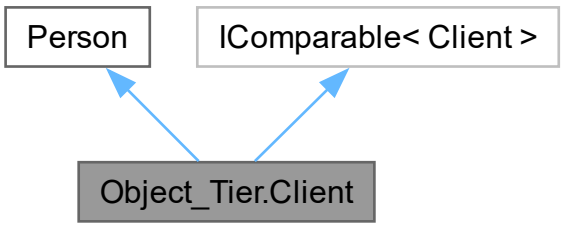
Chapter 6

Class Documentation

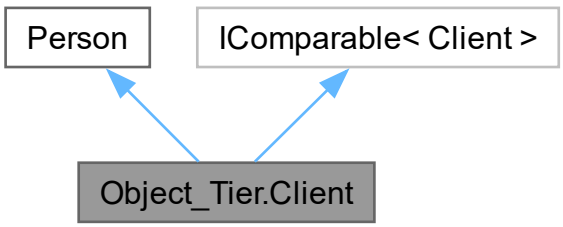
6.1 Object_Tier.Client Class Reference

Represents a client in the system, inheriting from the Person class.

Inheritance diagram for Object_Tier.Client:



Collaboration diagram for Object_Tier.Client:



Public Member Functions

- Client (string name, int contact)

  Initializes a new instance of the Client class with the specified name and contact. Automatically assigns a unique ID.
- override bool Equals (object obj)

  Determines whether the specified object is equal to the current client. Clients are considered equal if their contact information matches.
- override string ToString ()

  Returns a string representation of the client, including ID, name, and contact.
- int CompareTo (Client client)

  Compares the current client to another client based on their name.

Static Public Member Functions

- static Client CreateClient (string name, int contact)

  Creates a new Client instance.
- static bool operator- (Client client1, Client client2)

  Checks if two clients are equal using the "-" operator.
- static bool operator+ (Client client1, Client client2)

  Checks if two clients are not equal using the "+" operator.
- static bool getNextClientId ()

  Increments the static client ID counter.

Properties

- int ContactInfo `[get, set]`

  Gets or sets the client's contact information. One contact can only have 9 numbers.

Properties inherited from Object_Tier.Person

- int Id `[get]`

  Gets the ID of the person.
- string Name `[get, set]`

  Gets or sets the name of the person.

Additional Inherited Members

Protected Member Functions inherited from Object_Tier.Person

- Person (int id, string name)

  Initializes a new instance of the Person class with an ID and a name. The name is automatically formatted.

6.1.1 Detailed Description

Represents a client in the system, inheriting from the Person class.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Client()

```
Object_Tier.Client.Client (
            string name,
            int contact)
```

Initializes a new instance of the Client class with the specified name and contact. Automatically assigns a unique ID.

Parameters

name	The name of the client.
contact	The contact information of the client.

```
00061                                                : base(clientIdCounter++, name) //Send to the
      construct person.
00062         {
00063             ContactInfo = contact;
00064         }
```

6.1.3 Member Function Documentation

6.1.3.1 CompareTo()

```
int Object_Tier.Client.CompareTo (
              Client client)
```

Compares the current client to another client based on their name.

Parameters

client	The client to compare to.

Returns

A value indicating the relative order of the clients.

```
00151         {
00152             return Name.CompareTo(client.Name);
00153         }
```

6.1.3.2 CreateClient()

```
static Client Object_Tier.Client.CreateClient (
              string name,
              int contact)  [static]
```

Creates a new Client instance.

Parameters

name	The name of the client.
contact	The contact information of the client.

Returns

A new Client instance.

```
00114         {
00115             return new Client(name, contact);
00116         }
```

6.1.3.3 Equals()

```
override bool Object_Tier.Client.Equals (
              object obj)
```

Determines whether the specified object is equal to the current client. Clients are considered equal if their contact information matches.

Parameters

obj	The object to compare with the current client.

Returns

True if the objects are equal; otherwise, false.

```
00076         {
00077             if (obj == null)
00078             {
00079                 return false;
00080             }
00081 
00082             if (obj is Client)
00083             {
00084                 Client otherClient = obj as Client;
00085 
00086                 if (contactInfo == otherClient.contactInfo)
00087                 {
00088                     return true;
00089                 }
00090             }
00091 
00092             return false;
00093         }
```

6.1.3.4 getNextClientId()

```
static bool Object_Tier.Client.getNextClientId ()  [static]
```

Increments the static client ID counter.

```
00159         {
00160             clientIdCounter++;
00161             return true;
00162         }
```

6.1.3.5 operator+()

```
static bool Object_Tier.Client.operator+ (
             Client client1,
             Client client2)  [static]
```

Checks if two clients are not equal using the "+" operator.

Parameters

client1	The first client.
client2	The second client.

Returns

True if the clients are not equal; otherwise, false.

```
00141         {
00142             return !(client1 - client2);
00143         }
```

6.1.3.6 operator-()

```
static bool Object_Tier.Client.operator- (
             Client client1,
             Client client2)  [static]
```

Checks if two clients are equal using the "-" operator.

Parameters

client1	The first client.
client2	The second client.

Returns

True if the clients are equal; otherwise, false.

```
00125         {
00126             if (client1.Equals(client2))
00127             {
00128                 return true;
00129             }
00130 
00131             return false;
00132         }
```

6.1.3.7 ToString()

```
override string Object_Tier.Client.ToString ()
```

Returns a string representation of the client, including ID, name, and contact.

Returns

A string representation of the client.

```
00100         {
00101             return Id + " " + Name + " " + ContactInfo;
00102         }
```

6.1.4 Property Documentation

6.1.4.1 ContactInfo

```
int Object_Tier.Client.ContactInfo  [get], [set]
```

Gets or sets the client's contact information. One contact can only have 9 numbers.

```
00041         {
00042             set
00043             {
00044                 if (value >= 9)
00045                 {
00046                     contactInfo = value;
00047                 }
00048             }
00049             get { return contactInfo; }
00050         }
```

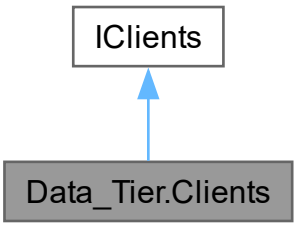
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Object Tier/Client.cs

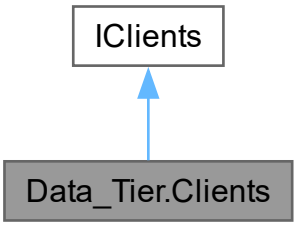
6.2 Data_Tier.Clients Class Reference

Singleton class that manages the list of clients. Allows adding, removing, updating and retrieving clients.

Inheritance diagram for Data_Tier.Clients:



Collaboration diagram for Data_Tier.Clients:



Public Member Functions

- int AddClient (Client client)

  Adds a new client to the list of clients and sorts the list.
- bool RemoveClient (int idClient)

  Removes a client from the list based on the client ID.
- bool ExistClient (Client client)

  Checks if a specific client exists in the list by comparing with another client object.
- bool ExistClient (int idClient)

  Checks if a client exists in the list based on the client ID.
- bool UpdateContact (int idClient, int contacto)

  Updates the contact information of a client based on the client ID.
- Client GetClient (int idClient)

  Retrieves a client from the list based on the client ID.

Public Member Functions inherited from Interface_Tier.IClients

Protected Member Functions

- Clients ()

  Initializes a new instance of the Clients class, with an empty list of clients.

Properties

- static Clients Instance [get]

  Gets the singleton instance of the Clients class.

6.2.1 Detailed Description

Singleton class that manages the list of clients. Allows adding, removing, updating and retrieving clients.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Clients()

```
Data_Tier.Clients.Clients ()  [protected]
```

Initializes a new instance of the Clients class, with an empty list of clients.

```
00071         {
00072             clients = new List<Client>(5);
00073         }
```

6.2.3 Member Function Documentation

6.2.3.1 AddClient()

```
int Data_Tier.Clients.AddClient (
            Client client)
```

Adds a new client to the list of clients and sorts the list.

Parameters

client	he client to add to the list.

Returns

The ID of the added client.

Exceptions

ConfigurationErrorException	Throws an exception if the client is null or if any error occurs during the addition.

Implements Interface_Tier.IClients.

```
00105         {
00106             if (client == null)
00107             {
00108                 throw new ConfigurationErrorException("100");
00109             }
00110 
00111             try
00112             {
00113                 clients.Add(client);
00114                 clients.Sort();
00115                 return client.Id;
00116             }
00117             catch (Exception ex)
00118             {
00119                 throw new ConfigurationErrorException("101", ex);
00120             }
00121         }
```

6.2.3.2 ExistClient() [1/2]

```
bool Data_Tier.Clients.ExistClient (
            Client client)
```

Checks if a specific client exists in the list by comparing with another client object.

Parameters

client	The client to check for existence in the list.

Returns

Returns true if the client exists, otherwise false.

```
00158         {
00159             foreach (Client existingClient in clients)
00160             {
00161                 if (existingClient - client)
00162                 {
00163                     return true;
00164                 }
00165             }
00166 
00167             return false;
00168         }
```

6.2.3.3 ExistClient() [2/2]

```
bool Data_Tier.Clients.ExistClient (
            int idClient)
```

Checks if a client exists in the list based on the client ID.

Parameters

idClient	The ID of the client to check for existence.

Returns

Returns true if the client exists, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while checking for existence.

Implements Interface_Tier.IClients.

```
00179         {
00180             try
00181             {
00182                 foreach (Client client in clients)
00183                 {
00184                     if (client.Id == idClient)
00185                     {
00186                         return true;
00187                     }
00188                 }
00189 
00190                 return false;
00191             }
00192             catch (Exception ex)
00193             {
00194                 throw new ConfigurationErrorException("103", ex);
00195             }
00196         }
```

6.2.3.4 GetClient()

```
Client Data_Tier.Clients.GetClient (
            int idClient)
```

Retrieves a client from the list based on the client ID.

Parameters

idClient	The ID of the client to retrieve.

Returns

Returns the client object with the specified ID, or null if the client is not found.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while retrieving the client.

Implements Interface_Tier.IClients.

```
00236         {
00237             try
00238             {
00239                 return FindClient(idClient);
00240             }
00241             catch (Exception ex)
00242             {
00243                 throw new ConfigurationErrorException("115", ex);
00244             }
00245         }
```

6.2.3.5 RemoveClient()

```
bool Data_Tier.Clients.RemoveClient (
            int idClient)
```

Removes a client from the list based on the client ID.

Parameters

idClient	The ID of the client to remove.

Returns

Returns true if the client is successfully removed, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal of the client.

Implements Interface_Tier.IClients.

```
00132         {
00133             try
00134             {
00135                 Client client = FindClient(idClient);
00136 
00137                 if (client != null)
00138                 {
00139                     clients.Remove(client);
00140                     clients.Sort();
00141                     return true;
00142                 }
00143             }
00144             catch (Exception ex)
00145             {
00146                 throw new ConfigurationErrorException("102", ex);
00147             }
00148 
00149             return false;
00150         }
```

6.2.3.6 UpdateContact()

```
bool Data_Tier.Clients.UpdateContact (
            int idClient,
            int contacto)
```

Updates the contact information of a client based on the client ID.

Parameters

idClient	The ID of the client to update.
contacto	The new contact information to set for the client.

Returns

Returns true if the contact information was updated successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the update process.

Implements Interface_Tier.IClients.

```
00208         {
00209             try
00210             {
00211                 Client client = FindClient(idClient);
00212 
00213                 if (client != null)
00214                 {
00215                     client.ContactInfo = contacto;
00216                     return true;
00217                 }
00218             }
00219             catch (Exception ex)
00220             {
00221                 throw new ConfigurationErrorException("105", ex);
00222             }
00223 
00224             return false;
00225         }
```

6.2.4 Property Documentation

6.2.4.1 Instance

```
Clients Data_Tier.Clients.Instance  [static], [get]
```

Gets the singleton instance of the Clients class.

```
00044         {
00045             get
00046             {
00047                 if (instance == null)
00048                 {
00049                     instance = new Clients();
00050                 }
00051 
00052                 return instance;
00053             }
00054         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/Clients.cs

6.3 Unit_Test.ClientsTest Class Reference

Public Member Functions

- void TestRegisterClient_ValidClient ()
- void TestGetClientById ()

6.3.1 Member Function Documentation

6.3.1.1 TestGetClientById()

```
void Unit_Test.ClientsTest.TestGetClientById ()
00025         {
00026             // Arrange
00027             Client client = new Client("Teste", 827234234);
00028             Company.RegisterClient(client);
00029
00030             // Act
00031             Client resultado = Company.GetClientById(client.Id);
00032
00033             //Assert
00034             Assert.AreEqual(client, resultado);
00035
00036         }
```

6.3.1.2 TestRegisterClient_ValidClient()

```
void Unit_Test.ClientsTest.TestRegisterClient_ValidClient ()
00012         {
00013             // Arrange
00014             Client c1 = Client.CreateClient("Test", 33232432);
00015
00016             // Act
00017             Company.RegisterClient(c1);
00018
00019             // Assert
00020             Assert.IsTrue(Clients.Instance.ExistClient(c1));
00021         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Unit Test/ClientsTest.cs

6.4 Business_Tier.Company Class Reference

The Enterprise class manages operations related to customers employees, materials and projects.It handles tasks such as registration, update and removal, manages project states and resource usage.Ensures proper validation and exception handling for each action.

Static Public Member Functions

- static bool SaveAllData (string path)

  Saves all application data to the specified file path.
- static bool LoadAllData (string path)

  Loads all application data from the specified file path.
- static int RegisterClient (Client client)

  Registers a new client in the system.
- static bool DeleteClient (int idClient)

  Deletes an existing client based on their unique ID.
- static bool IsClientRegistered (int idClient)

  Checks whether a client is registered in the system using their unique ID.
- static bool UpdateClientContact (int idClient, int contact)

  Updates the contact information of an existing client.

- static Client GetClientById (int idClient)

  Retrieves the details of a specific client using their unique ID.
- static int RegistEmployee (Employee employee)

  Registers a new employee in the system.
- static bool DeleteEmployee (int idEmployee)

  Deletes an existing employee based on their unique ID.
- static bool IsEmployeeRegistered (int idEmployee)

  Checks whether an employee is registered in the system using their unique ID.
- static bool UpdateEmployeeRole (int idEmployee, string role, double priceHourly)

  Updates the role and hourly price of an existing employee.
- static Employee GetEmployeeById (int idEmployee)

  Retrieves the details of a specific employee using their unique ID.
- static int RegisterMaterial (Material material, int quantity)

  Registers a material by adding it to the catalog and inventory.
- static bool IsMaterialRegistered (int idMaterial)

  Checks if a material is registered in the system.
- static bool UpdateStock (int idMaterial, int quantity)

  Updates the stock quantity of a material in the inventory.
- static bool UpdatePrice (int idMaterial, double price)

  Updates the price of a material in the catalog.
- static Material GetMaterial (int idMaterial)

  Retrieves the details of a material from the catalog.
- static MaterialQuantity GetQuantityOfMaterial (int idMaterial)

  Retrieves the quantity details of a material in the inventory.
- static int RegistProject (Project project)

  Registers a new project.
- static bool IsProjectRegistered (int idProject)

  Checks if a project is registered in the system.
- static bool UpdateStatusProject (int idProject, Status status)

  Updates the status of a project.
- static bool CloseProject (int idProject)

  Closes a project.
- static bool AddClientToProject (int idProject, int idClient)

  Adds a client to a project.
- static bool DeleteClientToProject (int idProject, int idClient)

  Removes a client from a project.
- static bool AddEmployeeToProject (int idProject, int idEmployee)

  Adds an employee to a project.
- static bool DeleteEmplyeeToProject (int idProject, int idEmployee)

  Removes an employee from a project.
- static bool UseMaterial (int idProject, int idMaterial, int quantity)

  Uses a specified quantity of a material for a project.

6.4.1 Detailed Description

The Enterprise class manages operations related to customers employees, materials and projects.It handles tasks such as registration, update and removal, manages project states and resource usage.Ensures proper validation and exception handling for each action.

6.4.2 Member Function Documentation

6.4.2.1 AddClientToProject()

```
static bool Business_Tier.Company.AddClientToProject (
            int idProject,
            int idClient)  [static]
```

Adds a client to a project.

Parameters

idProject	The unique ID of the project.
idClient	The unique ID of the client to add.

Returns

True if the client was successfully added to the project.

Exceptions

ConfigurationErrorException	Throws "820" if the project does not exist. Throws "821" if the client does not exist. Throws "822" for any unexpected error during the addition process.

```
00698         {
00699             if (!Projects.Instance.ProjectExists(idProject))
00700             {
00701                 throw new ConfigurationErrorException("820");
00702             }
00703 
00704             if (!Clients.Instance.ExistClient(idClient))
00705             {
00706                 throw new ConfigurationErrorException("821");
00707             }
00708 
00709             try
00710             {
00711                 return Projects.Instance.AddClient(idProject, idClient);
00712 
00713             }
00714             catch (Exception ex)
00715             {
00716                 throw new ConfigurationErrorException("822." + ex);
00717             }
00718 
00719         }
```

6.4.2.2 AddEmployeeToProject()

```
static bool Business_Tier.Company.AddEmployeeToProject (
            int idProject,
            int idEmployee)  [static]
```

Adds an employee to a project.

Parameters

idProject	The unique ID of the project.
idEmployee	The unique ID of the employee to add.

Returns

True if the employee was successfully added to the project.

Exceptions

ConfigurationErrorException	Throws "820" if the project does not exist. Throws "824" if the employee does not exist. Throws "825" for any unexpected error during the addition process.

```
00770         {
00771             if (!Projects.Instance.ProjectExists(idProject))
00772             {
00773                 throw new ConfigurationErrorException("820");
00774             }
00775 
00776             if (!Employees.Instance.EmployeeExist(idEmployee))
00777             {
00778                 throw new ConfigurationErrorException("824");
00779             }
00780 
00781             try
00782             {
00783                 bool result = Projects.Instance.AddEmployee(idProject, idEmployee);
00784                 return result;
00785 
00786             }
00787             catch (Exception ex)
00788             {
00789                 throw new ConfigurationErrorException("825" + ex);
00790             }
00791 
00792         }
```

6.4.2.3 CloseProject()

```
static bool Business_Tier.Company.CloseProject (
             int idProject)  [static]
```

Closes a project.

Parameters

idProject	The unique ID of the project to close.

Returns

True if the project was successfully closed.

Exceptions

ConfigurationErrorException	Throws "819" for any unexpected error during the closing process.

```
00673         {
00674             try
00675             {
00676                 return Projects.Instance.CloseProject(idProject);
00677             }
00678             catch (Exception ex)
00679             {
00680                 throw new ConfigurationErrorException("819" + ex);
00681             }
00682         }
```

6.4.2.4 DeleteClient()

```
static bool Business_Tier.Company.DeleteClient (
             int idClient)  [static]
```

Deletes an existing client based on their unique ID.

Parameters

idClient	The ID of the client to be removed.

Returns

True if the client was successfully deleted.

Exceptions

ConfigurationErrorException	Throws "112" if the client does not exist. Throws "109" for any unexpected error during deletion.

```
00127         {
00128             if (!Clients.Instance.ExistClient(idClient))
00129             {
00130                 throw new ConfigurationErrorException("112");
00131             }
00132 
00133             try
00134             {
00135                 return Clients.Instance.RemoveClient(idClient);
00136             }
00137             catch (Exception ex)
00138             {
00139                 throw new ConfigurationErrorException("109" + ex);
00140             }
00141         }
```

6.4.2.5 DeleteClientToProject()

```
static bool Business_Tier.Company.DeleteClientToProject (
             int idProject,
             int idClient)  [static]
```

Removes a client from a project.

Parameters

idProject	The unique ID of the project.
idClient	The unique ID of the client to remove.

Returns

True if the client was successfully removed from the project.

Exceptions

ConfigurationErrorException	Throws "820" if the project does not exist. Throws "821" if the client does not exist. Throws "823" for any unexpected error during the removal process.

```
00733         {
00734             if (!Projects.Instance.ProjectExists(idProject))
00735             {
00736                 throw new ConfigurationErrorException("820");
00737             }
00738 
00739             if (!Clients.Instance.ExistClient(idClient))
00740             {
00741                 throw new ConfigurationErrorException("821");
```

```
00742             }
00743
00744             try
00745             {
00746                 bool r = Projects.Instance.RemoveClient(idProject, idClient);
00747                 return r;
00748             }
00749             catch (Exception ex)
00750             {
00751                 throw new ConfigurationErrorException("823" + ex);
00752             }
00753         }
```

6.4.2.6 DeleteEmployee()

```
static bool Business_Tier.Company.DeleteEmployee (
             int idEmployee)  [static]
```

Deletes an existing employee based on their unique ID.

Parameters

idEmployee	The ID of the employee to be removed.

Returns

True if the employee was successfully deleted.

Exceptions

ConfigurationErrorException	Throws "412" if the employee does not exist. Throws "409" for any unexpected error during deletion.

```
00265         {
00266
00267             if (!Employees.Instance.EmployeeExist(idEmployee))
00268             {
00269                 throw new ConfigurationErrorException("412");
00270             }
00271             try
00272             {
00273                 return Employees.Instance.RemoveEmployee(idEmployee);
00274             }
00275             catch (Exception ex)
00276             {
00277                 throw new ConfigurationErrorException("409" + ex);
00278             }
00279         }
```

6.4.2.7 DeleteEmplyeeToProject()

```
static bool Business_Tier.Company.DeleteEmplyeeToProject (
             int idProject,
             int idEmployee)  [static]
```

Removes an employee from a project.

Parameters

idProject	The unique ID of the project.
idEmployee	The unique ID of the employee to remove.

Returns

True if the employee was successfully removed from the project.

Exceptions

ConfigurationErrorException	Throws "820" if the project does not exist. Throws "824" if the employee does not exist. Throws "826" for any unexpected error during the removal process.

```
00806         {
00807             if (!Projects.Instance.ProjectExists(idProject))
00808             {
00809                 throw new ConfigurationErrorException("820");
00810             }
00811 
00812             if (!Employees.Instance.EmployeeExist(idEmployee))
00813             {
00814                 throw new ConfigurationErrorException("824");
00815             }
00816 
00817             try
00818             {
00819                 bool r = Projects.Instance.RemoveEmployee(idProject, idEmployee);
00820                 return r;
00821             }
00822             catch (Exception ex)
00823             {
00824                 throw new ConfigurationErrorException("826" + ex);
00825             }
00826         }
```

6.4.2.8 GetClientById()

```
static Client Business_Tier.Company.GetClientById (
            int idClient)  [static]
```

Retrieves the details of a specific client using their unique ID.

Parameters

idClient	The ID of the client to retrieve.

Returns

A Client object containing the client's details.

Exceptions

ConfigurationErrorException	Throws "112" if the client does not exist. Throws "114" for any unexpected error during retrieval.

```
00206         {
00207             if (!Clients.Instance.ExistClient(idClient))
00208             {
00209                 throw new ConfigurationErrorException("112");
00210             }
00211             try
00212             {
00213                 return Clients.Instance.GetClient(idClient);
00214             }
00215             catch (Exception ex)
00216             {
00217                 throw new ConfigurationErrorException("114" + ex);
00218             }
00219         }
```

6.4.2.9 GetEmployeeById()

```
static Employee Business_Tier.Company.GetEmployeeById (
            int idEmployee)  [static]
```

Retrieves the details of a specific employee using their unique ID.

Parameters

idEmployee	The ID of the employee to retrieve.

Returns

An Employee object containing the employee's details.

Exceptions

ConfigurationErrorException	Throws "412" if the employee does not exist. Throws "415" for any unexpected error during retrieval.

```
00353         {
00354             if (!Employees.Instance.EmployeeExist(idEmployee))
00355             {
00356                 throw new ConfigurationErrorException("412");
00357             }
00358 
00359             try
00360             {
00361                 return Employees.Instance.GetEmployee(idEmployee);
00362             }
00363             catch (Exception ex)
00364             {
00365                 throw new ConfigurationErrorException("415", ex);
00366             }
00367         }
```

6.4.2.10 GetMaterial()

```
static Material Business_Tier.Company.GetMaterial (
            int idMaterial)  [static]
```

Retrieves the details of a material from the catalog.

Parameters

idMaterial	The unique ID of the material to retrieve.

Returns

A Material object containing the material's details.

Exceptions

ConfigurationErrorException	Throws "616" for any unexpected error during the retrieval process.

```
00553         {
00554             try
00555             {
00556                 return Materials.Instance.GetMaterial(idMaterial);
00557             }
00558             catch (Exception ex)
00559             {
00560                 throw new ConfigurationErrorException("616" + ex);
00561             }
00562         }
```

6.4.2.11 GetQuantityOfMaterial()

```
static MaterialQuantity Business_Tier.Company.GetQuantityOfMaterial (
            int idMaterial)  [static]
```

Retrieves the quantity details of a material in the inventory.

Parameters

idMaterial	The unique ID of the material to retrieve.

Returns

A MaterialQuantity object containing the material's inventory details.

Exceptions

ConfigurationErrorException	Throws "616" for any unexpected error during the retrieval process.

```
00573         {
00574             try
00575             {
00576                 return MaterialInventory.Instance.GetMaterialQuantity(idMaterial);
00577             }
00578             catch (Exception ex)
00579             {
00580                 throw new ConfigurationErrorException("616" + ex);
00581             }
00582         }
```

6.4.2.12 IsClientRegistered()

```
static bool Business_Tier.Company.IsClientRegistered (
             int idClient)  [static]
```

Checks whether a client is registered in the system using their unique ID.

Parameters

idClient	The ID of the client to check.

Returns

True if the client is registered, false otherwise.

Exceptions

ConfigurationErrorException	Throws "109" for any unexpected error during the operation.

```
00152         {
00153             try
00154             {
00155                 return Clients.Instance.ExistClient(idClient);
00156             }
00157             catch (Exception ex)
00158             {
00159                 throw new ConfigurationErrorException("110" + ex);
00160             }
00161         }
```

6.4.2.13 IsEmployeeRegistered()

```
static bool Business_Tier.Company.IsEmployeeRegistered (
             int idEmployee)  [static]
```

Checks whether an employee is registered in the system using their unique ID.

Parameters

idEmployee	The ID of the employee to check.

Returns

True if the employee is registered, false otherwise.

Exceptions

ConfigurationErrorException	Throws "410" for any unexpected error during the operation.

```
00290         {
00291             try
00292             {
00293                 return Employees.Instance.EmployeeExist(idEmployee);
00294             }
00295             catch (Exception ex)
00296             {
00297                 throw new ConfigurationErrorException("410" + ex);
00298             }
00299
00300         }
```

6.4.2.14 IsMaterialRegistered()

```
static bool Business_Tier.Company.IsMaterialRegistered (
             int idMaterial)  [static]
```

Checks if a material is registered in the system.

Parameters

idMaterial	The unique ID of the material to check.

Returns

True if the material is registered, false otherwise.

Exceptions

ConfigurationErrorException	Throws "612" for any unexpected error during the verification process.

```
00456         {
00457             try
00458             {
00459                 if (Materials.Instance.MaterialExist(idMaterial))
00460                 {
00461                     bool r = MaterialInventory.Instance.VerifyMaterialExistence(idMaterial);
00462                     return r;
00463                 }
00464             }
00465             catch (Exception ex)
00466             {
00467                 throw new ConfigurationErrorException("612" + ex);
00468             }
00469
00470             return false;
00471         }
```

6.4.2.15 IsProjectRegistered()

```
static bool Business_Tier.Company.IsProjectRegistered (
             int idProject)  [static]
```

Checks if a project is registered in the system.

Parameters

idProject	The unique ID of the project.

Returns

True if the project is registered, false otherwise.

Exceptions

ConfigurationErrorException	Throws "818" for any unexpected error during the check process.

```
00631         {
00632             try
00633             {
00634                 return Projects.Instance.ProjectExists(idProject);
00635             }
00636             catch (Exception ex)
00637             {
00638                 throw new ConfigurationErrorException("818" + ex);
00639             }
00640         }
```

6.4.2.16 LoadAllData()

```
static bool Business_Tier.Company.LoadAllData (
            string path)  [static]
```

Loads all application data from the specified file path.

Parameters

path	The file path from which the data will be loaded.

Returns

Returns `true` if the data is loaded successfully.

Exceptions

ConfigurationErrorException	Thrown if the file does not exist, data deserialization fails, or data cannot be updated in the system.

```
00060         {
00061             try
00062             {
00063                 if (!File.Exists(path))
00064                 {
00065                     throw new ConfigurationErrorException("703");
00066                 }
00067 
00068                 return Data.LoadData(path);
00069             }
00070             catch (Exception ex)
00071             {
00072                 throw new ConfigurationErrorException("701 " + ex);
00073             }
00074         }
```

6.4.2.17 RegistEmployee()

```
static int Business_Tier.Company.RegistEmployee (
            Employee employee)  [static]
```

Registers a new employee in the system.

Parameters

employee	The employee object to be registered.

Returns

The unique ID of the newly registered employee.

Exceptions

ConfigurationErrorException	Throws "405" if the employee object is null. Throws "407" if the employee already exists in the system. Throws "408" for any unexpected error during registration.

```
00234         {
00235             if (employee == null)
00236             {
00237                 throw new ConfigurationErrorException("405");
00238             }
00239 
00240             if (Employees.Instance.EmployeeExist(employee))
00241             {
00242                 throw new ConfigurationErrorException("407");
00243             }
00244 
00245             try
00246             {
00247                 return Employees.Instance.AddEmployee(employee);
00248             }
00249             catch (Exception ex)
00250             {
00251                 throw new ConfigurationErrorException("408" + ex);
00252             }
00253         }
```

6.4.2.18 RegisterClient()

```
static int Business_Tier.Company.RegisterClient (
            Client client)  [static]
```

Registers a new client in the system.

Parameters

client	The client object to be registered.

Returns

The unique ID of the newly registered client.

Exceptions

ConfigurationErrorException	Throws "106" if the client object is null. Throws "107" if the client already exists in the system. Throws "108" for any unexpected error during registration.

```
00090         {
00091             if (client.Name == string.Empty)
00092             {
00093                 throw new ConfigurationErrorException("116");
00094             }
00095 
00096             if (client == null)
00097             {
00098                 throw new ConfigurationErrorException("106");
00099             }
00100 
00101             if (Clients.Instance.ExistClient(client))
00102             {
00103                 throw new ConfigurationErrorException("107");
00104             }
00105 
00106             try
00107             {
00108                 return Clients.Instance.AddClient(client);
00109             }
00110 
00111             catch (Exception ex)
00112             {
00113                 throw new ConfigurationErrorException("108" + ex);
00114             }
00115         }
```

6.4.2.19 RegisterMaterial()

```
static int Business_Tier.Company.RegisterMaterial (
            Material material,
            int quantity)  [static]
```

Registers a material by adding it to the catalog and inventory.

Parameters

material	The material to be registered.
quantity	The initial quantity of the material.

Returns

The unique ID of the material in the inventory.

```
00379         {
00380             int idM = AddMaterialToCatalog(material);
00381             int idMI = AddMaterialToInventory(idM, quantity);
00382             return idMI;
00383         }
```

6.4.2.20 RegistProject()

```
static int Business_Tier.Company.RegistProject (
            Project project)  [static]
```

Registers a new project.

Parameters

project	The project to be registered.

Returns

The unique ID of the registered project.

Exceptions

ConfigurationErrorException	Throws "815" if the project is null. Throws "820" if the project already exists. Throws "817" for any unexpected error during the registration process.

```
00599         {
00600             if (project == null)
00601             {
00602                 throw new ConfigurationErrorException("815");
00603             }
00604 
00605             if (Projects.Instance.ProjectExists(project))
00606             {
00607                 throw new ConfigurationErrorException("816");
00608             }
00609 
00610             try
00611             {
00612                 System.Threading.Thread.Sleep(500);
00613                 int idProject = Projects.Instance.AddProject(ProjectData.CreateProjectData(project));
00614                 return idProject;
00615             }
00616             catch (Exception ex)
00617             {
00618                 throw new ConfigurationErrorException("817" + ex);
00619             }
00620         }
```

6.4.2.21 SaveAllData()

```
static bool Business_Tier.Company.SaveAllData (
             string path)  [static]
```

Saves all application data to the specified file path.

Parameters

path	The file path where the data will be saved.

Returns

Returns `true` if the data is saved successfully.

Exceptions

ConfigurationErrorException	Thrown if the data cannot be collected or the file operation fails.

```
00035         {
00036             try
00037             {
00038                 if (!File.Exists(path))
00039                 {
00040                     throw new FileNotFoundException("703");
00041                 }
00042 
00043                 return Data.SaveData(path);
00044             }
00045             catch (Exception ex)
00046             {
00047                 throw new ConfigurationErrorException("700" + ex);
00048             }
00049         }
```

6.4.2.22 UpdateClientContact()

```
static bool Business_Tier.Company.UpdateClientContact (
             int idClient,
             int contact)  [static]
```

Updates the contact information of an existing client.

Parameters

idClient	The ID of the client whose contact information needs updating.
contact	The new contact number.

Returns

True if the update was successful.

Exceptions

ConfigurationErrorException	Throws "111" if the contact number is invalid Throws "112" if the client does not exist. Throws "113" for any unexpected error during the update.

```
00175         {
00176             if (contact < 9)
00177             {
00178                 throw new ConfigurationErrorException("111");
00179             }
00180 
00181             if (!Clients.Instance.ExistClient(idClient))
00182             {
00183                 throw new ConfigurationErrorException("112");
00184             }
00185 
00186             try
00187             {
00188                 return Clients.Instance.UpdateContact(idClient, contact);
00189             }
00190             catch (Exception ex)
00191             {
00192                 throw new ConfigurationErrorException("113" + ex);
00193             }
00194         }
```

6.4.2.23 UpdateEmployeeRole()

```
static bool Business_Tier.Company.UpdateEmployeeRole (
             int idEmployee,
             string role,
             double priceHourly)  [static]
```

Updates the role and hourly price of an existing employee.

Parameters

idEmployee	The ID of the employee whose role and price need updating.
role	The new role of the employee.
priceHourly	The new hourly price for the employee.

Returns

True if the update was successful.

Exceptions

ConfigurationErrorException	Throws "412" if the employee does not exist. Throws "411" if the role is empty. Throws "413" if the hourly price is less than or equal to zero. Throws "414" for any unexpected error during the update.

```
00316         {
00317             if (!Employees.Instance.EmployeeExist(idEmployee))
00318             {
00319                 throw new ConfigurationErrorException("412");
00320             }
00321
00322             if (role == string.Empty)
00323             {
00324                 throw new ConfigurationErrorException("411");
00325             }
00326
00327             if (priceHourly <= 0)
00328             {
00329                 throw new ConfigurationErrorException("413");
00330             }
00331
00332             try
00333             {
00334                 return Employees.Instance.UpdateRole(idEmployee, role, priceHourly);
00335             }
00336
00337             catch (Exception ex)
00338             {
00339                 throw new ConfigurationErrorException("414" + ex);
00340             }
00341         }
```

6.4.2.24 UpdatePrice()

```
static bool Business_Tier.Company.UpdatePrice (
             int idMaterial,
             double price)  [static]
```

Updates the price of a material in the catalog.

Parameters

idMaterial	The unique ID of the material to update.
price	The new price to be set.

Returns

True if the price update was successful.

Exceptions

ConfigurationErrorException	Throws "615" if the price is less than zero. Throws "610" if the material does not exist in the catalog. Throws "616" for any unexpected error during the update process.

```
00520         {
00521             if (price < 0)
00522             {
00523                 throw new ConfigurationErrorException("615");
00524             }
00525
00526             if (!Materials.Instance.MaterialExist(idMaterial))
00527             {
00528                 throw new ConfigurationErrorException("610");
00529             }
```

```
00530
00531             try
00532             {
00533                 bool update = Materials.Instance.UpdatePrice(idMaterial, price);
00534                 return update;
00535             }
00536
00537             catch (Exception ex)
00538             {
00539                 throw new ConfigurationErrorException("614." + ex);
00540             }
00541
00542         }
```

6.4.2.25 UpdateStatusProject()

```
static bool Business_Tier.Company.UpdateStatusProject (
             int idProject,
             Status status)  [static]
```

Updates the status of a project.

Parameters

idProject	The unique ID of the project.
status	The new status of the project.

Returns

True if the status update was successful.

Exceptions

ConfigurationErrorException	Throws "819" for any unexpected error during the update process.

```
00652         {
00653             try
00654             {
00655                 return Projects.Instance.UpdateStatus(idProject, status);
00656             }
00657             catch (Exception ex)
00658             {
00659                 throw new ConfigurationErrorException("819" + ex);
00660             }
00661         }
```

6.4.2.26 UpdateStock()

```
static bool Business_Tier.Company.UpdateStock (
             int idMaterial,
             int quantity)  [static]
```

Updates the stock quantity of a material in the inventory.

Parameters

idMaterial	The unique ID of the material to update.
quantity	The new quantity to be set.

Returns

True if the stock update was successful.

Exceptions

ConfigurationErrorException	Throws "613" if the quantity is less than zero. Throws "610" if the material does not exist in the inventory. Throws "614" for any unexpected error during the update process.

```
00485         {
00486             if (quantity < 0)
00487             {
00488                 throw new ConfigurationErrorException("613");
00489             }
00490 
00491             if (!MaterialInventory.Instance.VerifyMaterialExistence(idMaterial))
00492             {
00493                 throw new ConfigurationErrorException("610");
00494             }
00495 
00496             try
00497             {
00498                 bool update = MaterialInventory.Instance.UpdateQuantity(idMaterial, quantity);
00499                 return update;
00500             }
00501             catch (Exception ex)
00502             {
00503                 throw new ConfigurationErrorException("618" + ex);
00504             }
00505 
00506         }
```

6.4.2.27 UseMaterial()

```
static bool Business_Tier.Company.UseMaterial (
             int idProject,
             int idMaterial,
             int quantity)  [static]
```

Uses a specified quantity of a material for a project.

Parameters

idProject	The unique ID of the project.
idMaterial	The unique ID of the material to use.
quantity	The quantity of the material to use.

Returns

True if the material was successfully used.

Exceptions

ConfigurationErrorException	Throws "820" if the project does not exist. Throws "827" if the material does not exist. Throws "828" if the quantity is insufficient. Throws "829" for any unexpected error during the process.

```
00845         {
00846             if (!Projects.Instance.ProjectExists(idProject))
00847             {
00848                 throw new ConfigurationErrorException("820");
00849             }
00850 
00851             if (!MaterialInventory.Instance.VerifyMaterialExistence(idMaterial))
00852             {
00853                 throw new ConfigurationErrorException("827");
00854             }
```

```
00855
00856             if (!MaterialInventory.Instance.VerifyMaterialQuantity(idMaterial, quantity))
00857             {
00858                 throw new ConfigurationErrorException("828");
00859             }
00860
00861             try
00862             {
00863                 bool result = MaterialInventory.Instance.UseMaterial(idMaterial, quantity);
00864                 bool result1 = Projects.Instance.UseMaterial(idProject, idMaterial, quantity);
00865                 return result1;
00866
00867             }
00868             catch (Exception ex)
00869             {
00870                 throw new ConfigurationErrorException("829" + ex);
00871             }
00872
00873         }
```

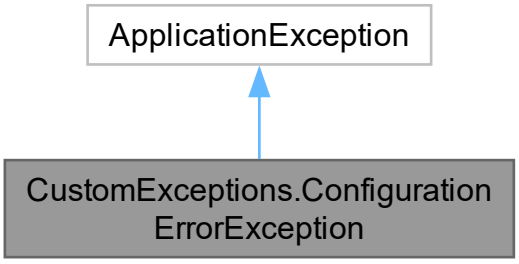
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Business Tier/Company.cs

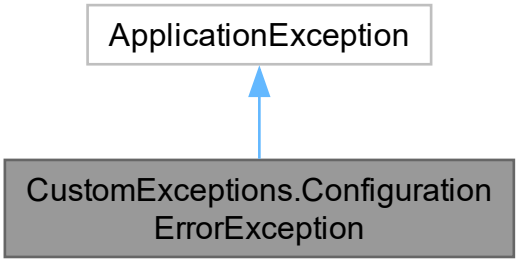
6.5 CustomExceptions.ConfigurationErrorException Class Reference

Custom exception used for handling configuration-related errors in the application.

Inheritance diagram for CustomExceptions.ConfigurationErrorException:



Collaboration diagram for CustomExceptions.ConfigurationErrorException:



Public Member Functions

- ConfigurationErrorException ()

  Initializes a new instance of the ConfigurationErrorException class with a default error message.
- ConfigurationErrorException (string error)

  Initializes a new instance of theConfigurationErrorException class with a specified error message.
- ConfigurationErrorException (string error, Exception exception)

  Initializes a new instance of the ConfigurationErrorException class with a specified error message and a reference to the inner exception that is the cause of this exception.

6.5.1 Detailed Description

Custom exception used for handling configuration-related errors in the application.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ConfigurationErrorException() [1/3]

```
CustomExceptions.ConfigurationErrorException.ConfigurationErrorException ()
```

Initializes a new instance of the ConfigurationErrorException class with a default error message.

```
00021                                           : base("An error occurred in the application.")
00022         {
00023
00024         }
```

6.5.2.2 ConfigurationErrorException() [2/3]

```
CustomExceptions.ConfigurationErrorException.ConfigurationErrorException (
            string error)
```

Initializes a new instance of theConfigurationErrorException class with a specified error message.

Parameters

error	The error message that explains the reason for the exception.

```
00030                                                       : base(error)
00031         {
00032
00033         }
```

6.5.2.3 ConfigurationErrorException() [3/3]

```
CustomExceptions.ConfigurationErrorException.ConfigurationErrorException (
            string error,
            Exception exception)
```

Initializes a new instance of the ConfigurationErrorException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

error	The error message that explains the reason for the exception.
exception	The exception that is the cause of the current exception, or `null` if no inner exception is specified.

```
00040                                                                   : base(error ,
      exception)
00041         {
00042
00043         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ CustomExceptions/Execeptions.cs

6.6 Data_Tier.Data Class Reference

The `Data` class serves as a centralized data manager for the application. It collects data from various parts of the system, organizes it into lists, and provides methods for saving to and loading from a binary file.

Public Member Functions

- bool CollectData ()

  Collects data from various system modules and stores it in the class.
- bool PutData ()

  Updates the system modules with the data stored in the class's lists.

Static Public Member Functions

- static bool SaveData (string path)

  Saves the current data to a binary file at the specified path.
- static bool LoadData (string path)

  Loads data from a binary file and updates the system with the loaded data.

6.6.1 Detailed Description

The `Data` class serves as a centralized data manager for the application. It collects data from various parts of the system, organizes it into lists, and provides methods for saving to and loading from a binary file.

6.6.2 Member Function Documentation

6.6.2.1 CollectData()

```
bool Data_Tier.Data.CollectData ()
```

Collects data from various system modules and stores it in the class.

Returns

Returns `true` if the data is collected successfully.

Exceptions

ConfigurationErrorException	Thrown if data collection fails.

```
00058         {
00059             try
00060             {
00061                 clients = Clients.Instance.ClientsD;
00062                 employees = Employees.Instance.EmployeesD;
00063                 materials = Materials.Instance.MaterialD;
00064                 inventory = MaterialInventory.Instance.InventoryD;
00065                 projects = Projects.Instance.ProjectsD;
00066                 return true;
00067             }
00068             catch (Exception)
00069             {
00070                 throw new ConfigurationErrorException("700");
00071             }
00072         }
```

6.6.2.2 LoadData()

```
static bool Data_Tier.Data.LoadData (
            string path)  [static]
```

Loads data from a binary file and updates the system with the loaded data.

Parameters

path	The file path from where the data will be loaded.

Returns

Returns `true` if the data is successfully loaded and updated.

Exceptions

ConfigurationErrorException	Thrown if an error occurs during data loading.

```
00150         {
00151             try
00152             {
00153                 Data data = new Data();
00154                 Stream s = File.Open(path, FileMode.Open, FileAccess.Read);
00155                 BinaryFormatter b = new BinaryFormatter();
00156                 data = (Data)b.Deserialize(s);
00157                 s.Close();
00158 
00159                 if (!data.PutData())
00160                 {
00161                     throw new Exception("700");
00162                 }
00163 
00164                 return true;
00165             }
00166             catch (Exception ex)
00167             {
00168                 throw new ConfigurationErrorException("706" + ex.Message);
00169             }
00170 
00171         }
```

6.6.2.3 PutData()

```
bool Data_Tier.Data.PutData ()
```

Updates the system modules with the data stored in the class's lists.

Returns

Returns `true` if the data is successfully updated.

Exceptions

Exception	Thrown if updating data fails.

```
00080         {
00081             try
00082             {
00083                 Clients.Instance.ClientsD = clients;
00084                 Employees.Instance.EmployeesD = employees;
00085                 MaterialInventory.Instance.InventoryD = inventory;
00086                 Materials.Instance.MaterialD = materials;
00087                 Projects.Instance.ProjectsD = projects;
00088                 SetCounterEqual();
00089 
00090                 return true;
00091             }
00092             catch (Exception)
00093             {
00094                 throw new Exception("701");
00095             }
00096 
00097         }
```

6.6.2.4 SaveData()

```
static bool Data_Tier.Data.SaveData (
             string path)  [static]
```

Saves the current data to a binary file at the specified path.

Parameters

path	The file path where the data will be saved.

Returns

Returns `true` if the data is successfully saved.

Exceptions

ConfigurationErrorException	Thrown if an error occurs during data saving.

```
00120         {
00121             try
00122             {
00123                 Data data = new Data();
00124 
00125                 if (!data.CollectData())
00126                 {
00127                     throw new ConfigurationErrorException("702");
00128                 }
00129 
00130                 Stream fs = new FileStream(path, FileMode.Create);
00131                 BinaryFormatter binaryFormatter = new BinaryFormatter();
00132                 binaryFormatter.Serialize(fs, data);
00133                 fs.Close();
00134                 return true;
00135             }
00136             catch (Exception ex)
00137             {
00138                 throw new ConfigurationErrorException("705" + ex.Message);
00139             }
00140 
00141         }
```

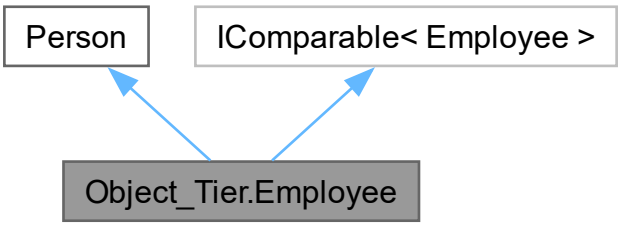
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/Data.cs

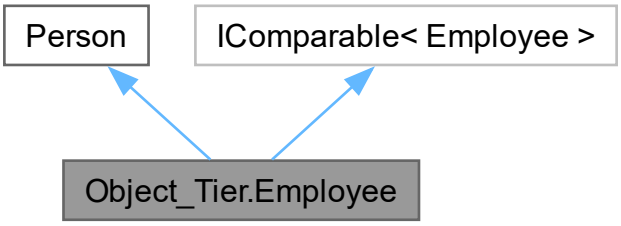
6.7 Object_Tier.Employee Class Reference

Represents an employee in the system, inheriting from the Person class.

Inheritance diagram for Object_Tier.Employee:



Collaboration diagram for Object_Tier.Employee:



Public Member Functions

- Employee (string name, string role, double hourlyrate)

  Initializes a new instance of the Employee class with the specified name, role, and hourly rate. Automatically assigns a unique ID.
- override bool Equals (object obj)

  Determines whether the specified object is equal to the current employee. Employees are equal if their name, role, and hourly rate match.
- override string ToString ()

  Returns a string representation of the employee, including ID, name, role, and hourly rate.
- int CompareTo (Employee employee)

  Compares the current employee to another employee based on their name.

Static Public Member Functions

- static Employee CreateEmployee (string name, string role, double hourlyRate)

  Creates a new Employee instance.
- static bool operator- (Employee employee1, Employee employee2)

  Checks if two employees are equal using the "-" operator.
- static bool operator+ (Employee employee1, Employee employee2)

  Checks if two employees are not equal using the "+" operator.
- static bool getNextEmployeeId ()

  Increments the static employee ID counter.

Properties

- string Role `[get, set]`

  Gets or sets the employee's role. Ensures the role is not empty.
- double HourlyRate `[get, set]`

  Gets or sets the employee's hourly rate.

Properties inherited from Object_Tier.Person

- int Id `[get]`

  Gets the ID of the person.
- string Name `[get, set]`

  Gets or sets the name of the person.

Additional Inherited Members

Protected Member Functions inherited from Object_Tier.Person

- Person (int id, string name)

  Initializes a new instance of the Person class with an ID and a name. The name is automatically formatted.

6.7.1 Detailed Description

Represents an employee in the system, inheriting from the Person class.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Employee()

```
Object_Tier.Employee.Employee (
            string name,
            string role,
            double hourlyrate)
```

Initializes a new instance of the Employee class with the specified name, role, and hourly rate. Automatically assigns a unique ID.

Parameters

name	The name of the employee.
role	The role of the employee.
hourlyrate	The hourly rate of the employee.

```
00085                                                                   : base(employeeIdCounter++, name)
      // Send for constructor Person
00086         {
00087             Role = role.ToUpper().Trim();
00088             HourlyRate = hourlyrate;
00089         }
```

6.7.3 Member Function Documentation

6.7.3.1 CompareTo()

```
int Object_Tier.Employee.CompareTo (
            Employee employee)
```

Compares the current employee to another employee based on their name.

Parameters

employee	The employee to compare to.

Returns

A value indicating the relative order of the employees.

```
00177         {
00178             return Name.CompareTo(employee.Name);
00179         }
```

6.7.3.2 CreateEmployee()

```
static Employee Object_Tier.Employee.CreateEmployee (
            string name,
            string role,
            double hourlyRate)  [static]
```

Creates a new Employee instance.

Parameters

name	The name of the employee.
role	The role of the employee.
hourlyRate	The hourly rate of the employee.

Returns

A new Employee instance.

```
00140         {
00141             return new Employee(name, role, hourlyRate);
00142         }
```

6.7.3.3 Equals()

```
override bool Object_Tier.Employee.Equals (
            object obj)
```

Determines whether the specified object is equal to the current employee. Employees are equal if their name, role, and hourly rate match.

Parameters

obj	The object to compare with the current employee.

Returns

True if the objects are equal; otherwise, false.

```
00101         {
00102             if (obj == null)
00103             {
00104                 return false;
00105             }
00106 
00107             if (obj is Employee)
00108             {
00109                 Employee otherEmp = obj as Employee;
00110 
00111                 if (Name == otherEmp.Name && Role == otherEmp.Role && HourlyRate ==
      otherEmp.HourlyRate)
00112                 {
00113                     return true;
00114                 }
00115             }
00116 
00117             return false;
00118         }
```

6.7.3.4 getNextEmployeeId()

```
static bool Object_Tier.Employee.getNextEmployeeId ()  [static]
```

Increments the static employee ID counter.

Returns

True after incrementing the counter.

```
00186         {
00187             employeeIdCounter++;
00188             return true;
00189         }
```

6.7.3.5 operator+()

```
static bool Object_Tier.Employee.operator+ (
             Employee employee1,
             Employee employee2)  [static]
```

Checks if two employees are not equal using the "+" operator.

Parameters

employee1	The first employee.
employee2	The second employee.

Returns

True if the employees are not equal; otherwise, false.

```
00167         {
00168             return !(employee1 - employee2);
00169         }
```

6.7.3.6 operator-()

```
static bool Object_Tier.Employee.operator- (
             Employee employee1,
             Employee employee2)  [static]
```

Checks if two employees are equal using the "-" operator.

Parameters

employee1	The first employee.
employee2	The second employee.

Returns

True if the employees are equal; otherwise, false.

```
00151         {
00152             if (employee1.Equals(employee2))
00153             {
00154                 return true;
00155             }
00156 
00157             return false;
00158         }
```

6.7.3.7 ToString()

```
override string Object_Tier.Employee.ToString ()
```

Returns a string representation of the employee, including ID, name, role, and hourly rate.

Returns

A string representation of the employee.

```
00125         {
00126             return Id + " " + Name + " " + Role + " " + HourlyRate;
00127         }
```

6.7.4 Property Documentation

6.7.4.1 HourlyRate

```
double Object_Tier.Employee.HourlyRate  [get], [set]
```

Gets or sets the employee's hourly rate.

```
00063         {
00064             set
00065             {
00066                 if (value > hourlyRate)
00067                 {
00068                     hourlyRate = value;
00069                 }
00070             }
00071             get { return hourlyRate; }
00072         }
```

6.7.4.2 Role

```
string Object_Tier.Employee.Role  [get], [set]
```

Gets or sets the employee's role. Ensures the role is not empty.

```
00048         {
00049             set
00050             {
00051                 if (role != string.Empty)
00052                 {
00053                     role = value;
00054                 }
00055             }
00056             get { return role; }
00057         }
```

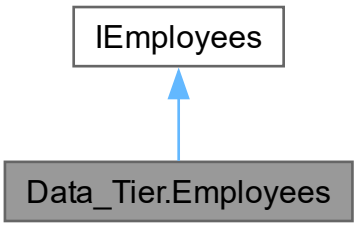
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Object Tier/Employee.cs

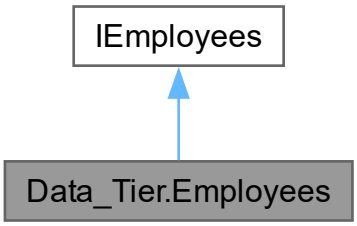
6.8 Data_Tier.Employees Class Reference

Singleton class that manages a list of employees. Allows adding, removing, updating and retrieving employees.

Inheritance diagram for Data_Tier.Employees:



Collaboration diagram for Data_Tier.Employees:



Public Member Functions

- int AddEmployee (Employee employee)

  Adds a new employee to the list and sorts the list.
- bool RemoveEmployee (int idEmployee)

  Removes an employee by their ID from the list.
- bool EmployeeExist (Employee employee)

  Checks if a specific employee exists in the list based on the employee object.
- bool EmployeeExist (int idEmployee)

  Checks if an employee exists in the list based on their ID.
- bool UpdateRole (int idEmployee, string role, double hourly)

  Updates an employee's role and hourly rate.
- Employee GetEmployee (int idEmployee)

  Retrieves an employee based on their ID.

Public Member Functions inherited from Interface_Tier.IEmployees

Protected Member Functions

- Employees ()

  Initializes a new instance of the `Employees` class, with an empty list of employees.

Properties

- static Employees Instance `[get]`

  Gets the singleton instance of the `Employees` class.

6.8.1 Detailed Description

Singleton class that manages a list of employees. Allows adding, removing, updating and retrieving employees.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Employees()

```
Data_Tier.Employees.Employees ()  [protected]
```

Initializes a new instance of the `Employees` class, with an empty list of employees.

```
00071         {
00072             employees = new List<Employee>(5);
00073         }
```

6.8.3 Member Function Documentation

6.8.3.1 AddEmployee()

```
int Data_Tier.Employees.AddEmployee (
            Employee employee)
```

Adds a new employee to the list and sorts the list.

Parameters

employee	The employee to add.

Returns

The ID of the added employee.

Exceptions

ConfigurationErrorException	Throws if the employee is null or if an error occurs during the addition.

Implements Interface_Tier.IEmployees.

```
00106         {
00107             if (employee == null)
00108             {
00109                 throw new ConfigurationErrorException("400");
00110             }
00111             try
00112             {
00113                 employees.Add(employee);
00114                 employees.Sort();
00115                 return employee.Id;
00116             }
00117             catch (Exception ex)
00118             {
00119                 throw new ConfigurationErrorException("401", ex);
00120             }
00121 
00122         }
```

6.8.3.2 EmployeeExist() [1/2]

```
bool Data_Tier.Employees.EmployeeExist (
            Employee employee)
```

Checks if a specific employee exists in the list based on the employee object.

Parameters

employee	The employee to check for existence.

Returns

`true` if the employee exists, otherwise `false`.

```
00158         {
00159             foreach (Employee existingEmployee in employees)
00160             {
00161                 if (existingEmployee - employee)
00162                 {
00163                     return true;
00164                 }
00165             }
00166 
00167             return false;
00168         }
```

6.8.3.3 EmployeeExist() [2/2]

```
bool Data_Tier.Employees.EmployeeExist (
            int idEmployee)
```

Checks if an employee exists in the list based on their ID.

Parameters

idEmployee	The ID of the employee to check.

Returns

`true` if the employee exists, otherwise `false`.

Exceptions

ConfigurationErrorException	Throws if an error occurs during the existence check.

Implements Interface_Tier.IEmployees.

```
00179         {
00180             try
00181             {
00182                 foreach (Employee employee in employees)
00183                 {
00184                     if (employee.Id == idEmployee)
00185                     {
00186                         return true;
00187                     }
00188                 }
00189 
00190                 return false;
00191             }
00192             catch (Exception ex)
00193             {
00194                 throw new ConfigurationErrorException("403", ex);
00195             }
00196 
00197         }
```

6.8.3.4 GetEmployee()

```
Employee Data_Tier.Employees.GetEmployee (
            int idEmployee)
```

Retrieves an employee based on their ID.

Parameters

idEmployee	The ID of the employee to retrieve.

Returns

The employee object if found, otherwise `null`.

Exceptions

ConfigurationErrorException	Throws if the retrieval fails.

Implements Interface_Tier.IEmployees.

```
00237         {
00238             try
00239             {
00240                 return FindEmployee(idEmployee);
00241             }
00242             catch (Exception ex)
00243             {
00244                 throw new ConfigurationErrorException("403", ex);
00245             }
00246         }
```

6.8.3.5 RemoveEmployee()

```
bool Data_Tier.Employees.RemoveEmployee (
            int idEmployee)
```

Removes an employee by their ID from the list.

Parameters

idEmployee	The ID of the employee to remove.

Returns

`true` if the employee was removed, otherwise `false`.

Exceptions

ConfigurationErrorException	Throws if an error occurs during the removal.

Implements Interface_Tier.IEmployees.

```
00133         {
00134             try
00135             {
00136                 Employee employee = FindEmployee(idEmployee);
00137 
00138                 if (employee != null)
00139                 {
00140                     employees.Remove(employee);
00141                     employees.Sort();
00142                     return true;
00143                 }
00144             }
00145             catch (Exception ex)
00146             {
00147                 throw new ConfigurationErrorException("402", ex);
00148             }
00149             return false;
00150         }
```

6.8.3.6 UpdateRole()

```
bool Data_Tier.Employees.UpdateRole (
            int idEmployee,
            string role,
            double hourly)
```

Updates an employee's role and hourly rate.

Parameters

idEmployee	The ID of the employee to update.
role	The new role of the employee.
hourly	The new hourly rate of the employee.

Returns

`true` if the update was successful, otherwise `false`.

Exceptions

ConfigurationErrorException	Throws if the update fails.

Implements Interface_Tier.IEmployees.

```
00208         {
00209             try
00210             {
00211                 Employee employee = FindEmployee(idEmployee);
00212 
00213                 if (employee != null)
00214                 {
00215                     employee.Role = role;
00216                     employee.HourlyRate = hourly;
00217                     return true;
00218                 }
00219             }
00220             catch (Exception ex)
00221             {
00222                 throw new ConfigurationErrorException("404", ex);
00223             }
00224 
00225             return false;
00226         }
```

6.8.4 Property Documentation

6.8.4.1 Instance

```
Employees Data_Tier.Employees.Instance  [static], [get]
```

Gets the singleton instance of the `Employees` class.

```
00043         {
00044             get
00045             {
00046                 if (instance == null)
00047                 {
00048                     instance = new Employees();
00049                 }
00050 
00051                 return instance;
00052             }
00053         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/Employees.cs

6.9 Data_Tier.EmployeesService Class Reference

Class that manages employees associated with projects.

Public Member Functions

- EmployeesService ()

  Initializes a new instance of the EmployeesService class.
- bool ExistExistEmployee (int employeeId)

  Checks if an employee exists in the list of employees for the project.
- bool AddEmployee (int employeeId)

  Adds an employee to the list of employees for the project.
- bool RemoveEmployee (int employeeId)

  Removes an employee from the list of employees for the project.

6.9.1 Detailed Description

Class that manages employees associated with projects.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 EmployeesService()

```
Data_Tier.EmployeesService.EmployeesService ()
```

Initializes a new instance of the EmployeesService class.

```
00036         {
00037             employees = new List<int>(5);
00038         }
```

6.9.3 Member Function Documentation

6.9.3.1 AddEmployee()

```
bool Data_Tier.EmployeesService.AddEmployee (
            int employeeId)
```

Adds an employee to the list of employees for the project.

Parameters

employeeId	The ID of the employee to add.

Returns

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

Exceptions

ConfigurationErrorException	Thrown if an error occurs during the addition of the employee.

```
00061         {
00062             try
00063             {
00064                 if (!ExistExistEmployee(employeeId))
00065                 {
00066                     employees.Add(employeeId);
00067                     return true;
00068                 }
00069
00070             }
00071             catch (Exception ex)
00072             {
00073                 throw new ConfigurationErrorException("810", ex);
00074             }
00075             return false;
00076         }
```

6.9.3.2 ExistExistEmployee()

```
bool Data_Tier.EmployeesService.ExistExistEmployee (
            int employeeId)
```

Checks if an employee exists in the list of employees for the project.

Parameters

employeeId	The ID of the employee to check.

Returns

True if the employee exists in the list; otherwise, false.

```
00050         {
00051             return employees.Contains(employeeId);
00052         }
```

6.9.3.3 RemoveEmployee()

```
bool Data_Tier.EmployeesService.RemoveEmployee (
            int employeeId)
```

Removes an employee from the list of employees for the project.

Parameters

employeeId	The ID of the employee to remove.

Returns

True if the employee was removed successfully; otherwise, false.

Exceptions

ConfigurationErrorException	Thrown if an error occurs during the removal of the employee.

```
00085         {
00086             try
00087             {
00088                 if (ExistExistEmployee(employeeId))
00089                 {
00090                     employees.Remove(employeeId);
00091                     return true;
00092                 }
00093             }
00094             catch (Exception ex)
00095             {
00096                 throw new ConfigurationErrorException("811", ex);
00097             }
00098
00099             return false;
00100         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/Data Tier/EmployeesService.cs

6.10 Unit_Test.EmployeeTest Class Reference

Public Member Functions

- void TestDeleteEmployee ()

6.10.1 Member Function Documentation

6.10.1.1 TestDeleteEmployee()

```
void Unit_Test.EmployeeTest.TestDeleteEmployee ()
00012     {
00013         // Arrange
00014         Employee e1 = Employee.CreateEmployee("Luis","Eng",12.2);
00015 
00016         // Act
00017         Company.RegistEmployee(e1);
00018 
00019         // Assert
00020         Assert.IsTrue(Employees.Instance.EmployeeExist(e1));
00021     }
```

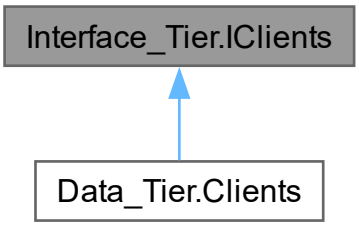
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Unit Test/EmployeeTest.cs

6.11 Interface_Tier.IClients Interface Reference

Methods to implement in the clients class.

Inheritance diagram for Interface_Tier.IClients:



Public Member Functions

- int AddClient (Client client)

  Adds a client to the client list.
- bool RemoveClient (int idClient)

  Removes a client from the client list based on their ID.
- bool ExistClient (int idClient)

  Checks if a client exists in the client list based on their ID.
- bool UpdateContact (int idClient, int contacto)

  Updates the contact information of a client based on their ID.
- Client GetClient (int idClient)

  Retrieves a client from the client list based on their ID.

6.11.1 Detailed Description

Methods to implement in the clients class.

6.11.2 Member Function Documentation

6.11.2.1 AddClient()

```
int Interface_Tier.IClients.AddClient (
            Client client)
```

Adds a client to the client list.

Parameters

client	The client object to add.

Returns

The ID of the added client.

Implemented in Data_Tier.Clients.

6.11.2.2 ExistClient()

```
bool Interface_Tier.IClients.ExistClient (
            int idClient)
```

Checks if a client exists in the client list based on their ID.

Parameters

idClient	The ID of the client to check for.

Returns

Returns true if the client exists, otherwise false.

Implemented in Data_Tier.Clients.

6.11.2.3 GetClient()

```
Client Interface_Tier.IClients.GetClient (
            int idClient)
```

Retrieves a client from the client list based on their ID.

Parameters

idClient	The ID of the client to retrieve.

Returns

The client object with the specified ID, or null if not found.

Implemented in Data_Tier.Clients.

6.11.2.4 RemoveClient()

```
bool Interface_Tier.IClients.RemoveClient (
            int idClient)
```

Removes a client from the client list based on their ID.

Parameters

idClient	The ID of the client to remove.

Returns

Returns true if the client was successfully removed, otherwise false.

Implemented in Data_Tier.Clients.

6.11.2.5 UpdateContact()

```
bool Interface_Tier.IClients.UpdateContact (
            int idClient,
            int contacto)
```

Updates the contact information of a client based on their ID.

Parameters

idClient	The ID of the client to update.
contacto	The new contact information to set.

Returns

Returns true if the contact information was successfully updated, otherwise false.

Implemented in Data_Tier.Clients.

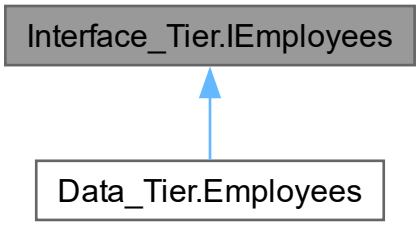
The documentation for this interface was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Interface Tier/IClients.cs

6.12 Interface_Tier.IEmployees Interface Reference

Methods to implement in the employees class.

Inheritance diagram for Interface_Tier.IEmployees:



Public Member Functions

- int AddEmployee (Employee employee)

  Adds a new employee to the list and sorts the list.
- bool RemoveEmployee (int idEmployee)

  Removes an employee by their ID from the list.
- bool EmployeeExist (int idEmployee)

  Checks if an employee exists in the list based on their ID.
- bool UpdateRole (int idEmployee, string role, double hourly)

  Updates an employee's role and hourly rate.
- Employee GetEmployee (int idEmployee)

  Retrieves an employee based on their ID.

6.12.1 Detailed Description

Methods to implement in the employees class.

6.12.2 Member Function Documentation

6.12.2.1 AddEmployee()

```
int Interface_Tier.IEmployees.AddEmployee (
            Employee employee)
```

Adds a new employee to the list and sorts the list.

Parameters

employee	The employee to add.

Returns

The ID of the added employee.

Exceptions

ConfigurationErrorException	Throws if the employee is null or if an error occurs during the addition.

Implemented in Data_Tier.Employees.

6.12.2.2 EmployeeExist()

```
bool Interface_Tier.IEmployees.EmployeeExist (
            int idEmployee)
```

Checks if an employee exists in the list based on their ID.

Parameters

idEmployee	The ID of the employee to check.

Returns

`true` if the employee exists, otherwise `false`.

Exceptions

ConfigurationErrorException	Throws if an error occurs during the existence check.

Implemented in Data_Tier.Employees.

6.12.2.3 GetEmployee()

```
Employee Interface_Tier.IEmployees.GetEmployee (
            int idEmployee)
```

Retrieves an employee based on their ID.

Parameters

idEmployee	The ID of the employee to retrieve.

Returns

The employee object if found, otherwise `null`.

Exceptions

ConfigurationErrorException	Throws if the retrieval fails.

Implemented in Data_Tier.Employees.

6.12.2.4 RemoveEmployee()

```
bool Interface_Tier.IEmployees.RemoveEmployee (
            int idEmployee)
```

Removes an employee by their ID from the list.

Parameters

idEmployee	The ID of the employee to remove.

Returns

`true` if the employee was removed, otherwise `false`.

Exceptions

ConfigurationErrorException	Throws if an error occurs during the removal.

Implemented in Data_Tier.Employees.

6.12.2.5 UpdateRole()

```
bool Interface_Tier.IEmployees.UpdateRole (
            int idEmployee,
            string role,
            double hourly)
```

Updates an employee's role and hourly rate.

Parameters

idEmployee	The ID of the employee to update.
role	The new role of the employee.
hourly	The new hourly rate of the employee.

Returns

`true` if the update was successful, otherwise `false`.

Exceptions

ConfigurationErrorException	Throws if the update fails.

Implemented in Data_Tier.Employees.

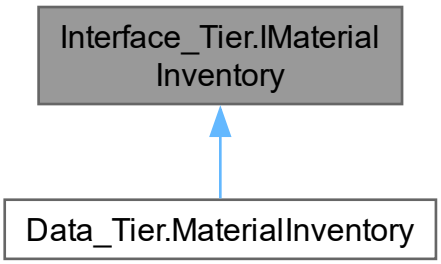
The documentation for this interface was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Interface Tier/IEmployees.cs

6.13 Interface_Tier.IMaterialInventory Interface Reference

Methods to implement in the inventory class.

Inheritance diagram for Interface_Tier.IMaterialInventory:



Public Member Functions

- int AddMaterial (MaterialQuantity inventoryQuantity)
  Adds a new material to the inventory.
- bool VerifyMaterialExistence (int idMaterial)
  Verifies if a material exists in the inventory based on the material ID.
- bool VerifyMaterialQuantity (int idMaterial, int quantity)
  Verifies if there is enough quantity of a material in the inventory.
- bool UpdateQuantity (int idMaterial, int quantityUpdate)
  Updates the quantity of a material in the inventory.
- bool UseMaterial (int idMaterial, int quantity)
  Decreases the quantity of a material in the inventory when it is used.

6.13.1 Detailed Description

Methods to implement in the inventory class.

6.13.2 Member Function Documentation

6.13.2.1 AddMaterial()

```
int Interface_Tier.IMaterialInventory.AddMaterial (
            MaterialQuantity inventoryQuantity)
```

Adds a new material to the inventory.

Parameters

inventoryQuantity	The material quantity to add to the inventory.

Returns

The ID of the added material.

Exceptions

ConfigurationErrorException	Throws an exception if the material is null or if any error occurs during the addition.

Implemented in Data_Tier.MaterialInventory.

6.13.2.2 UpdateQuantity()

```
bool Interface_Tier.IMaterialInventory.UpdateQuantity (
            int idMaterial,
            int quantityUpdate)
```

Updates the quantity of a material in the inventory.

Parameters

idMaterial	The ID of the material to update.
quantityUpdate	The new quantity of the material.

Returns

Returns true if the quantity was updated successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the update process.

Implemented in Data_Tier.MaterialInventory.

6.13.2.3 UseMaterial()

```
bool Interface_Tier.IMaterialInventory.UseMaterial (
            int idMaterial,
            int quantity)
```

Decreases the quantity of a material in the inventory when it is used.

Parameters

idMaterial	The ID of the material to use.
quantity	The quantity of material to use.

Returns

Returns true if the material quantity was successfully decreased, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the use process.

Implemented in Data_Tier.MaterialInventory.

6.13.2.4 VerifyMaterialExistence()

```
bool Interface_Tier.IMaterialInventory.VerifyMaterialExistence (
            int idMaterial)
```

Verifies if a material exists in the inventory based on the material ID.

Parameters

idMaterial	The ID of the material to verify.

Returns

Returns true if the material exists, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the verification process.

Implemented in Data_Tier.MaterialInventory.

6.13.2.5 VerifyMaterialQuantity()

```
bool Interface_Tier.IMaterialInventory.VerifyMaterialQuantity (
            int idMaterial,
            int quantity)
```

Verifies if there is enough quantity of a material in the inventory.

Parameters

idMaterial	The ID of the material to check.
quantity	The quantity of material to check for.

Returns

Returns true if there is enough quantity, otherwise false.

Implemented in Data_Tier.MaterialInventory.

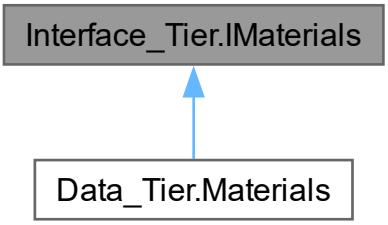
The documentation for this interface was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Interface Tier/IMaterialInventory.cs

6.14 Interface_Tier.IMaterials Interface Reference

Methods to implement in the materials class.

Inheritance diagram for Interface_Tier.IMaterials:



Public Member Functions

- int AddMaterial (Material material)

  Adds a new material to the materials list.
- bool MaterialExist (int idMaterial)

  Checks if a material exists in the list based on its ID.
- bool UpdatePrice (int idMaterial, double price)

  Updates the price of a material in the list based on its ID.

6.14.1 Detailed Description

Methods to implement in the materials class.

6.14.2 Member Function Documentation

6.14.2.1 AddMaterial()

```
int Interface_Tier.IMaterials.AddMaterial (
            Material material)
```

Adds a new material to the materials list.

Parameters

material	The material to add.

Returns

The ID of the added material.

Exceptions

ConfigurationErrorException	Throws an exception if the material is null or any error occurs during the addition.

Implemented in Data_Tier.Materials.

6.14.2.2 MaterialExist()

```
bool Interface_Tier.IMaterials.MaterialExist (
            int idMaterial)
```

Checks if a material exists in the list based on its ID.

Parameters

idMaterial	The ID of the material to check.

Returns

True if the material exists, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the check.

Implemented in Data_Tier.Materials.

6.14.2.3 UpdatePrice()

```
bool Interface_Tier.IMaterials.UpdatePrice (
            int idMaterial,
            double price)
```

Updates the price of a material in the list based on its ID.

Parameters

idMaterial	The ID of the material to update.
price	The new price to set.

Returns

True if the price was updated successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the update.

Implemented in Data_Tier.Materials.

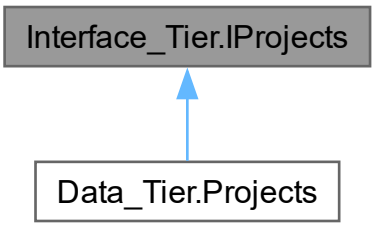
The documentation for this interface was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Interface Tier/IMaterials.cs

6.15 Interface_Tier.IProjects Interface Reference

Methods to implement in the projects class.

Inheritance diagram for Interface_Tier.IProjects:



Public Member Functions

- bool RemoveProject (int idProject)

  Removes a project from the projects list by its ID.
- bool ProjectExists (int idProject)

  Checks if a project exists by its ID.
- bool CloseProject (int idProject)

  Closes a project by setting its end date and status to completed.
- bool AddClient (int idProject, int idClient)

  Adds a client to a project.
- bool RemoveClient (int idProject, int idClient)

  Removes a client from a project.
- bool AddEmployee (int idProject, int idEmployee)

  Adds an employee to a project.
- bool RemoveEmployee (int idProject, int idEmployee)

  Removes an employee from a project.
- bool UseMaterial (int idProject, int idMaterial, int quantity)

  Uses material in a project.

6.15.1 Detailed Description

Methods to implement in the projects class.

6.15.2 Member Function Documentation

6.15.2.1 AddClient()

```
bool Interface_Tier.IProjects.AddClient (
            int idProject,
            int idClient)
```

Adds a client to a project.

Parameters

idProject	The ID of the project to add the client to.
idClient	The ID of the client to add.

Returns

True if the client was added successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the addition.

Implemented in Data_Tier.Projects.

6.15.2.2 AddEmployee()

```
bool Interface_Tier.IProjects.AddEmployee (
            int idProject,
            int idEmployee)
```

Adds an employee to a project.

Parameters

idProject	The ID of the project to add the employee to.
idEmployee	The ID of the employee to add.

Returns

True if the employee was added successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the addition.

Implemented in Data_Tier.Projects.

6.15.2.3 CloseProject()

```
bool Interface_Tier.IProjects.CloseProject (
            int idProject)
```

Closes a project by setting its end date and status to completed.

Parameters

idProject	The ID of the project to close.

Returns

True if the project was successfully closed, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the closure.

Implemented in Data_Tier.Projects.

6.15.2.4 ProjectExists()

```
bool Interface_Tier.IProjects.ProjectExists (
            int idProject)
```

Checks if a project exists by its ID.

Parameters

projectId	The ID of the project to check.

Returns

True if the project exists, otherwise false.

Implemented in Data_Tier.Projects.

6.15.2.5 RemoveClient()

```
bool Interface_Tier.IProjects.RemoveClient (
            int idProject,
            int idClient)
```

Removes a client from a project.

Parameters

idProject	The ID of the project to remove the client from.
idClient	The ID of the client to remove.

Returns

True if the client was removed successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal.

Implemented in Data_Tier.Projects.

6.15.2.6 RemoveEmployee()

```
bool Interface_Tier.IProjects.RemoveEmployee (
            int idProject,
            int idEmployee)
```

Removes an employee from a project.

Parameters

idProject	The ID of the project to remove the employee from.
idEmployee	The ID of the employee to remove.

Returns

True if the employee was removed successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal.

Implemented in Data_Tier.Projects.

6.15.2.7 RemoveProject()

```
bool Interface_Tier.IProjects.RemoveProject (
            int idProject)
```

Removes a project from the projects list by its ID.

Parameters

idProject	The ID of the project to remove.

Returns

True if the project was successfully removed, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal.

Implemented in Data_Tier.Projects.

6.15.2.8 UseMaterial()

```
bool Interface_Tier.IProjects.UseMaterial (
            int idProject,
            int idMaterial,
            int quantity)
```

Uses material in a project.

Parameters

idProject	The ID of the project to use material in.
idMaterial	The ID of the material to use.
quantity	The quantity of the material to use.

Returns

True if the material was used successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the material use.

Implemented in Data_Tier.Projects.

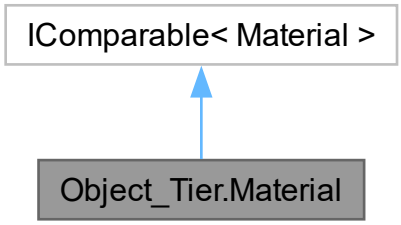
The documentation for this interface was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Interface Tier/IProjects.cs

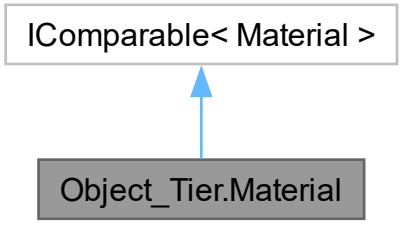
6.16 Object_Tier.Material Class Reference

Represents a material with ID, name, unit price, and registration date.

Inheritance diagram for Object_Tier.Material:



Collaboration diagram for Object_Tier.Material:



Public Member Functions

- Material (string name, double price)

  Initializes a new instance of the Material class with a name and unit price. Automatically assigns a unique ID.
- override bool Equals (object obj)

  Determines whether the specified object is equal to the current material. Materials are equal if their name and unit price match.
- override string ToString ()

  Returns a string representation of the material, including ID, name, and unit price.
- int CompareTo (Material material)

  Compares the current material to another material based on their name.

Static Public Member Functions

- static Material CreateMaterial (string name, double price)

  Creates a new Material instance.
- static bool operator- (Material material1, Material material2)

  Checks if two materials are equal using the "-" operator.
- static bool operator+ (Material material1, Material material2)

  Checks if two materials are not equal using the "+" operator.
- static bool getNextMaterialId ()

  Increments the static material ID counter.

Properties

- int Id `[get]`

  Gets the ID of the material. Read-only.
- string Name `[get, set]`

  Gets or sets the name of the material.
- double UnitPrice `[get, set]`

  Gets or sets the unit price of the material. The value must be greater than zero.
- DateTime LastRegiste `[get, set]`

  Gets or sets the last registration date of the material.

6.16.1 Detailed Description

Represents a material with ID, name, unit price, and registration date.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Material()

```
Object_Tier.Material.Material (
            string name,
            double price)
```

Initializes a new instance of the Material class with a name and unit price. Automatically assigns a unique ID.

Parameters

name	The name of the material.
price	The unit price of the material.

```
00103         {
00104             id = materialIdCounter++;
00105             Name = name.ToUpper().Trim();
00106             UnitPrice = price;
00107             LastRegiste = DateTime.Now;
00108         }
```

6.16.3 Member Function Documentation

6.16.3.1 CompareTo()

```
int Object_Tier.Material.CompareTo (
            Material material)
```

Compares the current material to another material based on their name.

Parameters

material	The material to compare to.

Returns

A value indicating the relative order of the materials.

```
00196         {
00197             return Name.CompareTo(material.Name);
00198         }
```

6.16.3.2 CreateMaterial()

```
static Material Object_Tier.Material.CreateMaterial (
            string name,
            double price)  [static]
```

Creates a new Material instance.

Parameters

name	The name of the material.
price	The unit price of the material.

Returns

A new Material object.

```
00159         {
00160             return new Material(name, price);
00161         }
```

6.16.3.3 Equals()

```
override bool Object_Tier.Material.Equals (
            object obj)
```

Determines whether the specified object is equal to the current material. Materials are equal if their name and unit price match.

Parameters

obj	The object to compare with the current material.

Returns

True if the objects are equal; otherwise, false.

```
00121         {
00122             if (obj == null)
00123             {
00124                 return false;
00125             }
00126 
00127             if (obj is Material)
00128             {
00129                 Material otherMaterial = obj as Material;
00130 
00131                 if (name == otherMaterial.Name && unitPrice == otherMaterial.UnitPrice)
00132                 {
00133                     return true;
00134                 }
00135             }
00136 
00137             return false;
00138         }
```

6.16.3.4 getNextMaterialId()

```
static bool Object_Tier.Material.getNextMaterialId ()  [static]
```

Increments the static material ID counter.

Returns

True after incrementing the counter.

```
00205         {
00206             materialIdCounter++;
00207             return true;
00208         }
```

6.16.3.5 operator+()

```
static bool Object_Tier.Material.operator+ (
            Material material1,
            Material material2)  [static]
```

Checks if two materials are not equal using the "+" operator.

Parameters

material1	The first material.
material2	The second material.

Returns

True if the materials are not equal; otherwise, false.

```
00186         {
00187             return !(material1 - material2);
00188         }
```

6.16.3.6 operator-()

```
static bool Object_Tier.Material.operator- (
            Material material1,
            Material material2)  [static]
```

Checks if two materials are equal using the "-" operator.

Parameters

material1	The first material.
material2	The second material.

Returns

True if the materials are equal; otherwise, false.

```
00170         {
00171             if (material1.Equals(material2))
00172             {
00173                 return true;
00174             }
00175 
00176             return false;
00177         }
```

6.16.3.7 ToString()

```
override string Object_Tier.Material.ToString ()
```

Returns a string representation of the material, including ID, name, and unit price.

Returns

A string representation of the material.

```
00145         {
00146             return Id + " " + Name + " " + UnitPrice;
00147         }
```

6.16.4 Property Documentation

6.16.4.1 Id

```
int Object_Tier.Material.Id  [get]
```

Gets the ID of the material. Read-only.

```
00055         {
00056             get { return id; }
00057         }
```

6.16.4.2 LastRegiste

```
DateTime Object_Tier.Material.LastRegiste  [get], [set]
```

Gets or sets the last registration date of the material.

```
00088         {
00089             set { lastRegiste = value; }
00090             get { return lastRegiste; }
00091         }
```

6.16.4.3 Name

```
string Object_Tier.Material.Name  [get], [set]
```

Gets or sets the name of the material.

```
00063         {
00064             set { name = value; }
00065             get { return name; }
00066         }
```

6.16.4.4 UnitPrice

```
double Object_Tier.Material.UnitPrice  [get], [set]
```

Gets or sets the unit price of the material. The value must be greater than zero.

```
00073         {
00074             set
00075             {
00076                 if (value > 0)
00077                 {
00078                     unitPrice = value;
00079                 }
00080             }
00081             get { return unitPrice; }
00082         }
```

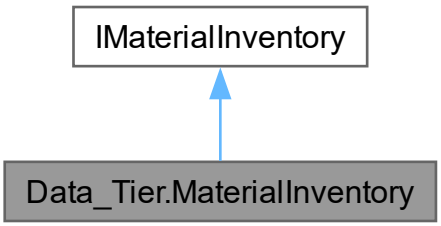
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Object Tier/Material.cs

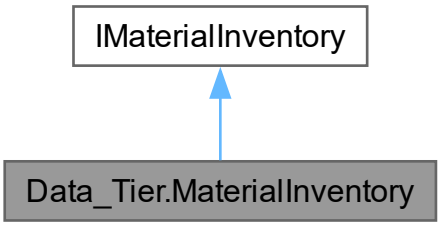
6.17 Data_Tier.MaterialInventory Class Reference

Singleton class that manages the material inventory. Allows adding, removing, updating, and retrieving materials.

Inheritance diagram for Data_Tier.MaterialInventory:



Collaboration diagram for Data_Tier.MaterialInventory:



Public Member Functions

- int AddMaterial (MaterialQuantity inventoryQuantity)

  Adds a new material to the inventory.
- bool VerifyMaterialExistence (int idMaterial)

  Verifies if a material exists in the inventory based on the material ID.
- bool VerifyMaterialQuantity (int idMaterial, int quantity)

  Verifies if there is enough quantity of a material in the inventory.
- bool UpdateQuantity (int idMaterial, int quantityUpdate)

  Updates the quantity of a material in the inventory.
- bool UseMaterial (int idMaterial, int quantity)

  Decreases the quantity of a material in the inventory when it is used.
- MaterialQuantity GetMaterialQuantity (int idMaterial)

  Retrieves the material and its quantity from the inventory based on the material ID.

Public Member Functions inherited from Interface_Tier.IMaterialInventory

Protected Member Functions

- MaterialInventory ()

  Initializes a new instance of the MaterialInventory class with an empty inventory.

Properties

- static MaterialInventory Instance `[get]`

  Gets the singleton instance of the MaterialInventory class.

6.17.1 Detailed Description

Singleton class that manages the material inventory. Allows adding, removing, updating, and retrieving materials.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 MaterialInventory()

```
Data_Tier.MaterialInventory.MaterialInventory ()  [protected]
```

Initializes a new instance of the MaterialInventory class with an empty inventory.

```
00070         {
00071             inventory = new List<MaterialQuantity>(5);
00072         }
```

6.17.3 Member Function Documentation

6.17.3.1 AddMaterial()

```
int Data_Tier.MaterialInventory.AddMaterial (
            MaterialQuantity inventoryQuantity)
```

Adds a new material to the inventory.

Parameters

inventoryQuantity	The material quantity to add to the inventory.

Returns

The ID of the added material.

Exceptions

ConfigurationErrorException	Throws an exception if the material is null or if any error occurs during the addition.

Implements Interface_Tier.IMaterialInventory.

```
00105         {
00106             if (inventoryQuantity == null)
00107             {
00108                 throw new ConfigurationErrorException("601");
00109             }
00110             try
00111             {
00112                 inventory.Add(inventoryQuantity);
00113                 return inventoryQuantity.IdMaterial;
00114             }
00115             catch (Exception ex)
00116             {
00117                 throw new ConfigurationErrorException("603", ex);
00118             }
00119         }
```

6.17.3.2 GetMaterialQuantity()

```
MaterialQuantity Data_Tier.MaterialInventory.GetMaterialQuantity (
            int idMaterial)
```

Retrieves the material and its quantity from the inventory based on the material ID.

Parameters

idMaterial	The ID of the material to retrieve.

Returns

Returns the material quantity object if found, otherwise null.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during retrieval.

```
00236         {
00237             try
00238             {
00239                 MaterialQuantity material = FindMaterial(idMaterial);
00240 
00241                 if (material != null)
00242                 {
00243                     return material;
00244                 }
00245             }
00246             catch (Exception ex)
00247             {
00248                 throw new ConfigurationErrorException("617", ex);
00249             }
00250 
00251             return null;
00252         }
```

6.17.3.3 UpdateQuantity()

```
bool Data_Tier.MaterialInventory.UpdateQuantity (
            int idMaterial,
            int quantityUpdate)
```

Updates the quantity of a material in the inventory.

Parameters

idMaterial	The ID of the material to update.
quantityUpdate	The new quantity of the material.

Returns

Returns true if the quantity was updated successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the update process.

Implements Interface_Tier.IMaterialInventory.

```
00179         {
00180             try
00181             {
00182                 MaterialQuantity material = FindMaterial(idMaterial);
00183 
00184                 if (material != null)
00185                 {
00186                     material.Quantity = quantityUpdate;
00187                     return true;
00188                 }
00189             }
00190             catch (Exception ex)
00191             {
00192                 throw new ConfigurationErrorException("607", ex);
00193             }
00194 
00195             return false;
00196         }
```

6.17.3.4 UseMaterial()

```
bool Data_Tier.MaterialInventory.UseMaterial (
            int idMaterial,
            int quantity)
```

Decreases the quantity of a material in the inventory when it is used.

Parameters

idMaterial	The ID of the material to use.
quantity	The quantity of material to use.

Returns

Returns true if the material quantity was successfully decreased, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the use process.

Implements Interface_Tier.IMaterialInventory.

```
00208         {
00209             try
00210             {
00211                 MaterialQuantity material = FindMaterial(idMaterial);
00212 
00213                 if (material != null)
00214                 {
00215                     material.Quantity -= quantity;
00216                     return true;
00217                 }
00218             }
00219             catch (Exception ex)
00220             {
00221                 throw new ConfigurationErrorException("608", ex);
00222             }
00223 
00224             return false;
00225         }
```

6.17.3.5 VerifyMaterialExistence()

```
bool Data_Tier.MaterialInventory.VerifyMaterialExistence (
            int idMaterial)
```

Verifies if a material exists in the inventory based on the material ID.

Parameters

idMaterial	The ID of the material to verify.

Returns

Returns true if the material exists, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the verification process.

Implements Interface_Tier.IMaterialInventory.

```
00130         {
00131             try
00132             {
00133                 foreach (MaterialQuantity material in inventory)
00134                 {
00135                     if (material.IdMaterial == idMaterial)
00136                     {
00137                         return true;
00138                     }
00139                 }
00140                 return false;
00141             }
00142             catch (Exception ex)
00143             {
00144                 throw new ConfigurationErrorException("605", ex);
00145             }
00146         }
```

6.17.3.6 VerifyMaterialQuantity()

```
bool Data_Tier.MaterialInventory.VerifyMaterialQuantity (
            int idMaterial,
            int quantity)
```

Verifies if there is enough quantity of a material in the inventory.

Parameters

idMaterial	The ID of the material to check.
quantity	The quantity of material to check for.

Returns

Returns true if there is enough quantity, otherwise false.

Implements Interface_Tier.IMaterialInventory.

```
00155         {
00156             MaterialQuantity material = FindMaterial(idMaterial);
00157
00158             if (material != null)
00159             {
00160                 if (material.Quantity >= quantity)
00161                 {
00162                     return true;
00163                 }
00164             }
00165
00166             return false;
00167         }
```

6.17.4 Property Documentation

6.17.4.1 Instance

```
MaterialInventory Data_Tier.MaterialInventory.Instance  [static], [get]
```

Gets the singleton instance of the MaterialInventory class.

```
00043         {
00044             get
00045             {
00046                 if (instance == null)
00047                 {
00048                     instance = new MaterialInventory();
00049                 }
00050 
00051                 return instance;
00052             }
00053         }
```

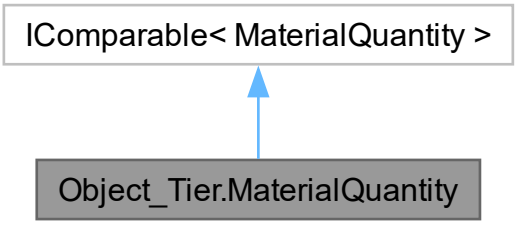
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/MaterialInventory.cs

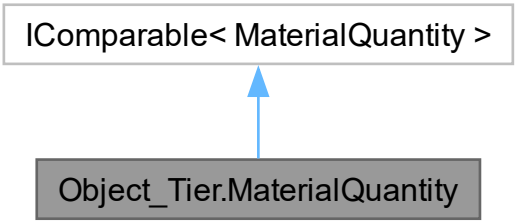
6.18 Object_Tier.MaterialQuantity Class Reference

Represents the quantity of a material and the date it was added.

Inheritance diagram for Object_Tier.MaterialQuantity:



Collaboration diagram for Object_Tier.MaterialQuantity:



Public Member Functions

- MaterialQuantity (int id, int quantity)

  Initializes a new instance of the MaterialQuantity class with an ID and quantity. The date is set to the current date and time.
- override bool Equals (object obj)

  Checks if the current MaterialQuantity is equal to another object. Two MaterialQuantity objects are equal if their material IDs are the same.
- override string ToString ()

  Returns a string representation of the MaterialQuantity. Includes the material ID, date, and quantity.
- int CompareTo (MaterialQuantity material)

  Compares the current MaterialQuantity to another based on their material IDs.

Static Public Member Functions

- static MaterialQuantity CreateMaterialQuantity (int id, int quantity)

  Creates a new MaterialQuantity instance.
- static bool operator- (MaterialQuantity material1, MaterialQuantity material2)

  Checks if two MaterialQuantity objects are equal using the "-" operator.
- static bool operator+ (MaterialQuantity material1, MaterialQuantity material2)

  Checks if two MaterialQuantity objects are not equal using the "+" operator.

Properties

- int IdMaterial `[get]`

  Gets the ID of the material.
- int Quantity `[get, set]`

  Gets or sets the quantity of the material. The value must be zero or greater.
- DateTime Date `[get, set]`

  Gets or sets the date the material was added.

6.18.1 Detailed Description

Represents the quantity of a material and the date it was added.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 MaterialQuantity()

```
Object_Tier.MaterialQuantity.MaterialQuantity (
            int id,
            int quantity)
```

Initializes a new instance of the MaterialQuantity class with an ID and quantity. The date is set to the current date and time.

Parameters

id	
quantity	

```
00085         {
00086             this.idMaterial = id;
00087             Quantity = quantity;
00088             date = DateTime.Now;
00089         }
```

6.18.3 Member Function Documentation

6.18.3.1 CompareTo()

```
int Object_Tier.MaterialQuantity.CompareTo (
            MaterialQuantity material)
```

Compares the current MaterialQuantity to another based on their material IDs.

Parameters

material	The MaterialQuantity to compare to.

Returns

A value indicating the relative order of the objects.

```
00137         {
00138             return IdMaterial.CompareTo(material.IdMaterial);
00139         }
```

6.18.3.2 CreateMaterialQuantity()

```
static MaterialQuantity Object_Tier.MaterialQuantity.CreateMaterialQuantity (
            int id,
            int quantity)  [static]
```

Creates a new MaterialQuantity instance.

Parameters

id	The ID of the material.
quantity	The quantity of the material.

Returns

A new MaterialQuantity object.

```
00151         {
00152             return new MaterialQuantity(id, quantity);
00153         }
```

6.18.3.3 Equals()

```
override bool Object_Tier.MaterialQuantity.Equals (
            object obj)
```

Checks if the current MaterialQuantity is equal to another object. Two MaterialQuantity objects are equal if their material IDs are the same.

Parameters

obj	The object to compare to.

Returns

True if the objects are equal; otherwise, false.

```
00102         {
00103             if (obj == null)
00104             {
00105                 return false;
00106             }
00107 
00108             if (obj is MaterialQuantity)
00109             {
00110                 MaterialQuantity othermaterial = obj as MaterialQuantity;
00111 
00112                 if (idMaterial == othermaterial.idMaterial)
00113                 {
00114                     return true;
00115                 }
00116             }
00117 
00118             return false;
00119         }
```

6.18.3.4 operator+()

```
static bool Object_Tier.MaterialQuantity.operator+ (
             MaterialQuantity material1,
             MaterialQuantity material2)  [static]
```

Checks if two MaterialQuantity objects are not equal using the "+" operator.

Parameters

material1	The first MaterialQuantity.
material2	The second MaterialQuantity.

Returns

True if the objects are not equal; otherwise, false.

```
00178         {
00179             return !(material1 - material2);
00180         }
```

6.18.3.5 operator-()

```
static bool Object_Tier.MaterialQuantity.operator- (
             MaterialQuantity material1,
             MaterialQuantity material2)  [static]
```

Checks if two MaterialQuantity objects are equal using the "-" operator.

Parameters

material1	The first MaterialQuantity.
material2	The second MaterialQuantity.

Returns

True if the objects are equal; otherwise, false.

```
00162         {
00163             if (material1.Equals(material2))
00164             {
00165                 return true;
00166             }
00167 
00168             return false;
00169         }
```

6.18.3.6 ToString()

```
override string Object_Tier.MaterialQuantity.ToString ()
```

Returns a string representation of the MaterialQuantity. Includes the material ID, date, and quantity.

Returns

A string representing the MaterialQuantity.

```
00127         {
00128             return idMaterial + " " + Date + " " + Quantity;
00129         }
```

6.18.4 Property Documentation

6.18.4.1 Date

```
DateTime Object_Tier.MaterialQuantity.Date  [get], [set]
```

Gets or sets the date the material was added.

```
00070         {
00071             set { date = value; }
00072             get { return date; }
00073         }
```

6.18.4.2 IdMaterial

```
int Object_Tier.MaterialQuantity.IdMaterial  [get]
```

Gets the ID of the material.

```
00046         {
00047             get { return idMaterial; }
00048         }
```

6.18.4.3 Quantity

```
int Object_Tier.MaterialQuantity.Quantity  [get], [set]
```

Gets or sets the quantity of the material. The value must be zero or greater.

```
00055         {
00056             set
00057             {
00058                 if (value >= 0)
00059                 {
00060                     quantity = value;
00061                 }
00062             }
00063             get { return quantity; }
00064         }
```

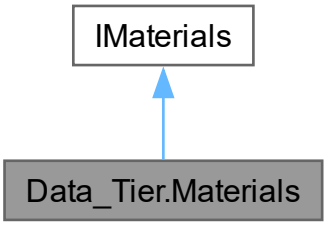
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Object Tier/MaterialQuantity.cs

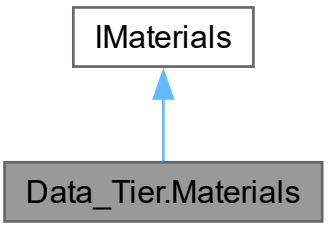
6.19 Data_Tier.Materials Class Reference

Singleton class that manages the materials in the system. Allows adding, checking, updating, and retrieving materials.

Inheritance diagram for Data_Tier.Materials:



Collaboration diagram for Data_Tier.Materials:



Public Member Functions

- int AddMaterial (Material material)

  Adds a new material to the materials list.
- bool MaterialExist (Material material)

  Checks if a material already exists in the list.
- bool MaterialExist (int idMaterial)

  Checks if a material exists in the list based on its ID.
- bool UpdatePrice (int idMaterial, double price)

  Updates the price of a material in the list based on its ID.
- Material GetMaterial (int idMaterial)

  Retrieves a material by its ID.

Public Member Functions inherited from Interface_Tier.IMaterials

Protected Member Functions

- Materials ()

  Initializes a new instance of the Materials class with an empty list of materials.

Properties

- static Materials Instance `[get]`

  Gets the singleton instance of the Materials class.

6.19.1 Detailed Description

Singleton class that manages the materials in the system. Allows adding, checking, updating, and retrieving materials.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Materials()

```
Data_Tier.Materials.Materials ()  [protected]
```

Initializes a new instance of the Materials class with an empty list of materials.

```
00072         {
00073             materials = new List<Material>(5);
00074         }
```

6.19.3 Member Function Documentation

6.19.3.1 AddMaterial()

```
int Data_Tier.Materials.AddMaterial (
            Material material)
```

Adds a new material to the materials list.

Parameters

material	The material to add.

Returns

The ID of the added material.

Exceptions

ConfigurationErrorException	Throws an exception if the material is null or any error occurs during the addition.

Implements Interface_Tier.IMaterials.

```
00107         {
00108             if (material == null)
00109             {
00110                 throw new ConfigurationErrorException("600");
00111             }
00112             try
00113             {
00114                 materials.Add(material);
00115                 return material.Id;
00116             }
00117             catch (Exception ex)
00118             {
00119                 throw new ConfigurationErrorException("602", ex);
00120             }
00121         }
```

6.19.3.2 GetMaterial()

```
Material Data_Tier.Materials.GetMaterial (
            int idMaterial)
```

Retrieves a material by its ID.

Parameters

idMaterial	The ID of the material to retrieve.

Returns

The material if found, otherwise null.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the retrieval.

```
00205         {
00206             try
00207             {
00208                 Material material = FindMaterial(idMaterial);
00209 
00210                 if (material != null)
00211                 {
00212                     return material;
00213                 }
00214             }
00215             catch (Exception ex)
00216             {
00217                 throw new ConfigurationErrorException("619", ex);
00218             }
00219             return null;
00220         }
```

6.19.3.3 MaterialExist() [1/2]

```
bool Data_Tier.Materials.MaterialExist (
            int idMaterial)
```

Checks if a material exists in the list based on its ID.

Parameters

idMaterial	The ID of the material to check.

Returns

True if the material exists, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the check.

Implements Interface_Tier.IMaterials.

```
00149         {
00150             try
00151             {
00152                 foreach (Material material in materials)
00153                 {
00154                     if (material.Id == idMaterial)
00155                     {
00156                         return true;
00157                     }
00158                 }
00159                 return false;
00160             }
00161             catch (Exception ex)
00162             {
00163                 throw new ConfigurationErrorException("604", ex);
00164             }
00165         }
```

6.19.3.4 MaterialExist() [2/2]

```
bool Data_Tier.Materials.MaterialExist (
            Material material)
```

Checks if a material already exists in the list.

Parameters

material	The material to check for existence.

Returns

True if the material exists, otherwise false.

```
00129         {
00130             foreach (Material existingMaterial in materials)
00131             {
00132                 if (existingMaterial - material)
00133                 {
00134                     return true;
00135                 }
00136             }
00137             return false;
00138         }
```

6.19.3.5 UpdatePrice()

```
bool Data_Tier.Materials.UpdatePrice (
            int idMaterial,
            double price)
```

Updates the price of a material in the list based on its ID.

Parameters

idMaterial	The ID of the material to update.
price	The new price to set.

Returns

True if the price was updated successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the update.

Implements Interface_Tier.IMaterials.

```
00177         {
00178             try
00179             {
00180                 Material material = FindMaterial(idMaterial);
00181 
00182                 if (material != null)
00183                 {
00184                     material.UnitPrice = price;
00185                     return true;
00186                 }
00187             }
00188             catch (Exception ex)
00189             {
00190                 throw new ConfigurationErrorException("606", ex);
00191             }
00192 
00193             return false;
00194         }
```

6.19.4 Property Documentation

6.19.4.1 Instance

```
Materials Data_Tier.Materials.Instance  [static], [get]
```

Gets the singleton instance of the Materials class.

```
00044         {
00045             get
00046             {
00047                 if (instance == null)
00048                 {
00049                     instance = new Materials();
00050                 }
00051 
00052                 return instance;
00053             }
00054         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/Materials.cs

6.20 Data_Layer.MaterialService Class Reference

Class for managing materials used in a project.

Public Member Functions

- MaterialService ()

  Initializes a new instance of the MaterialService class.
- bool ExistExistEmployee (MaterialQuantity material)

  Checks if the specified material exists in the list of used materials.
- bool AddMaterial (MaterialQuantity material)

  Adds a material to the list of used materials. If the material already exists, its quantity is updated.

6.20.1 Detailed Description

Class for managing materials used in a project.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 MaterialService()

```
Data_Layer.MaterialService.MaterialService ()
```

Initializes a new instance of the MaterialService class.

```
00038         {
00039             use = new List<MaterialQuantity>(5);
00040         }
```

6.20.3 Member Function Documentation

6.20.3.1 AddMaterial()

```
bool Data_Layer.MaterialService.AddMaterial (
            MaterialQuantity material)
```

Adds a material to the list of used materials. If the material already exists, its quantity is updated.

Parameters

material	The material to add or update.

Returns

True if the material was added or updated successfully; otherwise, false.

Exceptions

ConfigurationErrorException	Thrown if an error occurs during material addition.

```
00081         {
00082             try
00083             {
00084                 if (!ExistExistEmployee(material))
00085                 {
00086                     use.Add(material);
00087                     return true;
00088                 }
00089                 else
00090                 {
00091                     MaterialQuantity materialQuantity = FindMaterial(material);
00092                     materialQuantity.Quantity += material.Quantity;
00093                     materialQuantity.Date = DateTime.Now;
00094                     return true;
00095                 }
00096             }
00097             catch (Exception ex)
00098             {
00099                 throw new ConfigurationErrorException("814", ex);
00100             }
00101         }
```

6.20.3.2 ExistExistEmployee()

```
bool Data_Layer.MaterialService.ExistExistEmployee (
            MaterialQuantity material)
```

Checks if the specified material exists in the list of used materials.

Parameters

material	The material to check for existence.

Returns

True if the material exists in the list, otherwise false.

```
00070         {
00071             return use.Contains(material);
00072         }
```

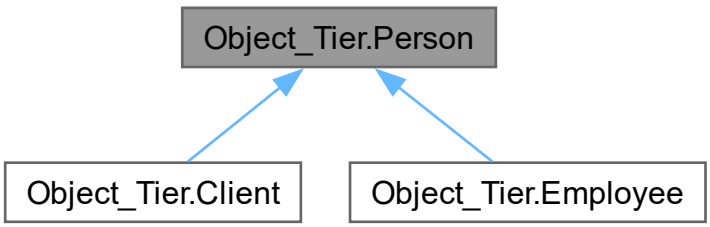
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/MaterialService.cs

6.21 Object_Tier.Person Class Reference

Represents a person with an ID and a name.

Inheritance diagram for Object_Tier.Person:



Protected Member Functions

- Person (int id, string name)

  Initializes a new instance of the Person class with an ID and a name. The name is automatically formatted.

Properties

- int Id [get]
  Gets the ID of the person.
- string Name [get, set]
  Gets or sets the name of the person.

6.21.1 Detailed Description

Represents a person with an ID and a name.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Person()

```
Object_Tier.Person.Person (
            int id,
            string name)  [protected]
```

Initializes a new instance of the Person class with an ID and a name. The name is automatically formatted.

Parameters

id	The ID of the person.
name	The name of the person.

```
00062         {
00063             this.id = id;
00064             Name = name;
00065         }
```

6.21.3 Property Documentation

6.21.3.1 Id

```
int Object_Tier.Person.Id  [get]
```

Gets the ID of the person.

```
00039         {
00040             get { return id; }
00041         }
```

6.21.3.2 Name

```
string Object_Tier.Person.Name  [get], [set]
```

Gets or sets the name of the person.

```
00047         {
00048             set { name = stringFormatada(value); }
00049             get { return name; }
00050         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Object Tier/Person.cs

6.22 Object_Tier.Project Class Reference

Represents a project with information about its status, start date, end date, and ID management.

Public Member Functions

- Project (Status status)

  Initializes a new instance of the Project class with a specified status. The start date is set to the current date, and the end date is initialized to the default value.
- override bool Equals (object obj)

  Determines whether the specified object is equal to the current project.
- override string ToString ()

  Returns a string representation of the project.

Static Public Member Functions

- static Project CreateProject (Status status)

  Creates a new project with the specified status.
- static bool operator- (Project project1, Project project2)

  Compares two projects for equality based on their properties.
- static bool operator+ (Project project1, Project project2)

  Determines whether two projects are not equal.
- static bool getNextProjectId ()

  Increments the static project ID counter.

Properties

- int Id `[get, set]`

  Gets or sets the project ID.
- Status Status `[get, set]`

  Gets or sets the project status.
- DateTime StartDate `[get]`

  Gets the start date of the project.
- DateTime EndDate `[get, set]`

  Gets or sets the end date of the project.

6.22.1 Detailed Description

Represents a project with information about its status, start date, end date, and ID management.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Project()

```
Object_Tier.Project.Project (
            Status status)
```

Initializes a new instance of the Project class with a specified status. The start date is set to the current date, and the end date is initialized to the default value.

Parameters

status	The initial status of the project.

```
00110         {
00111             Id = projectIdCounter++;
00112             Status = status;
00113             startDate = DateTime.Now;
00114             EndDate = new DateTime();
00115         }
```

6.22.3 Member Function Documentation

6.22.3.1 CreateProject()

```
static Project Object_Tier.Project.CreateProject (
            Status status)  [static]
```

Creates a new project with the specified status.

Parameters

status	The status of the new project.

Returns

A new Project instance.

```
00164         {
00165             return new Project(status);
00166         }
```

6.22.3.2 Equals()

```
override bool Object_Tier.Project.Equals (
            object obj)
```

Determines whether the specified object is equal to the current project.

Parameters

obj	The object to compare with the current project.

Returns

True if the objects are equal; otherwise, false.

```
00127         {
00128             if (obj == null)
00129             {
00130                 return false;
00131             }
00132 
00133             if (obj is Project)
00134             {
00135                 Project otherproject = obj as Project;
00136 
00137                 if (StartDate == otherproject.StartDate)
00138                 {
00139                     return true;
00140                 }
00141             }
00142 
00143             return false;
00144         }
```

6.22.3.3 getNextProjectId()

```
static bool Object_Tier.Project.getNextProjectId ()  [static]
```

Increments the static project ID counter.

Returns

True after incrementing the counter.

```
00201         {
00202             projectIdCounter++;
00203             return true;
00204         }
```

6.22.3.4 operator+()

```
static bool Object_Tier.Project.operator+ (
             Project project1,
             Project project2)  [static]
```

Determines whether two projects are not equal.

Parameters

project1	The first project.
project2	The second project.

Returns

True if the projects are not equal; otherwise, false.

```
00192         {
00193             return !(project1 - project2);
00194         }
```

6.22.3.5 operator-()

```
static bool Object_Tier.Project.operator- (
             Project project1,
             Project project2)  [static]
```

Compares two projects for equality based on their properties.

Parameters

project1	The first project.
project2	The second project.

Returns

True if the projects are equal; otherwise, false.

```
00176         {
00177             if (project1.Equals(project2))
00178             {
00179                 return true;
00180             }
00181
00182             return false;
00183         }
```

6.22.3.6 ToString()

```
override string Object_Tier.Project.ToString ()
```

Returns a string representation of the project.

Returns

A string containing the project ID, status, start date, and end date.

```
00151         {
00152             return Id + " " + Status + " " +StartDate + " " + EndDate;
00153         }
```

6.22.4 Property Documentation

6.22.4.1 EndDate

```
DateTime Object_Tier.Project.EndDate  [get], [set]
```

Gets or sets the end date of the project.

```
00095         {
00096             set { endDate = value; }
00097             get { return endDate; }
00098         }
```

6.22.4.2 Id

```
int Object_Tier.Project.Id  [get], [set]
```

Gets or sets the project ID.

```
00069         {
00070             set { idProject = value; }
00071             get { return idProject; }
00072         }
```

6.22.4.3 StartDate

```
DateTime Object_Tier.Project.StartDate  [get]
```

Gets the start date of the project.

```
00087         {
00088             get { return startDate; }
00089         }
```

6.22.4.4 Status

```
Status Object_Tier.Project.Status  [get], [set]
```

Gets or sets the project status.

```
00078         {
00079             set { status = value; }
00080             get { return status; }
00081         }
```

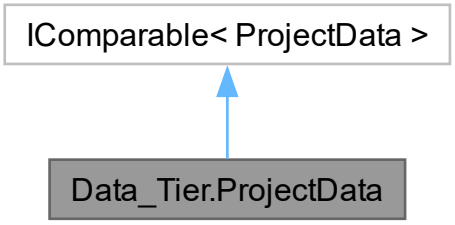
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Object Tier/Project.cs

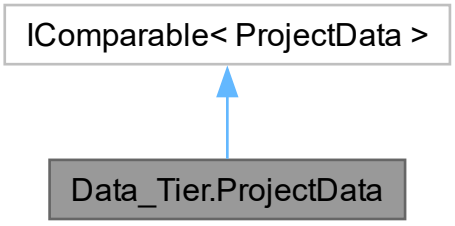
6.23 Data_Tier.ProjectData Class Reference

Class that represents project data, including operations related to clients, employees, and materials.

Inheritance diagram for Data_Tier.ProjectData:



Collaboration diagram for Data_Tier.ProjectData:



Public Member Functions

- ProjectData (Project project)

  Initializes a new instance of the ProjectData class with the specified project.
- int CompareTo (ProjectData project)

  Compares this ProjectData instance with another one based on their project IDs.
- bool AddClient (int clientId)

  Adds a client to the project.
- bool RemoveClient (int clientId)

  Removes a client from the project.
- bool AddEmployee (int employeeId)

  Adds an employee to the project.
- bool RemoveEmployee (int employeeId)

  Removes an employee from the project.
- bool UseMaterial (MaterialQuantity material)

  Uses material in the project.

Static Public Member Functions

- static ProjectData CreateProjectData (Project project)

  Factory method to create a new ProjectData instance for a given project.

Properties

- Project Project `[get, set]`

  Gets or sets the project associated with this data.

6.23.1 Detailed Description

Class that represents project data, including operations related to clients, employees, and materials.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 ProjectData()

```
Data_Tier.ProjectData.ProjectData (
            Project project)
```

Initializes a new instance of the ProjectData class with the specified project.

Parameters

project	The project to associate with this data.

```
00066         {
00067             Project = project;
00068             clientsService = new ClientsService();
00069             employeesService = new EmployeesService();
00070             materialService = new MaterialService();
00071         }
```

6.23.3 Member Function Documentation

6.23.3.1 AddClient()

```
bool Data_Tier.ProjectData.AddClient (
            int clientId)
```

Adds a client to the project.

Parameters

clientId	The ID of the client to add.

Returns

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

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while adding the client.

```
00104         {
00105             try
00106             {
00107                 return clientsService.AddClient(clientId);
00108             }
00109             catch (Exception ex)
00110             {
00111                 throw new ConfigurationErrorException("830" + ex);
00112             }
00113         }
```

6.23.3.2 AddEmployee()

```
bool Data_Tier.ProjectData.AddEmployee (
            int employeeId)
```

Adds an employee to the project.

Parameters

employeeId	The ID of the employee to add.

Returns

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

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while adding the employee.

```
00140         {
00141             try
00142             {
00143                 return employeesService.AddEmployee(employeeId);
00144             }
00145             catch (Exception ex)
00146             {
00147                 throw new ConfigurationErrorException("832" + ex);
00148             }
00149         }
```

6.23.3.3 CompareTo()

```
int Data_Tier.ProjectData.CompareTo (
            ProjectData project)
```

Compares this ProjectData instance with another one based on their project IDs.

Parameters

project	The other ProjectDatainstance to compare.

Returns

A value indicating the relative order of the projects.

```
00093         {
00094             return project.Project.Id.CompareTo(project.project.Id);
00095         }
```

6.23.3.4 CreateProjectData()

```
static ProjectData Data_Tier.ProjectData.CreateProjectData (
            Project project)  [static]
```

Factory method to create a new ProjectData instance for a given project.

Parameters

project	The project to create data for.

Returns

A new ProjectData instance.

```
00083         {
00084             return new ProjectData(project);
00085         }
```

6.23.3.5 RemoveClient()

```
bool Data_Tier.ProjectData.RemoveClient (
            int clientId)
```

Removes a client from the project.

Parameters

clientId	The ID of the client to remove.

Returns

True if the client was removed successfully; otherwise, false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while removing the client.

```
00122         {
00123             try
00124             {
00125                 return clientsService.RemoveClient(clientId);
00126             }
00127             catch (Exception ex)
00128             {
00129                 throw new ConfigurationErrorException("831" + ex);
00130             }
00131         }
```

6.23.3.6 RemoveEmployee()

```
bool Data_Tier.ProjectData.RemoveEmployee (
            int employeeId)
```

Removes an employee from the project.

Parameters

employeeId	The ID of the employee to remove.

Returns

True if the employee was removed successfully; otherwise, false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while removing the employee.

```
00158         {
00159             try
00160             {
00161                 return employeesService.RemoveEmployee(employeeId);
00162             }
00163             catch (Exception ex)
00164             {
00165                 throw new ConfigurationErrorException("833" + ex);
00166             }
00167         }
```

6.23.3.7 UseMaterial()

```
bool Data_Tier.ProjectData.UseMaterial (
            MaterialQuantity material)
```

Uses material in the project.

Parameters

material	The material and quantity to use.

Returns

True if the material was successfully used; otherwise, false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs while using the material.

```
00176         {
00177             try
00178             {
00179                 return materialService.AddMaterial(material);
00180             }
00181             catch (Exception ex)
00182             {
00183                 throw new ConfigurationErrorException("834" + ex.Message);
00184             }
00185         }
```

6.23.4 Property Documentation

6.23.4.1 Project

```
Project Data_Tier.ProjectData.Project  [get], [set]
```

Gets or sets the project associated with this data.

```
00053         {
00054             set { project = value; }
00055             get { return project; }
00056         }
```

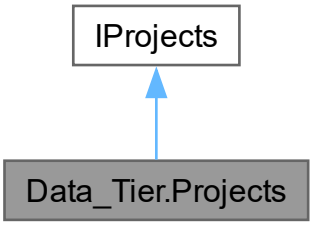
The documentation for this class was generated from the following file:

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/ProjectData.cs

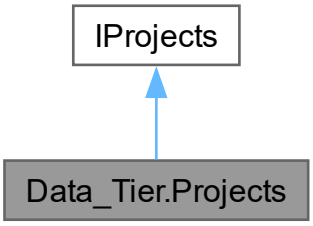
6.24 Data_Tier.Projects Class Reference

Singleton class that manages the projects in the system. Allows adding, removing, updating, and retrieving projects.

Inheritance diagram for Data_Tier.Projects:



Collaboration diagram for Data_Tier.Projects:



Public Member Functions

- int AddProject (ProjectData project)

  Adds a new project to the projects list.
- bool RemoveProject (int idProject)

  Removes a project from the projects list by its ID.
- bool ProjectExists (Project project2)

  Checks if a project exists by its ID.
- bool ProjectExists (int idProject)

  Checks if a project already exists based on its ID.
- bool UpdateStatus (int idProject, Status status)

  Updates the status of a project by its ID.
- bool CloseProject (int idProject)

  Closes a project by setting its end date and status to completed.
- bool AddClient (int idProject, int idClient)

  Adds a client to a project.
- bool RemoveClient (int idProject, int idClient)

  Removes a client from a project.
- bool AddEmployee (int idProject, int idEmployee)

  Adds an employee to a project.
- bool RemoveEmployee (int idProject, int idEmployee)

  Removes an employee from a project.
- bool UseMaterial (int idProject, int idMaterial, int quantity)

  Uses material in a project.

Public Member Functions inherited from Interface_Tier.IProjects

Protected Member Functions

- Projects ()

  Initializes a new instance of the Projects class with an empty list of projects.

Properties

- static Projects Instance `[get]`

  Gets the singleton instance of the Projects class.

6.24.1 Detailed Description

Singleton class that manages the projects in the system. Allows adding, removing, updating, and retrieving projects.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Projects()

```
Data_Tier.Projects.Projects ()  [protected]
```

Initializes a new instance of the Projects class with an empty list of projects.

```
00072         {
00073             projects = new List<ProjectData>(5);
00074         }
```

6.24.3 Member Function Documentation

6.24.3.1 AddClient()

```
bool Data_Tier.Projects.AddClient (
            int idProject,
            int idClient)
```

Adds a client to a project.

Parameters

idProject	The ID of the project to add the client to.
idClient	The ID of the client to add.

Returns

True if the client was added successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the addition.

Implements Interface_Tier.IProjects.

```
00279         {
00280             try
00281             {
00282                 ProjectData project = FindProject(idProject);
00283                 return project.AddClient(idClient);
00284             }
00285             catch (Exception ex)
00286             {
00287                 throw new ConfigurationErrorException("808" + ex);
00288             }
00289         }
```

6.24.3.2 AddEmployee()

```
bool Data_Tier.Projects.AddEmployee (
            int idProject,
            int idEmployee)
```

Adds an employee to a project.

Parameters

idProject	The ID of the project to add the employee to.
idEmployee	The ID of the employee to add.

Returns

True if the employee was added successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the addition.

Implements Interface_Tier.IProjects.

```
00327         {
00328             try
00329             {
00330                 ProjectData project = FindProject(idProject);
00331                 return project.AddEmployee(idEmployee);
00332             }
00333             catch (Exception ex)
00334             {
00335                 throw new ConfigurationErrorException("812" + ex);
00336             }
00337         }
```

6.24.3.3 AddProject()

```
int Data_Tier.Projects.AddProject (
            ProjectData project)
```

Adds a new project to the projects list.

Parameters

project	The project to add.

Returns

The ID of the added project.

Exceptions

ConfigurationErrorException	Throws an exception if the project is null or any error occurs during the addition.

```
00126         {
00127             if (project == null)
00128             {
00129                 throw new ConfigurationErrorException("800");
00130             }
00131 
00132             try
00133             {
00134                 projects.Add(project);
00135                 projects.Sort();
00136                 return project.Project.Id;
00137             }
00138             catch (Exception ex)
00139             {
00140                 throw new ConfigurationErrorException("801" + ex);
00141             }
00142         }
```

6.24.3.4 CloseProject()

```
bool Data_Tier.Projects.CloseProject (
            int idProject)
```

Closes a project by setting its end date and status to completed.

Parameters

idProject	The ID of the project to close.

Returns

True if the project was successfully closed, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the closure.

Implements Interface_Tier.IProjects.

```
00248         {
00249             try
00250             {
00251                 ProjectData project = FindProject(idProject);
00252 
00253                 if (project != null)
00254                 {
00255                     project.Project.EndDate = DateTime.Now;
00256                     project.Project.Status = Status.Completed;
00257                     return true;
00258                 }
00259             }
00260             catch (Exception ex)
00261             {
00262                 throw new ConfigurationErrorException("805" + ex);
00263             }
00264 
00265             return false;
00266         }
```

6.24.3.5 ProjectExists() [1/2]

```
bool Data_Tier.Projects.ProjectExists (
            int idProject)
```

Checks if a project already exists based on its ID.

Parameters

idProject	The ID of the project to check.

Returns

True if the project exists, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the check.

Implements Interface_Tier.IProjects.

```
00199         {
00200             try
00201             {
00202                 return ProjectExist(idProject);
00203             }
00204             catch (Exception ex)
00205             {
00206                 throw new ConfigurationErrorException("803" + ex);
00207             }
00208         }
```

6.24.3.6 ProjectExists() [2/2]

```
bool Data_Tier.Projects.ProjectExists (
            Project project2)
```

Checks if a project exists by its ID.

Parameters

projectId	The ID of the project to check.

Returns

True if the project exists, otherwise false.

```
00178         {
00179             foreach (ProjectData project1 in projects)
00180             {
00181                 if (project1.Project - project2)
00182                 {
00183                     return true;
00184                 }
00185             }
00186 
00187             return false;
00188         }
```

6.24.3.7 RemoveClient()

```
bool Data_Tier.Projects.RemoveClient (
            int idProject,
            int idClient)
```

Removes a client from a project.

Parameters

idProject	The ID of the project to remove the client from.
idClient	The ID of the client to remove.

Returns

True if the client was removed successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal.

Implements Interface_Tier.IProjects.

```
00301         {
00302             try
00303             {
00304                 ProjectData project = FindProject(idProject);
00305                 return project.RemoveClient(idClient);
00306             }
00307             catch (Exception ex)
00308             {
00309                 throw new ConfigurationErrorException("809" + ex);
00310             }
00311         }
```

6.24.3.8 RemoveEmployee()

```
bool Data_Tier.Projects.RemoveEmployee (
            int idProject,
            int idEmployee)
```

Removes an employee from a project.

Parameters

idProject	The ID of the project to remove the employee from.
idEmployee	The ID of the employee to remove.

Returns

True if the employee was removed successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal.

Implements Interface_Tier.IProjects.

```
00349         {
00350             try
00351             {
00352                 ProjectData project = FindProject(idProject);
00353                 return project.RemoveEmployee(idEmployee);
00354             }
00355             catch (Exception ex)
00356             {
00357                 throw new ConfigurationErrorException("813" + ex);
00358             }
00359         }
```

6.24.3.9 RemoveProject()

```
bool Data_Tier.Projects.RemoveProject (
            int idProject)
```

Removes a project from the projects list by its ID.

Parameters

idProject	The ID of the project to remove.

Returns

True if the project was successfully removed, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the removal.

Implements Interface_Tier.IProjects.

```
00153         {
00154             try
00155             {
00156                 ProjectData project = FindProject(idProject);
00157 
00158                 if (project != null)
00159                 {
00160                     projects.Remove(project);
00161                     return true;
00162                 }
00163             }
00164             catch (Exception ex)
00165             {
00166                 throw new ConfigurationErrorException("802" + ex);
00167             }
00168 
00169             return false;
00170         }
```

6.24.3.10 UpdateStatus()

```
bool Data_Tier.Projects.UpdateStatus (
            int idProject,
            Status status)
```

Updates the status of a project by its ID.

Parameters

idProject	The ID of the project to update.
status	The new status of the project.

Returns

True if the status was updated successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the update.

```
00220         {
00221             try
00222             {
00223                 ProjectData project = FindProject(idProject);
00224 
00225                 if (project != null)
00226                 {
00227                     project.Project.Status = status;
00228                     return true;
00229                 }
00230             }
00231             catch (Exception ex)
00232             {
00233                 throw new ConfigurationErrorException("804" + ex);
00234             }
00235 
00236             return false;
00237         }
```

6.24.3.11 UseMaterial()

```
bool Data_Tier.Projects.UseMaterial (
            int idProject,
            int idMaterial,
            int quantity)
```

Uses material in a project.

Parameters

idProject	The ID of the project to use material in.
idMaterial	The ID of the material to use.
quantity	The quantity of the material to use.

Returns

True if the material was used successfully, otherwise false.

Exceptions

ConfigurationErrorException	Throws an exception if an error occurs during the material use.

Implements Interface_Tier.IProjects.

```
00375         {
00376             try
00377             {
00378                 ProjectData project = FindProject(idProject);
00379                 return project.UseMaterial(MaterialQuantity.CreateMaterialQuantity(idMaterial,
      quantity));
00380             }
00381             catch (Exception ex)
00382             {
00383                 throw new ConfigurationErrorException("805" + ex);
00384             }
00385         }
```

6.24.4 Property Documentation

6.24.4.1 Instance

```
Projects Data_Tier.Projects.Instance  [static], [get]
```

Gets the singleton instance of the Projects class.

```
00044         {
00045             get
00046             {
00047                 if (instance == null)
00048                 {
00049                     instance = new Projects();
00050                 }
00051
00052                 return instance;
00053             }
00054         }
```

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

- C:/Users/hugoc/Desktop/2024_2025/Programação Orientada a Objetos/trabalhoPOO_23010_Fase2/src/↩ Data Tier/Projects.cs

Index