CRUDWebApp

Generated by Doxygen 1.13.2

Chapter 1

CRUDWebApp

1.1 Overview

The **CRUDWebApp** is composed of 2 projects:

- `CRUDService`: based on ASP.NET Core which serves as backend application
- `FrontEnd`: based on Blazor WebAssembly (WASM) which serves as SPA frontend application.

For detailed information open refman.pdf.

1.2 Caution

For the whole app to work properly as is, you need **Postgres Server** running on **localhost** on port **5432**.

If your configuration **differs**, edit connection specifics in **appsettings.json** in **root directory of CRUDService** to match yours.

1.3 Used Libraries

1.3.1 Backend

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Metadata.Builders
- Microsoft.AspNetCore.Mvc
- Microsoft.AspNetCore.Authorization
- Microsoft.EntityFrameworkCore
- System.ComponentModel.DataAnnotations
- System.Security.Cryptography

- Microsoft.EntityFrameworkCore.Migrations
- Npgsql.EntityFrameworkCore.PostgreSQL.Metadata
- Microsoft.EntityFrameworkCore.Storage.ValueConversion
- Microsoft.EntityFrameworkCore.Infrastructure
- Microsoft.IdentityModel.Tokens
- System.IdentityModel.Tokens.Jwt
- System.Security.Claims

1.3.2 Frontend

- System.ComponentModel.DataAnnotations
- System.Text.Json
- System.Net.Http.Headers
- System.Net.Http.Json
- Microsoft.AspNetCore.Components
- System.IdentityModel.Tokens.Jwt
- System.Security.Claims
- Blazored.LocalStorage
- Microsoft.AspNetCore.Components.Forms
- Microsoft.AspNetCore.Components.Routing
- Microsoft.AspNetCore.Components.Web
- Microsoft.AspNetCore.Components.Web.Virtualization
- Microsoft.AspNetCore.Components.WebAssembly.Http
- Microsoft.JSInterop
- Microsoft.AspNetCore.Components.WebAssembly.Hosting

1.4 How to compile

- Download code from github: `https://github.com/sigmor10/CRUDWebApp`
- Open Console / PowerShell in CRUDService project's root directory
- Run the "dotnet ef database update" command to run migrations, or you can load sql dump
- Open the main folder and run .sln file in Visual Studio 2022
- Run both projects CRUDService and FrontEnd in any order
  - Note 1: If FrontEnd is run before CRUDService, requests to backend will fail.
  - Note 2: On first run of CRUDService you will be prompted with whether to trust ASP.NET self signed Certificate. Agree to it so that the app will run with REST API communication over https

Chapter 2

Namespace Index

2.1 Namespace List

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

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 CRUDService Namespace Reference

5.2 CRUDService.Contracts Namespace Reference

Classes

- class CategoryConfiguration

  Class needed for configuring specific rules of its table in the db.
- class ContactConfiguration

  Class needed for configuring specific rules of Contacts table in the database.
- class SubCategoryConfiguration

  Class needed for configuring specific rules of its table in the db.

5.3 CRUDService.Controllers Namespace Reference

Classes

- class AuthController

  Controller class which handles http(s) communication, through the use of REST API. It only handles requests/responses, actual operations are relayed to service layer.
- class ContactController

  Controller class which handles http(s) communication, through the use of REST API. It only handles requests/responses, actual operations are relayed to service layer.

5.4 CRUDService.Data Namespace Reference

Classes

- class AppDbContext

  Class used to manage the connection to the database.
- class **Initializer**

  Populates empty database tables.

5.5 CRUDService.DTOs Namespace Reference

Classes

- class BaseContactDTO

  Base DTO holds every field that is used in every other contact DTO class Annotated for early valdiation and early rejection of invalid requests.
- class CreateContactRequest

  DTO class used for updates and creation requests, it posses every field informative field of contact. Annotated for early valdiation and early rejection of invalid requests.
- class DetailedContactDTO

  More detailed DTO used for adding missing fields from BASEContactDTO. It's used to minimize code duplication. Annotated for early valdiation and early rejection of invalid requests.
- class GetContactResponse

  Detailed DTO class used to transfer data needed for detailed view of contact info. Caution: Password field is purposefully omitted due to security reasons. Annotated for early valdiation and early rejection of invalid requests.
- class UpdateContactRequest

  DTO class for use when recieving Contact update requests. Annotated for early valdiation and early rejection of invalid requests.

5.6 CRUDService.Helpers Namespace Reference

Classes

- class **ContactMapper**

  Class contains methods used to transform Contact entity to and from DTO objects.
- class **HelperMethods**

  Collection of helper methods.

5.7 CRUDService.Migrations Namespace Reference

Classes

- class AppDbContextModelSnapshot
- class InitialCreate

5.8 CRUDService.Models Namespace Reference

Classes

- class Category

  Entity class for Dcitionary table Categories.
- class Contact

  Entity class for CRUD entity Contact For database table constraint details see Contracts/ContactConfiguration.cs file.
- class SubCategory

  Entity class for Dictionary entity SubCategory it is used only for dictionary entries For SubCategory that are limited to bussiness Category For database table constraint details see Contracts/SubCategoryConfiguration.cs file.
- class User

  Class representing a user.

5.9 CRUDService.Repository Namespace Reference

Classes

- class ContactRepository

  Implements repository logic related to access to the database.
- interface IContactRepository

  Interface posseses collection of methods for handling data stored in the database.
- interface IUserRepository

  Interface posseses collection of methods for handling data stored in the database.
- class UserRepository

  Implementatiion of IUserRepository interface.

5.10 CRUDService.Service Namespace Reference

Classes

- class ContactService

  Implements bussiness logic and mediates between contrler and repository layers.
- interface IContactService

  Defines bussiness logic operations for Contact management.
- interface IUserService

  Defines bussiness logic in regards authentication of users.
- class UserService

  Class that implements IUserService interface.

5.11 CRUDService.Validators Namespace Reference

Classes

- class PasswordValidator

  Custom password validator for use in automatic password validation through annotations.

5.12 FrontEnd Namespace Reference

5.13 FrontEnd.Models Namespace Reference

Classes

- class Category

  Entity class for Dictionary entity Category For database table constraint details see Contracts/Category←↩ Configuration.cs file Annotated for early valdiation and early rejection of invalid requests.
- class Contact

  Base DTO holds every field that is used in every other contact DTO class Annotated for early valdiation and early rejection of invalid requests.

- class CreateContactRequest

  dTO class used for updates and creation requests, it posses every field informative field of contact Annotated for early valdiation and early rejection of invalid requests.
- class DetailedContact

  More detailed DTO used for adding missing fields from BASEContactDTO. It's used to minimize code duplication. Annotated for early valdiation and early rejection of invalid requests.
- class GetContactResponse

  Detailed DTO class used to transfer data needed for detailed view of contact info. Caution: Password field is purposefully omitted due to security reasons. Annotated for early valdiation and early rejection of invalid requests.
- class SubCategory

  Entity class for Dictionary entity SubCategory it is used only for dictionary entries For SubCategory that are limited to bussiness Category For database table constraint details see Contracts/SubCategoryConfiguration.cs file Annotated for early valdiation and early rejection of invalid requests.
- class UpdateContactRequest

  DTO class for use when recieving Contact update requests Annotated for early valdiation and early rejection of invalid requests.
- class User

  Class representing a user. Annotated for early valdiation and early rejection of invalid requests.

5.14 FrontEnd.Pages Namespace Reference

Classes

- class ContactAddView

  Code behind of ContactAddView.
- class ContactDetails
- class ContactEditView
- class ContactList
- class Login

5.15 FrontEnd.Services Namespace Reference

Classes

- class AuthService

  Authentication and jwt token handling class.

5.16 FrontEnd.Validators Namespace Reference

Classes

- class PasswordValidator

  Custom password validator for use in automatic password validation through annotations.

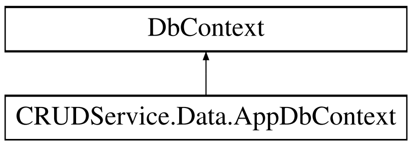
Chapter 6

Class Documentation

6.1 CRUDService.Data.AppDbContext Class Reference

Class used to manage the connection to the database.

Inheritance diagram for CRUDService.Data.AppDbContext:



Public Member Functions

- **AppDbContext** (DbContextOptions< AppDbContext > options)

Protected Member Functions

- override void OnModelCreating (ModelBuilder modelBuilder)
  Applies custom rules to properties of columns that are based on class fields.

Properties

- DbSet< Contact > **Contacts** `[get, set]`
- DbSet< Category > **Categories** `[get, set]`
- DbSet< SubCategory > **SubCategories** `[get, set]`

6.1.1 Detailed Description

Class used to manage the connection to the database.

6.1.2 Member Function Documentation

6.1.2.1 OnModelCreating()

```
override void CRUDService.Data.AppDbContext.OnModelCreating (
             ModelBuilder modelBuilder)  [inline], [protected]
```

Applies custom rules to properties of columns that are based on class fields.

Parameters

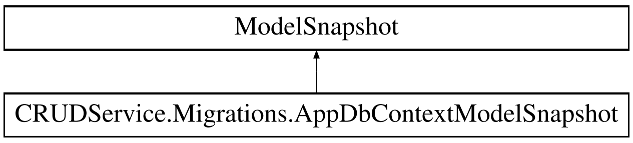
modelBuilder	

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

- CRUDService/Data/AppDbContext.cs

6.2 CRUDService.Migrations.AppDbContextModelSnapshot Class Reference

Inheritance diagram for CRUDService.Migrations.AppDbContextModelSnapshot:



Protected Member Functions

- override void **BuildModel** (ModelBuilder modelBuilder)

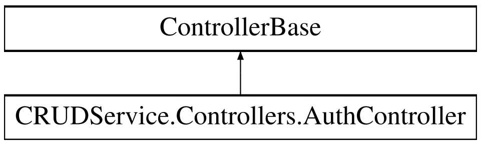
The documentation for this class was generated from the following file:

- CRUDService/Migrations/AppDbContextModelSnapshot.cs

6.3 CRUDService.Controllers.AuthController Class Reference

Controller class which handles http(s) communication, through the use of REST API. It only handles requests/responses, actual operations are relayed to service layer.

Inheritance diagram for CRUDService.Controllers.AuthController:



Public Member Functions

- AuthController (ILogger< ContactController > logger, IUserService service)

  Constructor with dependency injections.
- async Task< IActionResult > Authorize ([FromBody] User user)

  Returns paginated list of Contacts.

Private Attributes

- readonly IUserService **_userService**

  Propery used to access User Service.
- readonly ILogger< ContactController > **_logger**

6.3.1 Detailed Description

Controller class which handles http(s) communication, through the use of REST API. It only handles requests/responses, actual operations are relayed to service layer.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AuthController()

```
CRUDService.Controllers.AuthController.AuthController (
            ILogger< ContactController > logger,
            IUserService service) [inline]
```

Constructor with dependency injections.

Parameters

logger	
service	

6.3.3 Member Function Documentation

6.3.3.1 Authorize()

```
async Task< IActionResult > CRUDService.Controllers.AuthController.Authorize (
            [FromBody] User user) [inline]
```

Returns paginated list of Contacts.

Parameters

user	User object passed for use of in service layer

Returns

Jwt Token or 401 Unauthorized message

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

- CRUDService/Controllers/AuthController.cs

6.4 FrontEnd.Services.AuthService Class Reference

Authentication and jwt token handling class.

Public Member Functions

- **AuthService** (ILocalStorageService localStorage)
- async Task InitializeAsync ()

    Initializes token and authorization mechanics.
- async Task SetToken (string token)

    Puts token in local storage.
- async Task Logout ()

    Logs user out and deletes token.

Properties

- bool **IsLoggedIn** `[get, private set]`
- string? **UserEmail** `[get, private set]`

Events

- Action? **OnChange**

Private Member Functions

- void **NotifyStateChanged** ()

Private Attributes

- readonly ILocalStorageService **_localStorage**

6.4.1 Detailed Description

Authentication and jwt token handling class.

6.4.2 Member Function Documentation

6.4.2.1 InitializeAsync()

```
async Task FrontEnd.Services.AuthService.InitializeAsync ()  [inline]
```

Initializes token and authorization mechanics.

Returns

6.4.2.2 Logout()

```
async Task FrontEnd.Services.AuthService.Logout ()  [inline]
```

Logs user out and deletes token.

Returns

6.4.2.3 SetToken()

```
async Task FrontEnd.Services.AuthService.SetToken (
             string token)  [inline]
```

Puts token in local storage.

Parameters

token	Jwt token

Returns

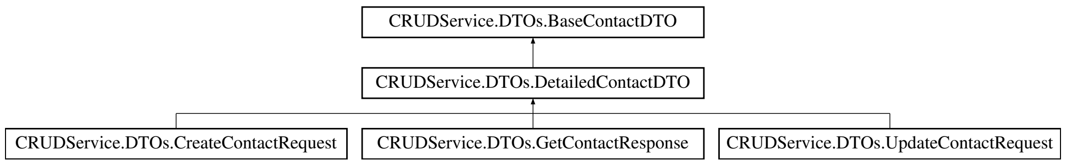
The documentation for this class was generated from the following file:

- FrontEnd/Services/AuthService.cs

6.5 CRUDService.DTOs.BaseContactDTO Class Reference

Base DTO holds every field that is used in every other contact DTO class Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for CRUDService.DTOs.BaseContactDTO:


CRUDService.DTOs.BaseContactDTO
CRUDService.DTOs.DetailedContactDTO
CRUDService.DTOs.CreateContactRequest
CRUDService.DTOs.GetContactResponse
CRUDService.DTOs.UpdateContactRequest

Properties

- Guid **Id** `[get, set]`
- required string **Name** `[get, set]`
- required string **Surname** `[get, set]`
- required string **Email** `[get, set]`

6.5.1 Detailed Description

Base DTO holds every field that is used in every other contact DTO class Annotated for early valdiation and early rejection of invalid requests.

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

- CRUDService/DTOs/BaseContactDTO.cs

6.6 CRUDService.Models.Category Class Reference

Entity class for Dcitionary table Categories.

Properties

- required int **Id** `[get, set]`
- required string **Name** `[get, set]`

6.6.1 Detailed Description

Entity class for Dcitionary table Categories.

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

- CRUDService/Models/Category.cs

6.7 FrontEnd.Models.Category Class Reference

Entity class for Dictionary entity Category For database table constraint details see Contracts/Category↩ Configuration.cs file Annotated for early valdiation and early rejection of invalid requests.

Properties

- int **Id** `[get, set]`
- required string **Name** `[get, set]`

6.7.1 Detailed Description

Entity class for Dictionary entity Category For database table constraint details see Contracts/Category↩ Configuration.cs file Annotated for early valdiation and early rejection of invalid requests.

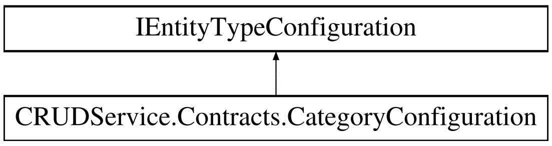
The documentation for this class was generated from the following file:

- FrontEnd/Models/Category.cs

6.8 CRUDService.Contracts.CategoryConfiguration Class Reference

Class needed for configuring specific rules of its table in the db.

Inheritance diagram for CRUDService.Contracts.CategoryConfiguration:



Public Member Functions

- void Configure (EntityTypeBuilder< Category > builder)

  Sets configuration of the Categories table.

6.8.1 Detailed Description

Class needed for configuring specific rules of its table in the db.

6.8.2 Member Function Documentation

6.8.2.1 Configure()

```
void CRUDService.Contracts.CategoryConfiguration.Configure (
            EntityTypeBuilder< Category > builder)  [inline]
```

Sets configuration of the Categories table.

Parameters

builder	

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

- CRUDService/Contracts/CategoryConfiguration.cs

6.9 CRUDService.Models.Contact Class Reference

Entity class for CRUD entity Contact For database table constraint details see Contracts/ContactConfiguration.cs file.

Properties

- Guid **Id** = Guid.NewGuid() `[get, set]`
- required string **Name** `[get, set]`
- required string **Surname** `[get, set]`
- required string **Email** `[get, set]`
- string? **PasswordHash** `[get, set]`
- required int **CategoryId** `[get, set]`
- string? **SubCategory** `[get, set]`
- required string **Phone** `[get, set]`
- required DateOnly **BirthDate** `[get, set]`

6.9.1 Detailed Description

Entity class for CRUD entity Contact For database table constraint details see Contracts/ContactConfiguration.cs file.

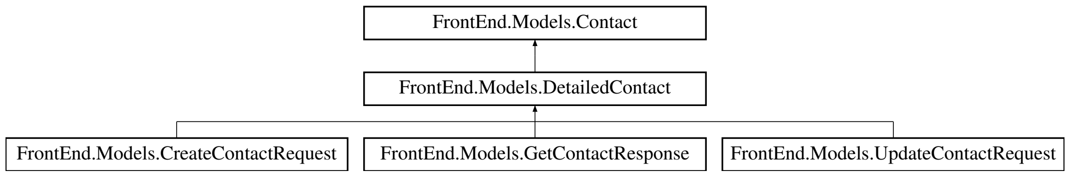
The documentation for this class was generated from the following file:

- CRUDService/Models/Contact.cs

6.10 FrontEnd.Models.Contact Class Reference

Base DTO holds every field that is used in every other contact DTO class Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for FrontEnd.Models.Contact:


FrontEnd.Models.Contact
FrontEnd.Models.DetailedContact
FrontEnd.Models.CreateContactRequest
FrontEnd.Models.GetContactResponse
FrontEnd.Models.UpdateContactRequest

Properties

- Guid **Id** `[get, set]`
- string? **Name** `[get, set]`
- string? **Surname** `[get, set]`
- string? **Email** `[get, set]`

6.10.1 Detailed Description

Base DTO holds every field that is used in every other contact DTO class Annotated for early valdiation and early rejection of invalid requests.

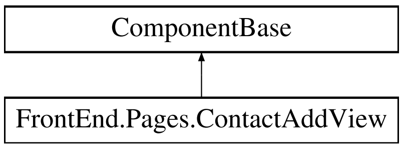
The documentation for this class was generated from the following file:

- FrontEnd/Models/Contact.cs

6.11 FrontEnd.Pages.ContactAddView Class Reference

Code behind of ContactAddView.

Inheritance diagram for FrontEnd.Pages.ContactAddView:



Protected Member Functions

- override async Task OnInitializedAsync ()

  Redirects unauthorized users to the login page.
- override async Task OnParametersSetAsync ()

  Initializes contact list.

Properties

- string **selectedSubCategory** `[get, set]`
- int **selectedCategory** `[get, set]`

Private Member Functions

- void **SetToDefaultValues** ()

  Sets Entity and form to default values.
- async Task LoadContent ()

  Fetches paginated list from backend.
- void **FilterSubCategories** ()

  Filters SubCategories to select only one viable for given category.
- async Task< bool > ValidateEmael ()

  Checks if given email is already in use by other user.
- async Task SaveContact ()

  Saves edited contact.
- void **RedirectToList** ()

  Redirects to contact list view.

Private Attributes

- CreateContactRequest? **contact** = new()
- List< Category >? **categories**
- List< SubCategory >? **subCats**
- List< SubCategory > **filteredSubCats** = new()
- bool **emailExists** = false
- bool **IsCustomSubCategory** = false
- string **_selectedSubCategory**
- int **_selectedCategory**

6.11.1 Detailed Description

Code behind of ContactAddView.

6.11.2 Member Function Documentation

6.11.2.1 LoadContent()

```
async Task FrontEnd.Pages.ContactAddView.LoadContent ()  [inline], [private]
```

Fetches paginated list from backend.

Returns

6.11.2.2 OnInitializedAsync()

```
override async Task FrontEnd.Pages.ContactAddView.OnInitializedAsync ()  [inline], [protected]
```

Redirects unauthorized users to the login page.

Returns

6.11.2.3 OnParametersSetAsync()

```
override async Task FrontEnd.Pages.ContactAddView.OnParametersSetAsync ()  [inline], [protected]
```

Initializes contact list.

Returns

6.11.2.4 SaveContact()

```
async Task FrontEnd.Pages.ContactAddView.SaveContact ()  [inline], [private]
```

Saves edited contact.

Returns

6.11.2.5 ValidateEmael()

```
async Task< bool > FrontEnd.Pages.ContactAddView.ValidateEmael ()  [inline], [private]
```

Checks if given email is already in use by other user.

Returns

bool saying whether email is valid and not taken

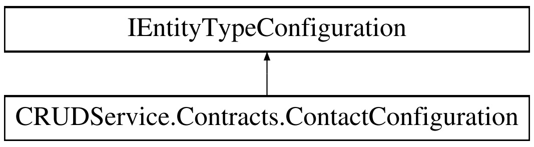
The documentation for this class was generated from the following file:

- FrontEnd/Pages/ContactAddView.razor.cs

6.12 CRUDService.Contracts.ContactConfiguration Class Reference

Class needed for configuring specific rules of Contacts table in the database.

Inheritance diagram for CRUDService.Contracts.ContactConfiguration:



Public Member Functions

- void Configure (EntityTypeBuilder< Contact > builder)
  Sets configuration of the Contacts table.

6.12.1 Detailed Description

Class needed for configuring specific rules of Contacts table in the database.

6.12.2 Member Function Documentation

6.12.2.1 Configure()

```
void CRUDService.Contracts.ContactConfiguration.Configure (
             EntityTypeBuilder< Contact > builder)  [inline]
```

Sets configuration of the Contacts table.

Parameters

builder	

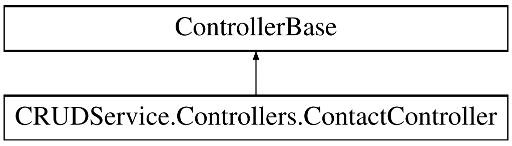
The documentation for this class was generated from the following file:

- CRUDService/Contracts/ContactConfiguration.cs

6.13 CRUDService.Controllers.ContactController Class Reference

Controller class which handles http(s) communication, through the use of REST API. It only handles requests/responses, actual operations are relayed to service layer.

Inheritance diagram for CRUDService.Controllers.ContactController:



Public Member Functions

- ContactController (ILogger< ContactController > logger, IContactService service)

  Constructor with dependency injections.
- async Task< IActionResult > GetContacts (int skip, int take)

  Returns paginated list of Contacts.
- async Task< IActionResult > GetContact (Guid id)

  Returns a detailed Contact with given id.
- async Task< IActionResult > GetCategories ()

  Returns list of all categories.
- async Task< IActionResult > GetSubCategories (int id)

  Returns list of anmes of all subcategories for given category id.
- async Task< IActionResult > GetSubAllCategories ()

  Returns list of anmes of all subcategories.
- async Task< IActionResult > CheckIfEmailTaken ([FromQuery] string email)

  Returns information if email is taken or not.
- async Task< IActionResult > CreateContact ([FromBody] CreateContactRequest request)

  Creates new Contact instance.
- async Task< IActionResult > UpdateContact (Guid id, [FromBody] UpdateContactRequest request)

  Updates existing Contact instance.
- async Task< IActionResult > DeleteContact (Guid id)

  Deletes existing Contact.

Private Attributes

- readonly IContactService **_contactService**

  Propery used to access Contact Service.
- readonly ILogger< ContactController > **_logger**

6.13.1 Detailed Description

Controller class which handles http(s) communication, through the use of REST API. It only handles requests/responses, actual operations are relayed to service layer.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 ContactController()

```
CRUDService.Controllers.ContactController.ContactController (
            ILogger< ContactController > logger,
            IContactService service) [inline]
```

Constructor with dependency injections.

Parameters

logger	
service	

6.13.3 Member Function Documentation

6.13.3.1 CheckIfEmailTaken()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.CheckIfEmailTaken (
            [FromQuery] string email) [inline]
```

Returns information if email is taken or not.

Parameters

email	Email that is to be checked

Returns

bool saying whether given email already exists in database or not

6.13.3.2 CreateContact()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.CreateContact (
            [FromBody] CreateContactRequest request) [inline]
```

Creates new Contact instance.

Parameters

request	CreateContactRequest object with data needed to create neew Contact

Returns

201 code or 400, depending on whether entry was created or not

6.13.3.3 DeleteContact()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.DeleteContact (
            Guid id)  [inline]
```

Deletes existing Contact.

Parameters

id	Contact's Guid

Returns

404 code if Contact with given id does not exist 200 code if Contact was sucessfully deleted

6.13.3.4 GetCategories()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.GetCategories ()  [inline]
```

Returns list of all categories.

Returns

List of all categories in the database

6.13.3.5 GetContact()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.GetContact (
             Guid id)  [inline]
```

Returns a detailed Contact with given id.

Parameters

id	Guid of the requested contact

Returns

404 or requested Contact transformed into GetContactResponse Object

6.13.3.6 GetContacts()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.GetContacts (
             int skip,
             int take)  [inline]
```

Returns paginated list of Contacts.

Parameters

skip	Defines how many records should be skipped
take	Defines max length of returned list

Returns

Returns list Of BaseContactDTO objects, can be empty

6.13.3.7 GetSubAllCategories()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.GetSubAllCategories ()
[inline]
```

Returns list of anmes of all subcategories.

Returns

list of all subcategoires in the database

6.13.3.8 GetSubCategories()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.GetSubCategories (
            int id) [inline]
```

Returns list of anmes of all subcategories for given category id.

Parameters

id	Id of the category, requested subcategories belongg to.

Returns

List of all subcategory names that belong to Category with given id

6.13.3.9 UpdateContact()

```
async Task< IActionResult > CRUDService.Controllers.ContactController.UpdateContact (
            Guid id,
            [FromBody] UpdateContactRequest request) [inline]
```

Updates existing Contact instance.

Parameters

id	Contact's Guid
request	UpdateContactRequest object containing new/updated data for existing Contact

Returns

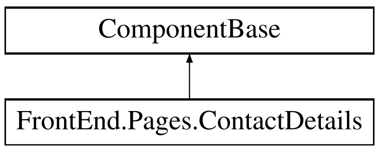
following http codes: 404 if no contact exists with given id, 204 if update was successful 400 if data in UpdateContactRequest was invalid

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

- CRUDService/Controllers/ContactController.cs

6.14 FrontEnd.Pages.ContactDetails Class Reference

Inheritance diagram for FrontEnd.Pages.ContactDetails:



Protected Member Functions

- override async Task OnParametersSetAsync ()

  Initializes contact list.

Properties

- Guid **id** `[get, set]`

Private Member Functions

- async Task LoadContactList ()

  Fetches paginated list from backend.
- async Task DeleteContact (Guid id)

  Sends delete request.
- void RedirectToEdit (Guid id)

  Redirects to Contact's edit view.
- void **RedirectToList** ()

  Redirects to contact list view.

Private Attributes

- GetContactResponse? **contact**

6.14.1 Member Function Documentation

6.14.1.1 DeleteContact()

```
async Task FrontEnd.Pages.ContactDetails.DeleteContact (
             Guid id)  [inline], [private]
```

Sends delete request.

Parameters

id	Contact's Guid

Returns

6.14.1.2 LoadContactList()

```
async Task FrontEnd.Pages.ContactDetails.LoadContactList ()  [inline], [private]
```

Fetches paginated list from backend.

Returns

6.14.1.3 OnParametersSetAsync()

```
override async Task FrontEnd.Pages.ContactDetails.OnParametersSetAsync ()  [inline], [protected]
```

Initializes contact list.

Returns

6.14.1.4 RedirectToEdit()

```
void FrontEnd.Pages.ContactDetails.RedirectToEdit (
             Guid id)  [inline], [private]
```

Redirects to Contact's edit view.

Parameters

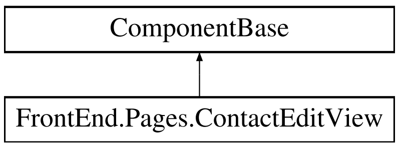
id	Contact's Guid

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

- FrontEnd/Pages/ContactDetails.razor.cs

6.15 FrontEnd.Pages.ContactEditView Class Reference

Inheritance diagram for FrontEnd.Pages.ContactEditView:



Protected Member Functions

- override async Task OnInitializedAsync ()

    Redirects unauthorized users to the login page.
- override async Task OnParametersSetAsync ()

    Initializes contact list.

Properties

- Guid **id** `[get, set]`
- string **selectedSubCategory** `[get, set]`
- int **selectedCategory** `[get, set]`

Private Member Functions

- void **SetToDefaultValues** ()

    Sets Entity and form to default values.
- async Task LoadContent ()

    Fetches paginated list from backend.
- void **FilterSubCategories** ()

    Filters SubCategories to select only one viable for given category.
- async Task< bool > ValidateEmael ()

    Checks if given email is already in use by other user.
- async Task SaveContact ()

    Saves edited contact.
- void RedirectToDetails (Guid id)

    Redirects to Contact's edit view.

Private Attributes

- GetContactResponse? **ogContact**
- UpdateContactRequest? **contact**
- List< Category >? **categories**
- List< SubCategory >? **subCats**
- List< SubCategory > **filteredSubCats** = new()
- bool **emailExists** = false
- bool **IsCustomSubCategory** = false
- string **_selectedSubCategory**
- int **_selectedCategory**

6.15.1 Member Function Documentation

6.15.1.1 LoadContent()

```
async Task FrontEnd.Pages.ContactEditView.LoadContent ()  [inline], [private]
```

Fetches paginated list from backend.

Returns

6.15.1.2 OnInitializedAsync()

```
override async Task FrontEnd.Pages.ContactEditView.OnInitializedAsync () [inline], [protected]
```

Redirects unauthorized users to the login page.

Returns

6.15.1.3 OnParametersSetAsync()

```
override async Task FrontEnd.Pages.ContactEditView.OnParametersSetAsync () [inline], [protected]
```

Initializes contact list.

Returns

6.15.1.4 RedirectToDetails()

```
void FrontEnd.Pages.ContactEditView.RedirectToDetails (
            Guid id) [inline], [private]
```

Redirects to Contact's edit view.

Parameters

id	Contact Guid

6.15.1.5 SaveContact()

```
async Task FrontEnd.Pages.ContactEditView.SaveContact () [inline], [private]
```

Saves edited contact.

Returns

6.15.1.6 ValidateEmael()

```
async Task< bool > FrontEnd.Pages.ContactEditView.ValidateEmael () [inline], [private]
```

Checks if given email is already in use by other user.

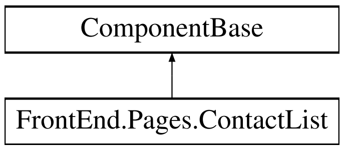
Returns

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

- FrontEnd/Pages/ContactEditView.razor.cs

6.16 FrontEnd.Pages.ContactList Class Reference

Inheritance diagram for FrontEnd.Pages.ContactList:



Protected Member Functions

- override async Task OnInitializedAsync ()

  Initializes contact list.

Private Member Functions

- async Task LoadContactList ()

  Fetches paginated list from backend.
- async Task IncrementSkip ()

  Next list page.
- async Task DecrementSkip ()

  Previous list page.
- async Task DeleteContact (Guid id)

  Sends delete request.
- void RedirectToEdit (Guid id)

  Redirects to Contact's edit view.
- void RedirectToDetails (Guid id)

  Redirects to Contact's details view.
- void **RedirectToAdd** ()

  Redirects to add contact view.

Private Attributes

- List< Contact >? **contacts**
- int **skip** = 0
- int **take** = 7

6.16.1 Member Function Documentation

6.16.1.1 DecrementSkip()

```
async Task FrontEnd.Pages.ContactList.DecrementSkip ()  [inline], [private]
```

Previous list page.

Returns

6.16.1.2 DeleteContact()

```
async Task FrontEnd.Pages.ContactList.DeleteContact (
            Guid id)  [inline], [private]
```

Sends delete request.

Parameters

id	Contact's Guid

Returns

6.16.1.3 IncrementSkip()

```
async Task FrontEnd.Pages.ContactList.IncrementSkip ()  [inline], [private]
```

Next list page.

Returns

6.16.1.4 LoadContactList()

```
async Task FrontEnd.Pages.ContactList.LoadContactList ()  [inline], [private]
```

Fetches paginated list from backend.

Returns

6.16.1.5 OnInitializedAsync()

```
override async Task FrontEnd.Pages.ContactList.OnInitializedAsync ()  [inline], [protected]
```

Initializes contact list.

Returns

6.16.1.6 RedirectToDetails()

```
void FrontEnd.Pages.ContactList.RedirectToDetails (
             Guid id)  [inline], [private]
```

Redirects to Contact's details view.

Parameters

id	Contact's Guid

6.16.1.7 RedirectToEdit()

```
void FrontEnd.Pages.ContactList.RedirectToEdit (
             Guid id)  [inline], [private]
```

Redirects to Contact's edit view.

Parameters

id	Contact's Guid

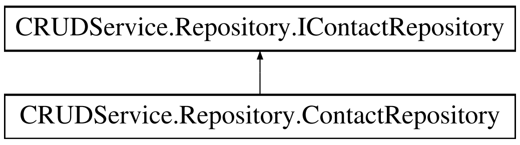
The documentation for this class was generated from the following file:

- FrontEnd/Pages/ContactList.razor.cs

6.17 CRUDService.Repository.ContactRepository Class Reference

Implements repository logic related to access to the database.

Inheritance diagram for CRUDService.Repository.ContactRepository:


CRUDService.Repository.IContactRepository
CRUDService.Repository.ContactRepository

Public Member Functions

- ContactRepository (AppDbContext context)

    Constructor.
- async Task< bool > CheckIfIdExists (Guid id)

    Checks if contact with given id exists.
- async Task< Contact?> FindContactById (Guid id)

    Retrieves contact by its id.
- async Task< List< Contact > > FindAllContacts (int skip, int take)

    Retrieves a paginated list of contacts.
- async Task< List< Category > > FindAllCategories ()

    Retrieves all categories.
- async Task< string?> FindCategoryNameById (int id)

    Retrieves a category name for given id.
- async Task< string?> FindUserEmailById (Guid id)

    Retrieves user email by her/his id.

- async Task< List< string > > FindAllSubCategoriesById (int id)

  Fetches all subcategoires for given id.
- async Task AddContact (Contact contact)

  Adds new contact.
- async Task< bool > UpdateContact (Contact contact)

  Updates existing contact.
- async Task< bool > DeleteContact (Guid id)

  Deletes existing contact.
- Task< bool > CheckIfEmailTaken (string email)

  Retrieves contact by its Email.
- async Task< List< SubCategory > > FindAllSubCategories ()

  Fetches all subcategoires.

Private Attributes

- readonly AppDbContext **_context**

6.17.1 Detailed Description

Implements repository logic related to access to the database.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 ContactRepository()

```
CRUDService.Repository.ContactRepository.ContactRepository (
            AppDbContext context)  [inline]
```

Constructor.

Parameters

context	

6.17.3 Member Function Documentation

6.17.3.1 AddContact()

```
async Task CRUDService.Repository.ContactRepository.AddContact (
            Contact contact)  [inline]
```

Adds new contact.

Parameters

contact	Contact object

Returns

Implements CRUDService.Repository.IContactRepository.

6.17.3.2 CheckIfEmailTaken()

```
Task< bool > CRUDService.Repository.ContactRepository.CheckIfEmailTaken (
             string email)  [inline]
```

Retrieves contact by its Email.

Parameters

email	

Returns

true if given email exists or flase if not

Implements CRUDService.Repository.IContactRepository.

6.17.3.3 CheckIfIdExists()

```
async Task< bool > CRUDService.Repository.ContactRepository.CheckIfIdExists (
             Guid id)  [inline]
```

Checks if contact with given id exists.

Parameters

id	Contacts Guid

Returns

true if exists false if not

Implements CRUDService.Repository.IContactRepository.

6.17.3.4 DeleteContact()

```
async Task< bool > CRUDService.Repository.ContactRepository.DeleteContact (
             Guid id)  [inline]
```

Deletes existing contact.

Parameters

id	Contact's Guid

Returns

true if object was deleted successfully false if not

Implements CRUDService.Repository.IContactRepository.

6.17.3.5 FindAllCategories()

```
async Task< List< Category > > CRUDService.Repository.ContactRepository.FindAllCategories ()
[inline]
```

Retrieves all categories.

Returns

Category object

Implements CRUDService.Repository.IContactRepository.

6.17.3.6 FindAllContacts()

```
async Task< List< Contact > > CRUDService.Repository.ContactRepository.FindAllContacts (
             int skip,
             int take)  [inline]
```

Retrieves a paginated list of contacts.

Parameters

skip	Defines how many records should be skipped
take	Defines max length of returned list

Returns

Either Contact object or null

Implements CRUDService.Repository.IContactRepository.

6.17.3.7 FindAllSubCategories()

```
async Task< List< SubCategory > > CRUDService.Repository.ContactRepository.FindAllSubCategories
()  [inline]
```

Fetches all subcategoires.

Returns

List of all subcategories

Implements CRUDService.Repository.IContactRepository.

6.17.3.8 FindAllSubCategoriesById()

```
async Task< List< string > > CRUDService.Repository.ContactRepository.FindAllSubCategories←
ById (
             int id)  [inline]
```

Fetches all subcategoires for given id.

Parameters

id	Id of a Category

Returns

List of names of all the subcategory names belonging to the given Category

Implements CRUDService.Repository.IContactRepository.

6.17.3.9 FindCategoryNameById()

```
async Task< string?> CRUDService.Repository.ContactRepository.FindCategoryNameById (
             int id)  [inline]
```

Retrieves a category name for given id.

Parameters

id	Id of a Category

Returns

Category's name

Implements CRUDService.Repository.IContactRepository.

6.17.3.10 FindContactById()

```
async Task< Contact?> CRUDService.Repository.ContactRepository.FindContactById (
             Guid id)  [inline]
```

Retrieves contact by its id.

Parameters

id	Contacts Guid

Returns

Either Contact object or null

Implements CRUDService.Repository.IContactRepository.

6.17.3.11 FindUserEmailById()

```
async Task< string?> CRUDService.Repository.ContactRepository.FindUserEmailById (
             Guid id)  [inline]
```

Retrieves user email by her/his id.

Parameters

id	Contact's Guid

Returns

Contact's email

Implements CRUDService.Repository.IContactRepository.

6.17.3.12 UpdateContact()

```
async Task< bool > CRUDService.Repository.ContactRepository.UpdateContact (
            Contact contact)  [inline]
```

Updates existing contact.

Parameters

contact	Contact object

Returns

true if object was updated successfully false if not

Implements CRUDService.Repository.IContactRepository.

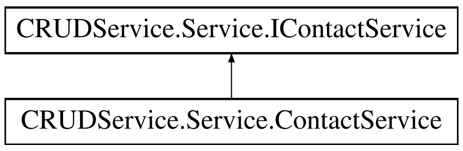
The documentation for this class was generated from the following file:

- CRUDService/Repository/ContactRepository.cs

6.18 CRUDService.Service.ContactService Class Reference

Implements bussiness logic and mediates between contrler and repository layers.

Inheritance diagram for CRUDService.Service.ContactService:


CRUDService.Service.IContactService
CRUDService.Service.ContactService

Public Member Functions

- ContactService (IContactRepository contactRepo)

  Constructor for supporting injections.
- async Task< bool > AddContact (Contact contact)

  Adds new contact.
- async Task< bool > CheckIfEmailTaken (string email)

  Retrieves contact by its Email.
- async Task< bool > CheckIfIdExists (Guid id)

  Checks if contact with given id exists.
- async Task< bool > CommonContactValidation (Contact contact)

  Performs validations for update and creation of contact.
- async Task< bool > DeleteContact (Guid id)

  Deletes existing contact.
- async Task< List< Category > > FindAllCategories ()

  Retrieves all categories.
- async Task< List< Contact > > FindAllContacts (int skip, int take)

  Retrieves a paginated list of contacts.
- async Task< List< SubCategory > > FindAllSubCategories ()

  Fetches all subcategoires.
- async Task< List< string > > FindAllSubCategoriesById (int id)

  Fetches all subcategoires by category id.
- async Task< string?> FindCategoryNameById (int id)

  Retrieves a category name for given id.
- async Task< Contact?> FindContactById (Guid id)

  Retrieves contact by its id.
- async Task< string?> FindUserEmailById (Guid id)

  Retrieves user email by her/his id.
- async Task< bool > UpdateContact (Contact contact)

  Updates existing contact.

Private Attributes

- readonly IContactRepository **_contactRepo**

6.18.1 Detailed Description

Implements bussiness logic and mediates between contrler and repository layers.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 ContactService()

```
CRUDService.Service.ContactService.ContactService (
            IContactRepository contactRepo)  [inline]
```

Constructor for supporting injections.

Parameters

contactRepo	Injected ContactRepository object

6.18.3 Member Function Documentation

6.18.3.1 AddContact()

```
async Task< bool > CRUDService.Service.ContactService.AddContact (
            Contact contact)  [inline]
```

Adds new contact.

Parameters

contact	Contact object

Returns

bool whether Contact object was successfully added to the database

Implements CRUDService.Service.IContactService.

6.18.3.2 CheckIfEmailTaken()

```
async Task< bool > CRUDService.Service.ContactService.CheckIfEmailTaken (
            string email)  [inline]
```

Retrieves contact by its Email.

Parameters

email	

Returns

bool saying whether email exists or not

Implements CRUDService.Service.IContactService.

6.18.3.3 CheckIfIdExists()

```
async Task< bool > CRUDService.Service.ContactService.CheckIfIdExists (
            Guid id)  [inline]
```

Checks if contact with given id exists.

Parameters

id	Contact's Guid

Returns

bool saying whether object with given id exists or not

Implements CRUDService.Service.IContactService.

6.18.3.4 CommonContactValidation()

```
async Task< bool > CRUDService.Service.ContactService.CommonContactValidation (
            Contact contact) [inline]
```

Performs validations for update and creation of contact.

Parameters

contact	Contact object

Returns

bool saying whether given Contact object's properties

Implements CRUDService.Service.IContactService.

6.18.3.5 DeleteContact()

```
async Task< bool > CRUDService.Service.ContactService.DeleteContact (
            Guid id) [inline]
```

Deletes existing contact.

Parameters

id	Contact's Guid

Returns

bool sqying whether deletion was successful or not

Implements CRUDService.Service.IContactService.

6.18.3.6 FindAllCategories()

```
async Task< List< Category > > CRUDService.Service.ContactService.FindAllCategories ()  [inline]
```

Retrieves all categories.

Returns

List of Category objects in the database

Implements CRUDService.Service.IContactService.

6.18.3.7 FindAllContacts()

```
async Task< List< Contact > > CRUDService.Service.ContactService.FindAllContacts (
            int skip,
            int take)  [inline]
```

Retrieves a paginated list of contacts.

Parameters

skip	Defines how many records should be skipped
take	Defines max length of returned list

Returns

Either Contact object or null

Implements CRUDService.Service.IContactService.

6.18.3.8 FindAllSubCategories()

```
async Task< List< SubCategory > > CRUDService.Service.ContactService.FindAllSubCategories ()
[inline]
```

Fetches all subcategoires.

Returns

List of all SubCategory objects in the database

Implements CRUDService.Service.IContactService.

6.18.3.9 FindAllSubCategoriesById()

```
async Task< List< string > > CRUDService.Service.ContactService.FindAllSubCategoriesById (
            int id)  [inline]
```

Fetches all subcategoires by category id.

Parameters

id	

Returns

List of names of all the subcategory names belonging to the given Category

Implements CRUDService.Service.IContactService.

6.18.3.10 FindCategoryNameById()

```
async Task< string?> CRUDService.Service.ContactService.FindCategoryNameById (
             int id)  [inline]
```

Retrieves a category name for given id.

Parameters

id	Id of Category object

Returns

List of names of all subcategories of category with given id

Implements CRUDService.Service.IContactService.

6.18.3.11 FindContactById()

```
async Task< Contact?> CRUDService.Service.ContactService.FindContactById (
             Guid id)  [inline]
```

Retrieves contact by its id.

Parameters

id	Contact's Guid

Returns

Contact object or null

Implements CRUDService.Service.IContactService.

6.18.3.12 FindUserEmailById()

```
async Task< string?> CRUDService.Service.ContactService.FindUserEmailById (
             Guid id)  [inline]
```

Retrieves user email by her/his id.

Parameters

id	Contact's Guid

Returns

Email as string or null

Implements CRUDService.Service.IContactService.

6.18.3.13 UpdateContact()

```
async Task< bool > CRUDService.Service.ContactService.UpdateContact (
            Contact contact)  [inline]
```

Updates existing contact.

Parameters

contact	Contact object

Returns

bool saying whether update was successful or not

Implements CRUDService.Service.IContactService.

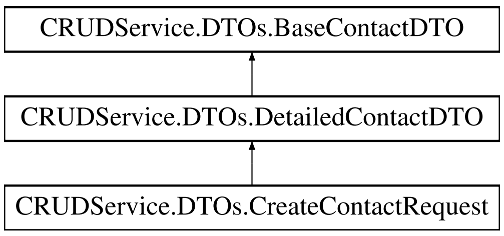
The documentation for this class was generated from the following file:

- CRUDService/Service/ContactService.cs

6.19 CRUDService.DTOs.CreateContactRequest Class Reference

DTO class used for updates and creation requests, it posses every field informative field of contact. Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for CRUDService.DTOs.CreateContactRequest:


CRUDService.DTOs.BaseContactDTO
CRUDService.DTOs.DetailedContactDTO
CRUDService.DTOs.CreateContactRequest

Properties

- required int **CategoryId** `[get, set]`
- required string **Password** `[get, set]`

Properties inherited from CRUDService.DTOs.DetailedContactDTO

- string? **SubCategory** `[get, set]`
- required string **Phone** `[get, set]`
- required DateOnly **BirthDate** `[get, set]`

Properties inherited from CRUDService.DTOs.BaseContactDTO

- Guid **Id** `[get, set]`
- required string **Name** `[get, set]`
- required string **Surname** `[get, set]`
- required string **Email** `[get, set]`

6.19.1 Detailed Description

DTO class used for updates and creation requests, it posses every field informative field of contact. Annotated for early valdiation and early rejection of invalid requests.

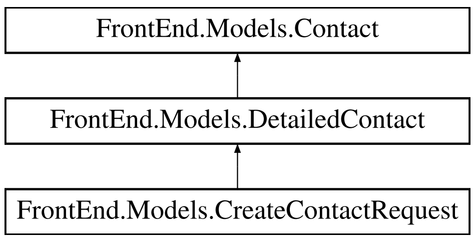
The documentation for this class was generated from the following file:

- CRUDService/DTOs/CreateContactRequest.cs

6.20 FrontEnd.Models.CreateContactRequest Class Reference

dTO class used for updates and creation requests, it posses every field informative field of contact Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for FrontEnd.Models.CreateContactRequest:


FrontEnd.Models.Contact
FrontEnd.Models.DetailedContact
FrontEnd.Models.CreateContactRequest

Properties

- int? **CategoryId** `[get, set]`
- string? **Password** `[get, set]`

Properties inherited from FrontEnd.Models.DetailedContact

- string? **SubCategory** `[get, set]`
- string? **Phone** `[get, set]`
- DateOnly **BirthDate** `[get, set]`

Properties inherited from FrontEnd.Models.Contact

- Guid **Id** `[get, set]`
- string? **Name** `[get, set]`
- string? **Surname** `[get, set]`
- string? **Email** `[get, set]`

6.20.1 Detailed Description

dTO class used for updates and creation requests, it posses every field informative field of contact Annotated for early valdiation and early rejection of invalid requests.

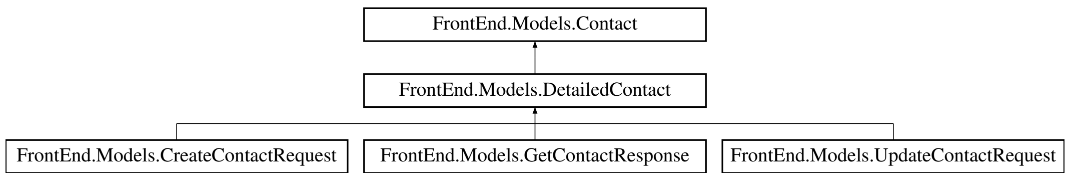
The documentation for this class was generated from the following file:

- FrontEnd/Models/CreateContactRequest.cs

6.21 FrontEnd.Models.DetailedContact Class Reference

More detailed DTO used for adding missing fields from BASEContactDTO. It's used to minimize code duplication. Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for FrontEnd.Models.DetailedContact:


FrontEnd.Models.Contact
FrontEnd.Models.DetailedContact
FrontEnd.Models.CreateContactRequest
FrontEnd.Models.GetContactResponse
FrontEnd.Models.UpdateContactRequest

Properties

- string? **SubCategory** `[get, set]`
- string? **Phone** `[get, set]`
- DateOnly **BirthDate** `[get, set]`

Properties inherited from FrontEnd.Models.Contact

- Guid **Id** `[get, set]`
- string? **Name** `[get, set]`
- string? **Surname** `[get, set]`
- string? **Email** `[get, set]`

6.21.1 Detailed Description

More detailed DTO used for adding missing fields from BASEContactDTO. It's used to minimize code duplication. Annotated for early valdiation and early rejection of invalid requests.

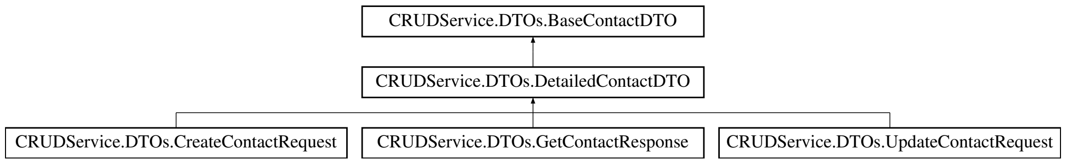
The documentation for this class was generated from the following file:

- FrontEnd/Models/DetailedContact.cs

6.22 CRUDService.DTOs.DetailedContactDTO Class Reference

More detailed DTO used for adding missing fields from BASEContactDTO. It's used to minimize code duplication. Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for CRUDService.DTOs.DetailedContactDTO:


CRUDService.DTOs.BaseContactDTO
CRUDService.DTOs.DetailedContactDTO
CRUDService.DTOs.CreateContactRequest
CRUDService.DTOs.GetContactResponse
CRUDService.DTOs.UpdateContactRequest

Properties

- string? **SubCategory** `[get, set]`
- required string **Phone** `[get, set]`
- required DateOnly **BirthDate** `[get, set]`

Properties inherited from CRUDService.DTOs.BaseContactDTO

- Guid **Id** `[get, set]`
- required string **Name** `[get, set]`
- required string **Surname** `[get, set]`
- required string **Email** `[get, set]`

6.22.1 Detailed Description

More detailed DTO used for adding missing fields from BASEContactDTO. It's used to minimize code duplication. Annotated for early valdiation and early rejection of invalid requests.

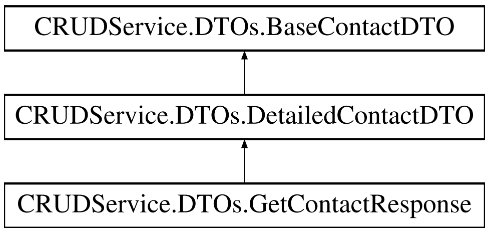
The documentation for this class was generated from the following file:

- CRUDService/DTOs/DetailedContactDTO.cs

6.23 CRUDService.DTOs.GetContactResponse Class Reference

Detailed DTO class used to transfer data needed for detailed view of contact info. Caution: Password field is purposefully omitted due to security reasons. Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for CRUDService.DTOs.GetContactResponse:


CRUDService.DTOs.BaseContactDTO
CRUDService.DTOs.DetailedContactDTO
CRUDService.DTOs.GetContactResponse

Properties

- required string **Category** `[get, set]`

Properties inherited from CRUDService.DTOs.DetailedContactDTO

- string? **SubCategory** `[get, set]`
- required string **Phone** `[get, set]`
- required DateOnly **BirthDate** `[get, set]`

Properties inherited from CRUDService.DTOs.BaseContactDTO

- Guid **Id** `[get, set]`
- required string **Name** `[get, set]`
- required string **Surname** `[get, set]`
- required string **Email** `[get, set]`

6.23.1 Detailed Description

Detailed DTO class used to transfer data needed for detailed view of contact info. Caution: Password field is purposefully omitted due to security reasons. Annotated for early valdiation and early rejection of invalid requests.

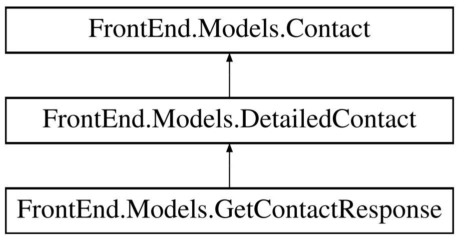
The documentation for this class was generated from the following file:

- CRUDService/DTOs/GetContactResponse.cs

6.24 FrontEnd.Models.GetContactResponse Class Reference

Detailed DTO class used to transfer data needed for detailed view of contact info. Caution: Password field is purposefully omitted due to security reasons. Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for FrontEnd.Models.GetContactResponse:


FrontEnd.Models.Contact
FrontEnd.Models.DetailedContact
FrontEnd.Models.GetContactResponse

Properties

- string? **Category** `[get, set]`

Properties inherited from FrontEnd.Models.DetailedContact

- string? **SubCategory** `[get, set]`
- string? **Phone** `[get, set]`
- DateOnly **BirthDate** `[get, set]`

Properties inherited from FrontEnd.Models.Contact

- Guid **Id** `[get, set]`
- string? **Name** `[get, set]`
- string? **Surname** `[get, set]`
- string? **Email** `[get, set]`

6.24.1 Detailed Description

Detailed DTO class used to transfer data needed for detailed view of contact info. Caution: Password field is purposefully omitted due to security reasons. Annotated for early valdiation and early rejection of invalid requests.

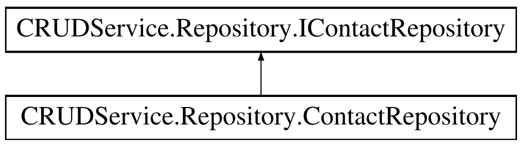
The documentation for this class was generated from the following file:

- FrontEnd/Models/GetContactResponse.cs

6.25 CRUDService.Repository.IContactRepository Interface Reference

Interface posseses collection of methods for handling data stored in the database.

Inheritance diagram for CRUDService.Repository.IContactRepository:


CRUDService.Repository.IContactRepository
CRUDService.Repository.ContactRepository

Public Member Functions

- Task< bool > CheckIfIdExists (Guid id)

    Checks if contact with given id exists.
- Task< Contact?> FindContactById (Guid id)

    Retrieves contact by its id.
- Task< bool > CheckIfEmailTaken (string email)

    Retrieves contact by its Email.
- Task< List< Contact > > FindAllContacts (int skip, int take)

    Retrieves a paginated list of contacts.
- Task< List< Category > > FindAllCategories ()

    Retrieves all categories.
- Task< string?> FindCategoryNameById (int id)

    Retrieves a category name for given id.
- Task< string?> FindUserEmailById (Guid id)

    Retrieves user email by her/his id.
- Task< List< string > > FindAllSubCategoriesById (int id)

    Fetches all subcategoires for given id.
- Task< List< SubCategory > > FindAllSubCategories ()

    Fetches all subcategoires.
- Task AddContact (Contact contact)

    Adds new contact.
- Task< bool > UpdateContact (Contact contact)

    Updates existing contact.
- Task< bool > DeleteContact (Guid id)

    Deletes existing contact.

6.25.1 Detailed Description

Interface posseses collection of methods for handling data stored in the database.

6.25.2 Member Function Documentation

6.25.2.1 AddContact()

```
Task CRUDService.Repository.IContactRepository.AddContact (
            Contact contact)
```

Adds new contact.

Parameters

contact	Contact object

Returns

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.2 CheckIfEmailTaken()

```
Task< bool > CRUDService.Repository.IContactRepository.CheckIfEmailTaken (
            string email)
```

Retrieves contact by its Email.

Parameters

email	

Returns

true if given email exists or flase if not

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.3 CheckIfIdExists()

```
Task< bool > CRUDService.Repository.IContactRepository.CheckIfIdExists (
             Guid id)
```

Checks if contact with given id exists.

Parameters

id	Contacts Guid

Returns

true if exists false if not

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.4 DeleteContact()

```
Task< bool > CRUDService.Repository.IContactRepository.DeleteContact (
             Guid id)
```

Deletes existing contact.

Parameters

id	Contact's Guid

Returns

true if object was deleted successfully false if not

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.5 FindAllCategories()

```
Task< List< Category > > CRUDService.Repository.IContactRepository.FindAllCategories ()
```

Retrieves all categories.

Returns

Category object

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.6 FindAllContacts()

```
Task< List< Contact > > CRUDService.Repository.IContactRepository.FindAllContacts (
            int skip,
            int take)
```

Retrieves a paginated list of contacts.

Parameters

skip	Defines how many records should be skipped
take	Defines max length of returned list

Returns

Either Contact object or null

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.7 FindAllSubCategories()

```
Task< List< SubCategory > > CRUDService.Repository.IContactRepository.FindAllSubCategories ()
```

Fetches all subcategoires.

Returns

List of all subcategories

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.8 FindAllSubCategoriesById()

```
Task< List< string > > CRUDService.Repository.IContactRepository.FindAllSubCategoriesById (
            int id)
```

Fetches all subcategoires for given id.

Parameters

id	Id of a Category

Returns

List of names of all the subcategory names belonging to the given Category

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.9 FindCategoryNameById()

```
Task< string?> CRUDService.Repository.IContactRepository.FindCategoryNameById (
            int id)
```

Retrieves a category name for given id.

Parameters

id	Id of a Category

Returns

Category's name

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.10 FindContactById()

```
Task< Contact?> CRUDService.Repository.IContactRepository.FindContactById (
            Guid id)
```

Retrieves contact by its id.

Parameters

id	Contacts Guid

Returns

Either Contact object or null

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.11 FindUserEmailById()

```
Task< string?> CRUDService.Repository.IContactRepository.FindUserEmailById (
            Guid id)
```

Retrieves user email by her/his id.

Parameters

id	Contact's Guid

Returns

Contact's email

Implemented in CRUDService.Repository.ContactRepository.

6.25.2.12 UpdateContact()

```
Task< bool > CRUDService.Repository.IContactRepository.UpdateContact (
             Contact contact)
```

Updates existing contact.

Parameters

contact	Contact object

Returns

true if object was updated successfully false if not

Implemented in CRUDService.Repository.ContactRepository.

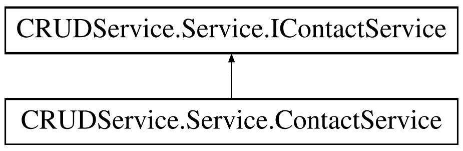
The documentation for this interface was generated from the following file:

- CRUDService/Repository/IContactRepository.cs

6.26 CRUDService.Service.IContactService Interface Reference

Defines bussiness logic operations for Contact management.

Inheritance diagram for CRUDService.Service.IContactService:


CRUDService.Service.IContactService
CRUDService.Service.ContactService

Public Member Functions

- Task< bool > CheckIfEmailTaken (string email)

    Retrieves contact by its Email.
- Task< bool > CheckIfIdExists (Guid id)

    Checks if contact with given id exists.
- Task< Contact?> FindContactById (Guid id)

    Retrieves contact by its id.
- Task< List< Contact > > FindAllContacts (int skip, int take)

    Retrieves a paginated list of contacts.
- Task< List< Category > > FindAllCategories ()

    Retrieves all categories.
- Task< string?> FindCategoryNameById (int id)

    Retrieves a category name for given id.
- Task< string?> FindUserEmailById (Guid id)

    Retrieves user email by her/his id.
- Task< List< string > > FindAllSubCategoriesById (int id)

    Fetches all subcategoires by category id.
- Task< List< SubCategory > > FindAllSubCategories ()

    Fetches all subcategoires.
- Task< bool > AddContact (Contact contact)

    Adds new contact.
- Task< bool > UpdateContact (Contact contact)

    Updates existing contact.
- Task< bool > DeleteContact (Guid id)

    Deletes existing contact.
- Task< bool > CommonContactValidation (Contact contact)

    Performs validations for update and creation of contact.

6.26.1 Detailed Description

Defines bussiness logic operations for Contact management.

6.26.2 Member Function Documentation

6.26.2.1 AddContact()

```
Task< bool > CRUDService.Service.IContactService.AddContact (
            Contact contact)
```

Adds new contact.

Parameters

contact	Contact object

Returns

bool whether Contact object was successfully added to the database

Implemented in CRUDService.Service.ContactService.

6.26.2.2 CheckIfEmailTaken()

```
Task< bool > CRUDService.Service.IContactService.CheckIfEmailTaken (
            string email)
```

Retrieves contact by its Email.

Parameters

email	

Returns

bool saying whether email exists or not

Implemented in CRUDService.Service.ContactService.

6.26.2.3 CheckIfIdExists()

```
Task< bool > CRUDService.Service.IContactService.CheckIfIdExists (
            Guid id)
```

Checks if contact with given id exists.

Parameters

id	Contact's Guid

Returns

bool saying whether object with given id exists or not

Implemented in CRUDService.Service.ContactService.

6.26.2.4 CommonContactValidation()

```
Task< bool > CRUDService.Service.IContactService.CommonContactValidation (
            Contact contact)
```

Performs validations for update and creation of contact.

Parameters

contact	Contact object

Returns

bool saying whether given Contact object's properties

Implemented in CRUDService.Service.ContactService.

6.26.2.5 DeleteContact()

```
Task< bool > CRUDService.Service.IContactService.DeleteContact (
            Guid id)
```

Deletes existing contact.

Parameters

id	Contact's Guid

Returns

bool sqying whether deletion was successful or not

Implemented in CRUDService.Service.ContactService.

6.26.2.6 FindAllCategories()

```
Task< List< Category > > CRUDService.Service.IContactService.FindAllCategories ()
```

Retrieves all categories.

Returns

List of Category objects in the database

Implemented in CRUDService.Service.ContactService.

6.26.2.7 FindAllContacts()

```
Task< List< Contact > > CRUDService.Service.IContactService.FindAllContacts (
            int skip,
            int take)
```

Retrieves a paginated list of contacts.

Parameters

skip	Defines how many records should be skipped
take	Defines max length of returned list

Returns

Either Contact object or null

Implemented in CRUDService.Service.ContactService.

6.26.2.8 FindAllSubCategories()

```
Task< List< SubCategory > > CRUDService.Service.IContactService.FindAllSubCategories ()
```

Fetches all subcategeoires.

Returns

List of all SubCategory objects in the database

Implemented in CRUDService.Service.ContactService.

6.26.2.9 FindAllSubCategoriesById()

```
Task< List< string > > CRUDService.Service.IContactService.FindAllSubCategoriesById (
            int id)
```

Fetches all subcategeoires by category id.

Parameters

id	

Returns

List of names of all the subcategory names belonging to the given Category

Implemented in CRUDService.Service.ContactService.

6.26.2.10 FindCategoryNameById()

```
Task< string?> CRUDService.Service.IContactService.FindCategoryNameById (
             int id)
```

Retrieves a category name for given id.

Parameters

id	Id of Category object

Returns

List of names of all subcategories of category with given id

Implemented in CRUDService.Service.ContactService.

6.26.2.11 FindContactById()

```
Task< Contact?> CRUDService.Service.IContactService.FindContactById (
             Guid id)
```

Retrieves contact by its id.

Parameters

id	Contact's Guid

Returns

Contact object or null

Implemented in CRUDService.Service.ContactService.

6.26.2.12 FindUserEmailById()

```
Task< string?> CRUDService.Service.IContactService.FindUserEmailById (
             Guid id)
```

Retrieves user email by her/his id.

Parameters

id	Contact's Guid

Returns

Email as string or null

Implemented in CRUDService.Service.ContactService.

6.26.2.13 UpdateContact()

```
Task< bool > CRUDService.Service.IContactService.UpdateContact (
            Contact contact)
```

Updates existing contact.

Parameters

contact	Contact object

Returns

bool saying whether update was successful or not

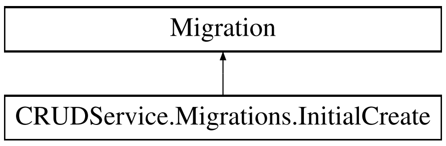
Implemented in CRUDService.Service.ContactService.

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

- CRUDService/Service/IContactService.cs

6.27 CRUDService.Migrations.InitialCreate Class Reference

Inheritance diagram for CRUDService.Migrations.InitialCreate:



Protected Member Functions

- override void **Up** (MigrationBuilder migrationBuilder)
- override void **Down** (MigrationBuilder migrationBuilder)
- override void **BuildTargetModel** (ModelBuilder modelBuilder)

6.27.1 Detailed Description

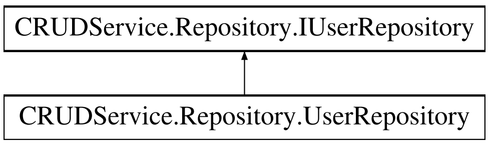
The documentation for this class was generated from the following files:

- CRUDService/Migrations/20250418231734_InitialCreate.cs
- CRUDService/Migrations/20250418231734_InitialCreate.Designer.cs

6.28 CRUDService.Repository.IUserRepository Interface Reference

Interface posseses collection of methods for handling data stored in the database.

Inheritance diagram for CRUDService.Repository.IUserRepository:


CRUDService.Repository.IUserRepository
CRUDService.Repository.UserRepository

Public Member Functions

- Task< string?> FindUserPasswordHash (string email)

    Retrieves password hash for given email.

6.28.1 Detailed Description

Interface posseses collection of methods for handling data stored in the database.

6.28.2 Member Function Documentation

6.28.2.1 FindUserPasswordHash()

```
Task< string?> CRUDService.Repository.IUserRepository.FindUserPasswordHash (
            string email)
```

Retrieves password hash for given email.

Parameters

email	

Returns

Hash of a passwrod associated with given email

Implemented in CRUDService.Repository.UserRepository.

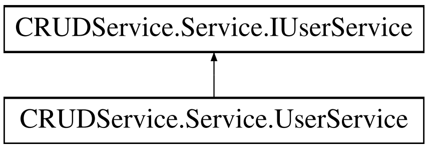
The documentation for this interface was generated from the following file:

- CRUDService/Repository/IUserRepository.cs

6.29 CRUDService.Service.IUserService Interface Reference

Defines bussiness logic in regards authentication of users.

Inheritance diagram for CRUDService.Service.IUserService:


CRUDService.Service.IUserService
CRUDService.Service.UserService

Public Member Functions

- Task< string?> FindUserPasswordHash (string email)

  Retrieves password hash for given email.
- Task< string?> Authenticate (User user)

  Hadnles user authentication.
- Task< bool > VerifyPassword (string givenHash, string email)

  Verifies given password with the one stored in database.
- string GenerateJwtToken (string email)

  Generates jwt token for authorization.

6.29.1 Detailed Description

Defines bussiness logic in regards authentication of users.

6.29.2 Member Function Documentation

6.29.2.1 Authenticate()

```
Task< string?> CRUDService.Service.IUserService.Authenticate (
            User user)
```

Hadnles user authentication.

Parameters

user	User object

Returns

Returns Jwttoken as string or null

Implemented in CRUDService.Service.UserService.

6.29.2.2 FindUserPasswordHash()

```
Task< string?> CRUDService.Service.IUserService.FindUserPasswordHash (
            string email)
```

Retrieves password hash for given email.

Parameters

email	

Returns

Hashed password from the database for the given email or null

Implemented in CRUDService.Service.UserService.

6.29.2.3 GenerateJwtToken()

```
string CRUDService.Service.IUserService.GenerateJwtToken (
            string email)
```

Generates jwt token for authorization.

Parameters

email	

Returns

Jwt token as string

Implemented in CRUDService.Service.UserService.

6.29.2.4 VerifyPassword()

```
Task< bool > CRUDService.Service.IUserService.VerifyPassword (
            string givenHash,
            string email)
```

Verifies given password with the one stored in database.

Parameters

givenHash	Hshed password
email	

Returns

bool sqying whether givenHash is the same as the one stored in the databse for the given email

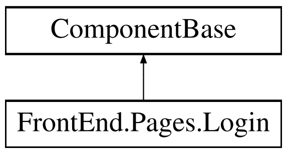
Implemented in CRUDService.Service.UserService.

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

- CRUDService/Service/IUserService.cs

6.30 FrontEnd.Pages.Login Class Reference

Inheritance diagram for FrontEnd.Pages.Login:



Protected Member Functions

- override async Task OnInitializedAsync ()

  Redirects logged in users to homepage.

Private Member Functions

- async Task LogIn ()

  Sends Login request.

Private Attributes

- User **user** = new()

6.30.1 Member Function Documentation

6.30.1.1 LogIn()

```
async Task FrontEnd.Pages.Login.LogIn ()  [inline], [private]
```

Sends Login request.

Returns

6.30.1.2 OnInitializedAsync()

```
override async Task FrontEnd.Pages.Login.OnInitializedAsync ()  [inline], [protected]
```

Redirects logged in users to homepage.

Returns

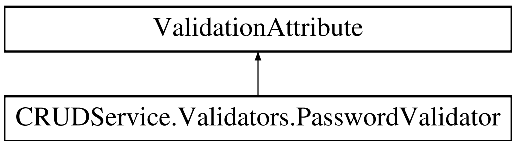
The documentation for this class was generated from the following file:

- FrontEnd/Pages/Login.razor.cs

6.31 CRUDService.Validators.PasswordValidator Class Reference

Custom password validator for use in automatic password validation through annotations.

Inheritance diagram for CRUDService.Validators.PasswordValidator:



Public Member Functions

- override bool IsValid (object value)

  Checks if given value is valid.
- override string FormatErrorMessage (string name)

  Returns error message when validation failed.

Properties

- bool **IsEdit** `[get, set]`

6.31.1 Detailed Description

Custom password validator for use in automatic password validation through annotations.

6.31.2 Member Function Documentation

6.31.2.1 FormatErrorMessage()

```
override string CRUDService.Validators.PasswordValidator.FormatErrorMessage (
             string name)  [inline]
```

Returns error message when validation failed.

Parameters

name	Unused parameter from overriden method

Returns

Error message

6.31.2.2 IsValid()

```
override bool CRUDService.Validators.PasswordValidator.IsValid (
             object value)  [inline]
```

Checks if given value is valid.

Parameters

value	

Returns

true iof valid false otherwise

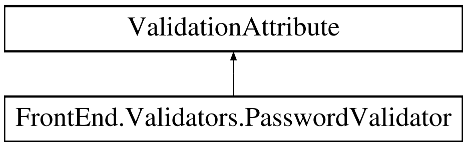
The documentation for this class was generated from the following file:

- CRUDService/Validators/PasswordValidator.cs

6.32 FrontEnd.Validators.PasswordValidator Class Reference

Custom password validator for use in automatic password validation through annotations.

Inheritance diagram for FrontEnd.Validators.PasswordValidator:



Public Member Functions

- override bool IsValid (object value)

    Checks if given value is valid.
- override string FormatErrorMessage (string name)

    Returns error message when validation failed.

Properties

- bool **IsEdit** `[get, set]`

6.32.1 Detailed Description

Custom password validator for use in automatic password validation through annotations.

6.32.2 Member Function Documentation

6.32.2.1 FormatErrorMessage()

```
override string FrontEnd.Validators.PasswordValidator.FormatErrorMessage (
            string name)  [inline]
```

Returns error message when validation failed.

Parameters

name	Unused parameter from overriden method

Returns

Error message

6.32.2.2 IsValid()

```
override bool FrontEnd.Validators.PasswordValidator.IsValid (
             object value)  [inline]
```

Checks if given value is valid.

Parameters

value	

Returns

true iof valid false otherwise

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

- FrontEnd/Validators/PasswordValidator.cs

6.33 CRUDService.Models.SubCategory Class Reference

Entity class for Dictionary entity SubCategory it is used only for dictionary entries For SubCategory that are limited to bussiness Category For database table constraint details see Contracts/SubCategoryConfiguration.cs file.

Properties

- int **Id** `[get, set]`
- required string **Name** `[get, set]`
- required int **CategoryId** `[get, set]`

6.33.1 Detailed Description

Entity class for Dictionary entity SubCategory it is used only for dictionary entries For SubCategory that are limited to bussiness Category For database table constraint details see Contracts/SubCategoryConfiguration.cs file.

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

- CRUDService/Models/SubCategory.cs

6.34 FrontEnd.Models.SubCategory Class Reference

Entity class for Dictionary entity SubCategory it is used only for dictionary entries For SubCategory that are limited to bussiness Category For database table constraint details see Contracts/SubCategoryConfiguration.cs file Annotated for early valdiation and early rejection of invalid requests.

Properties

- int **Id** `[get, set]`
- required string **Name** `[get, set]`
- required int **CategoryId** `[get, set]`

6.34.1 Detailed Description

Entity class for Dictionary entity SubCategory it is used only for dictionary entries For SubCategory that are limited to bussiness Category For database table constraint details see Contracts/SubCategoryConfiguration.cs file Annotated for early valdiation and early rejection of invalid requests.

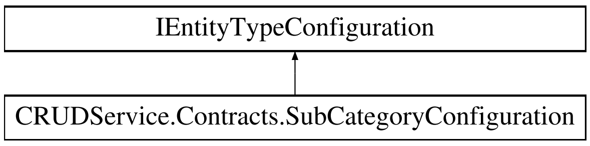
The documentation for this class was generated from the following file:

- FrontEnd/Models/SubCategory.cs

6.35 CRUDService.Contracts.SubCategoryConfiguration Class Reference

Class needed for configuring specific rules of its table in the db.

Inheritance diagram for CRUDService.Contracts.SubCategoryConfiguration:



Public Member Functions

- void Configure (EntityTypeBuilder< SubCategory > builder)
  Sets configuration of the Contacts table.

6.35.1 Detailed Description

Class needed for configuring specific rules of its table in the db.

6.35.2 Member Function Documentation

6.35.2.1 Configure()

```
void CRUDService.Contracts.SubCategoryConfiguration.Configure (
            EntityTypeBuilder< SubCategory > builder)  [inline]
```

Sets configuration of the Contacts table.

Parameters

builder	

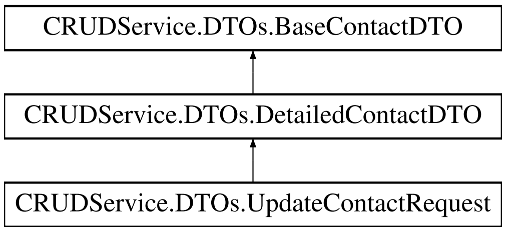
The documentation for this class was generated from the following file:

- CRUDService/Contracts/SubCategoryConfiguration.cs

6.36 CRUDService.DTOs.UpdateContactRequest Class Reference

DTO class for use when recieving Contact update requests. Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for CRUDService.DTOs.UpdateContactRequest:


CRUDService.DTOs.BaseContactDTO
CRUDService.DTOs.DetailedContactDTO
CRUDService.DTOs.UpdateContactRequest

Properties

- required int **CategoryId** `[get, set]`
- string? **Password** `[get, set]`

Properties inherited from CRUDService.DTOs.DetailedContactDTO

- string? **SubCategory** `[get, set]`
- required string **Phone** `[get, set]`
- required DateOnly **BirthDate** `[get, set]`

Properties inherited from CRUDService.DTOs.BaseContactDTO

- Guid **Id** `[get, set]`
- required string **Name** `[get, set]`
- required string **Surname** `[get, set]`
- required string **Email** `[get, set]`

6.36.1 Detailed Description

DTO class for use when recieving Contact update requests. Annotated for early valdiation and early rejection of invalid requests.

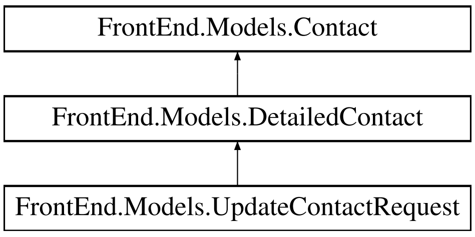
The documentation for this class was generated from the following file:

- CRUDService/DTOs/UpdateContactRequest.cs

6.37 FrontEnd.Models.UpdateContactRequest Class Reference

DTO class for use when recieving Contact update requests Annotated for early valdiation and early rejection of invalid requests.

Inheritance diagram for FrontEnd.Models.UpdateContactRequest:


FrontEnd.Models.Contact
FrontEnd.Models.DetailedContact
FrontEnd.Models.UpdateContactRequest

Properties

- int? **CategoryId** `[get, set]`
- string? **Password** `[get, set]`

Properties inherited from FrontEnd.Models.DetailedContact

- string? **SubCategory** `[get, set]`
- string? **Phone** `[get, set]`
- DateOnly **BirthDate** `[get, set]`

Properties inherited from FrontEnd.Models.Contact

- Guid **Id** `[get, set]`
- string? **Name** `[get, set]`
- string? **Surname** `[get, set]`
- string? **Email** `[get, set]`

6.37.1 Detailed Description

DTO class for use when recieving Contact update requests Annotated for early valdiation and early rejection of invalid requests.

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

- FrontEnd/Models/UpdateContactRequest.cs

6.38 CRUDService.Models.User Class Reference

Class representing a user.

Properties

- required string **Email** `[get, set]`
- required string **Password** `[get, set]`

6.38.1 Detailed Description

Class representing a user.

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

- CRUDService/Models/User.cs

6.39 FrontEnd.Models.User Class Reference

Class representing a user. Annotated for early valdiation and early rejection of invalid requests.

Properties

- string? **Email** `[get, set]`
- string? **Password** `[get, set]`

6.39.1 Detailed Description

Class representing a user. Annotated for early valdiation and early rejection of invalid requests.

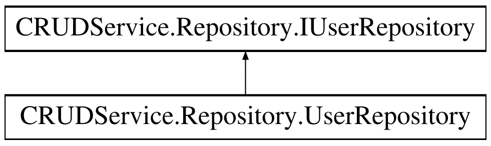
The documentation for this class was generated from the following file:

- FrontEnd/Models/User.cs

6.40 CRUDService.Repository.UserRepository Class Reference

Implementatiion of IUserRepository interface.

Inheritance diagram for CRUDService.Repository.UserRepository:


CRUDService.Repository.IUserRepository
CRUDService.Repository.UserRepository

Public Member Functions

- UserRepository (AppDbContext context)

  Constructor.
- async Task< string?> FindUserPasswordHash (string email)

  Retrieves password hash for given email.

Private Attributes

- readonly AppDbContext **_context**

6.40.1 Detailed Description

Implementatiion of IUserRepository interface.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 UserRepository()

```
CRUDService.Repository.UserRepository.UserRepository (
            AppDbContext context)  [inline]
```

Constructor.

Parameters

context	

6.40.3 Member Function Documentation

6.40.3.1 FindUserPasswordHash()

```
async Task< string?> CRUDService.Repository.UserRepository.FindUserPasswordHash (
            string email)  [inline]
```

Retrieves password hash for given email.

Parameters

email	

Returns

Hash of a passwrod associated with given email

Implements CRUDService.Repository.IUserRepository.

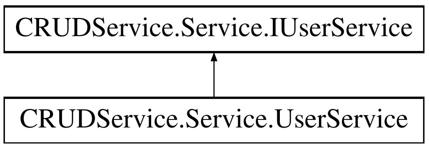
The documentation for this class was generated from the following file:

- CRUDService/Repository/UserRepository.cs

6.41 CRUDService.Service.UserService Class Reference

Class that implements IUserService interface.

Inheritance diagram for CRUDService.Service.UserService:


CRUDService.Service.IUserService
CRUDService.Service.UserService

Public Member Functions

- UserService (IUserRepository userRepo, IConfiguration config)

  Constructor supporting DI.
- async Task< string?> Authenticate (User user)

  Hadnles user authentication.
- async Task< string?> FindUserPasswordHash (string email)

  Retrieves password hash for given email.
- async Task< bool > VerifyPassword (string givenHash, string email)

  Verifies given password with the one stored in database.
- string GenerateJwtToken (string email)

  Generates jwt token for authorization.

Private Attributes

- readonly IUserRepository **_userRepo**
- readonly IConfiguration **_config**

6.41.1 Detailed Description

Class that implements IUserService interface.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 UserService()

```
CRUDService.Service.UserService.UserService (
                IUserRepository userRepo,
                IConfiguration config)  [inline]
```

Constructor supporting DI.

Parameters

userRepo	Injected IUserRepository object
config	Injected IConfiguration object

6.41.3 Member Function Documentation

6.41.3.1 Authenticate()

```
async Task< string?> CRUDService.Service.UserService.Authenticate (
            User user)  [inline]
```

Hadnles user authentication.

Parameters

user	User object

Returns

Returns Jwttoken as string or null

Implements CRUDService.Service.IUserService.

6.41.3.2 FindUserPasswordHash()

```
async Task< string?> CRUDService.Service.UserService.FindUserPasswordHash (
            string email)  [inline]
```

Retrieves password hash for given email.

Parameters

email	

Returns

Hashed password from the database for the given email or null

Implements CRUDService.Service.IUserService.

6.41.3.3 GenerateJwtToken()

```
string CRUDService.Service.UserService.GenerateJwtToken (
            string email)  [inline]
```

Generates jwt token for authorization.

Parameters

email	

Returns

Jwt token as string

Implements CRUDService.Service.IUserService.

6.41.3.4 VerifyPassword()

```
async Task< bool > CRUDService.Service.UserService.VerifyPassword (
            string givenHash,
            string email)  [inline]
```

Verifies given password with the one stored in database.

Parameters

givenHash	Hshed password
email	

Returns

bool sqying whether givenHash is the same as the one stored in the databse for the given email

Implements CRUDService.Service.IUserService.

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

- CRUDService/Service/UserService.cs

Index