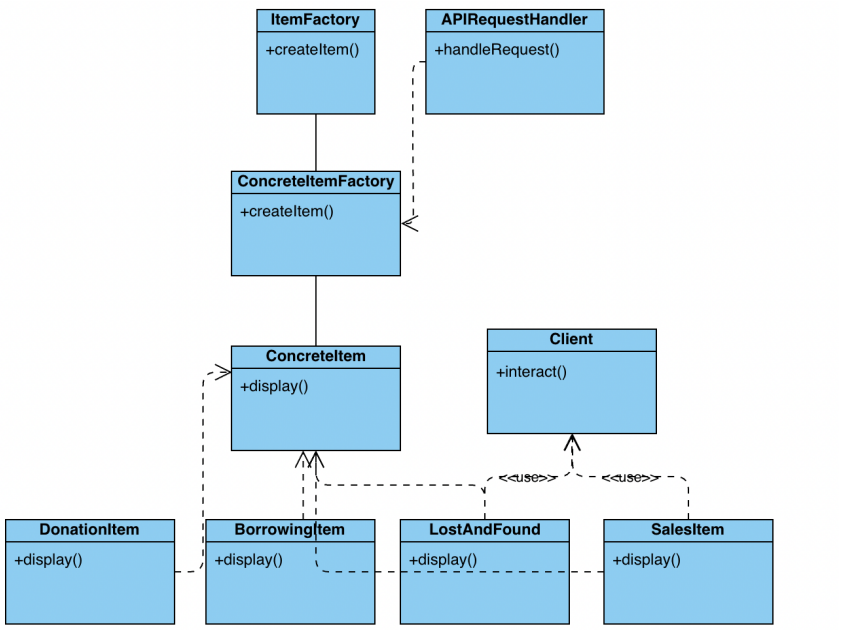
CS319 - Object Oriented Software Engineering

D5: Class Diagram, Design Patterns

Ahmet Kaan Sever - 22102278
Atakan Şerifoğlu - 22102313
Dila Tosun - 2212100
Konuralp Kılınç - 22101791
Perit Dinçer - 22103102
Selimhan Tokat - 22003145

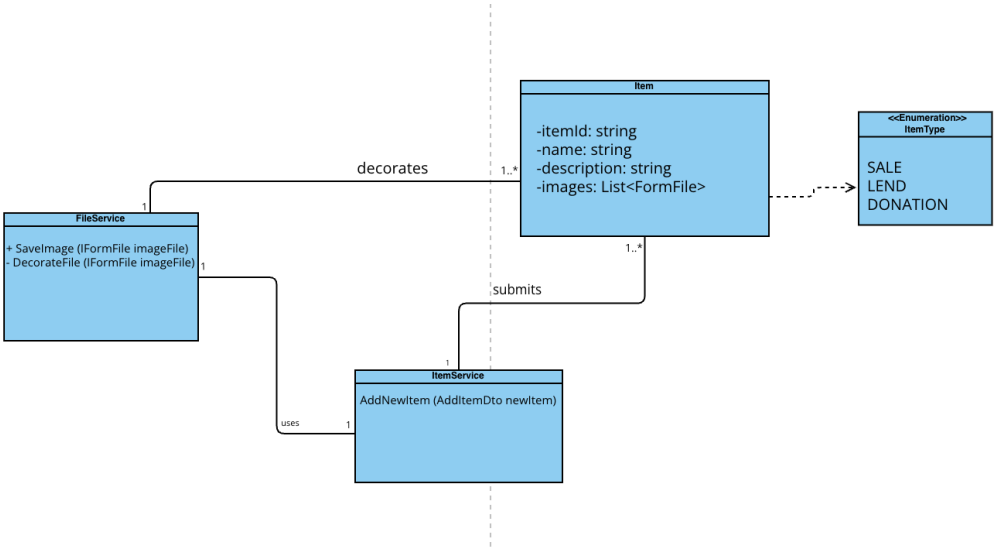
1. DESIGN PATTERNS

1.1. Factory



In CampusConnect, different items are shown as different card components such as donations, borrowing, sales and lost and found cards. To fetch the necessary item information a specialized API request to the database is sent. By using the Factory pattern, an abstract implementation for all items is implemented and specialized item logic is provided for different card items. With Factory design pattern, modularity and extensible architecture is sustained. Whenever a user interacts with a card component, the getter method is executed and the appropriate item instance is created through the API request. It is ensured that each card gets specialized versions of the item. Also, the addition of new card types without necessarily modifying the existing codebase it sustained, since there is an abstract item implementation. Moreover, the Factory design pattern helps with code reusability and encapsulation. Since there is a common pattern for the base of the items and it allows flexibility for creating new card items, it also promotes polymorphism.

1.2. Decorator



Both users and administrators actively engage in file upload and deletion operations on CampusConnect. Photos play a pivotal role in the CampusConnect platform, allowing users to upload up to five images for each item. To optimize security, the project employs Azure Blob storage for file storage. Before uploading, a unique string is generated and applied to the file name, enhancing the security measures by making it more challenging for unauthorized access. The Decorator Pattern is instrumental in dynamically enhancing the file naming process. This pattern enables the system to generate unique strings, thus fortifying the security of the files stored in Azure Blob. The Decorator classes seamlessly add user-specific information, such as user IDs, form types, and timestamps, to the file names. In our photo uploading system, we have integrated a dynamic file name modification feature using the Decorator Pattern. Traditionally, users are not required to manually enter a name for the uploaded file. Instead, a FileNameDecorator has been introduced as a decorator class, extending the base PhotoUploader component. This decorator automatically generates or modifies the file name during the uploading process, providing a seamless and user-friendly experience. The FileNameDecorator encapsulates the logic for generating unique file names, enhancing the system's functionality without altering the core code. This approach aligns with the principles of the Decorator Pattern, promoting an open/closed design where new features, such as automatic file name adjustments, can be effortlessly integrated, fostering extensibility and maintaining the modularity of our photo uploading system

2. LOW LEVEL DESIGN

2.1. Object Design Trade-offs

Object Design Trade-offs are a concept used in software engineering. With every design choice we make a decision and compromise on something else with these desicions.

Security v. Usability

Users will be able to enter the CampusConnect platform by only using their emails and password without a two-step authentication process. This will improve the usability of the application but compromise the security of the system.

Functionality v. Usability

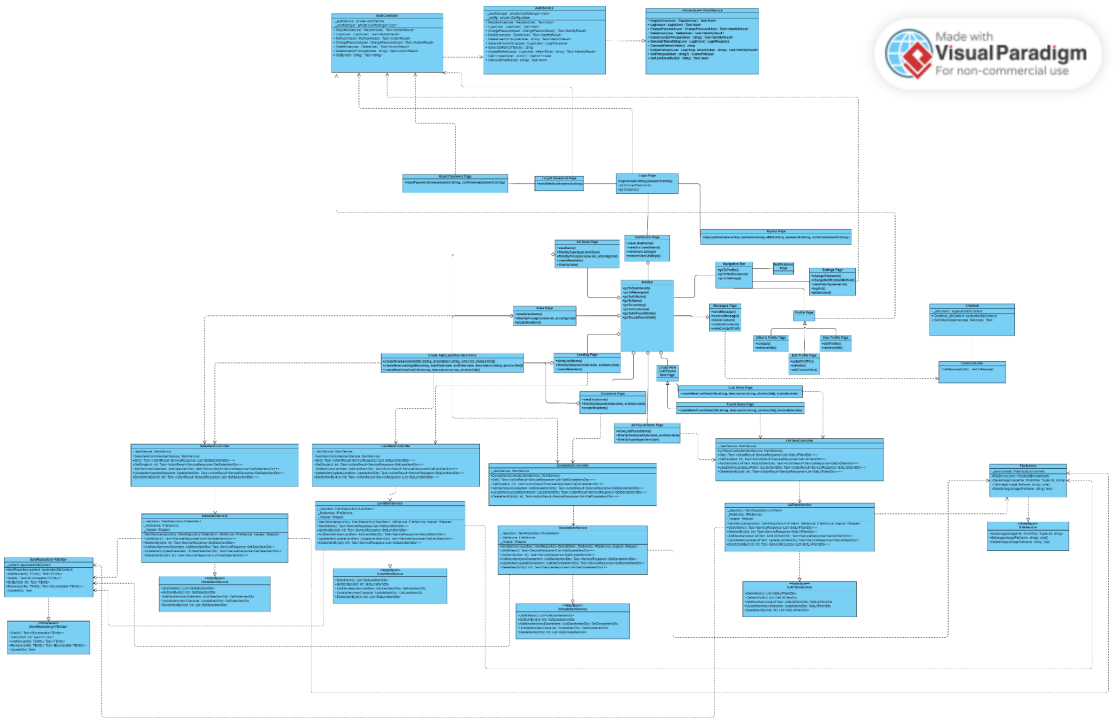
In our platform we decided not to implement some of the features that we thought we could add because either they were optional (e.g. implementing an online payment system is optional in the launch of the platform but can be implemented later.) or they were unnecessary (e.g. implementing shopping became unnecessary

because of the lack of an online payment system, checking out with items is up to the mutual consent of the users.). Thus, we focused on fewer features then we intended to, reducing the functionality of the platform, which allows us to create a simpler user interface.

Maintainability v. Performance

Our sales platform has many classes and roles for a user for each function. This design choice was made to easily write, change or update the platform for further development. This may slow down the rendering of the pages compared to a more compact platform. As a result this design compromises the performance of the system but increases maintainability and convenience of further development.

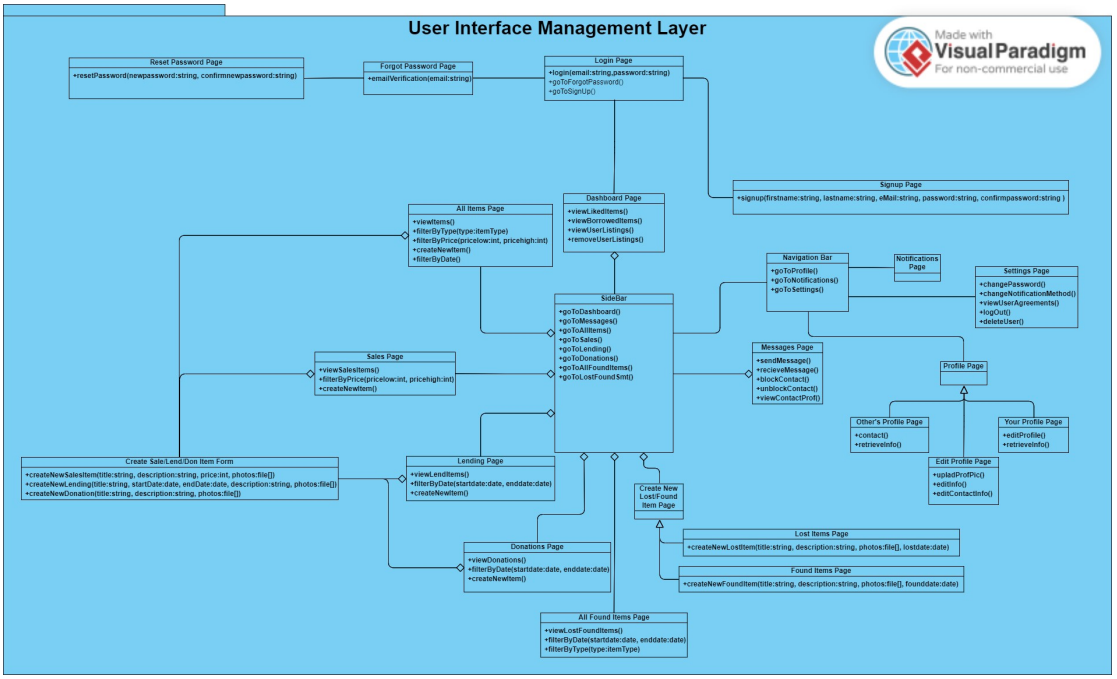
2.2. Final Object Design



FINAL CLASS DIAGRAM.vpd.png

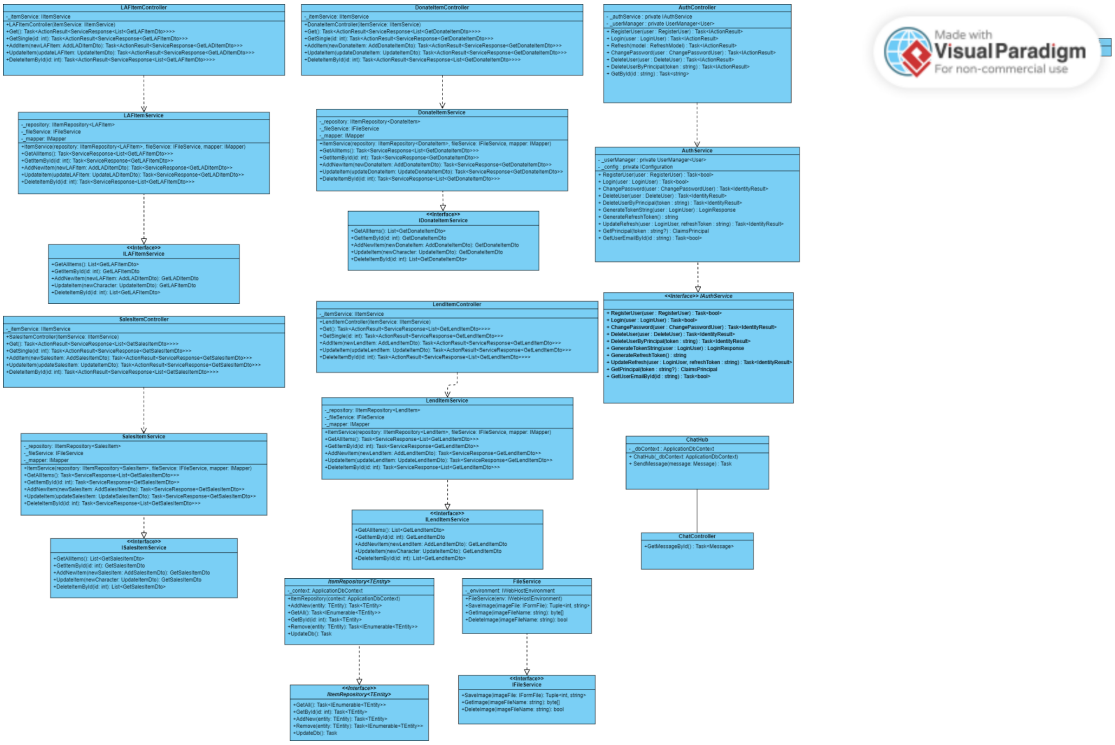
2.3. Layers

2.3.3. User Interface Management Layer



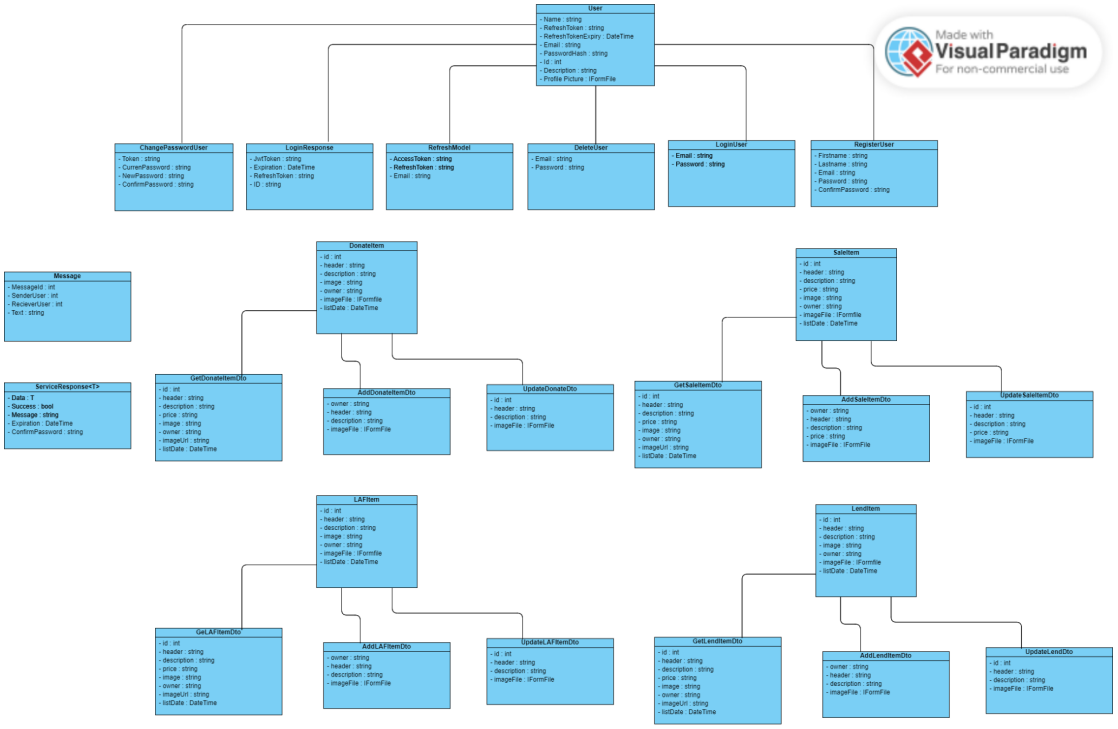
UIManLayFİNAL.vpd.png

2.3.4. Web Server Layer



a (3).png

2.3.5. Data Management Layer



Untitled (5).png

Reset Password Page

- resetPassword(newpassword: string, confirmpassword: string): This method allows a user to reset their password by entering a new password and confirming the password by entering it a second time.

Forgot Password Page

- emailVerification(email: string): This method likely sends a verification code or link to the user's email to initiate the password reset process.

Login Page

- login(email: string, password: string): Validates the user's email and password to log them into the system.
- goToForgotPassword(): Redirects the user to the Forgot Password Page.
- goToSignUp(): Redirects the user to the Signup Page.

Signup Page

- signup(firstname: string, lastname: string, email: string, password: string, confirmpassword: string): Registers a new user with the system using their personal details and desired credentials.

Dashboard Page

- viewLikedItems(): Displays the items that the user has marked as liked or favored.
- viewBorrowedItems(): Shows the items that the user is currently borrowing.
- viewUserListings(): Lists all items that the user has listed on the platform.
- removeUserListing(): Allows a user to remove one of their own listings from the platform.

All Items Page

- viewItems(): Displays all items available within the application.
- filterByType(type: itemType): Filters the displayed items based on their type/category.
- filterByPrice(priceLow: int, priceHigh: int): Filters the displayed items within a certain price range.
- createNewItem(): Allows the user to list a new item.
- filterByDate(): Filters the displayed items based on a specified date or date range.

Sidebar

- goToDashboard(): Navigates to the Dashboard Page.
- goToMessages(): Opens the Messages Page.
- goToAllItems(): Displays the All Items Page.
- goToSales(): Takes the user to the Sales Page.
- goToLending(): Redirects to the Lending Page.
- goToDonations(): Navigates to the Donations Page.
- goToAllFoundItems(): Opens the page displaying all found items.
- goToLostFoundSmt(): May navigate to a specialized search or submission tool for lost and found items.

Sales Page

- viewSalesItems(): Displays items that are currently for sale.
- filterByPrice(priceLow: int, priceHigh: int): Filters the sales items based on a price range specified by the user.
- createNewItem(): Initiates the process for a user to list a new item for sale.

Create Sale/Lend/Don Item Form

- createNewSalesItem(title: string, description: string, price: int, photos: file[]): Collects information and files to create a new sales listing.
- createNewLending(title: string, startDate: date, endDate: date, description: string, photos: file[]): Creates a new lending listing with specified start and end dates.
- createNewDonation(title: string, description: string, photos: file[]): Creates a new donation listing.

Lending Page

- viewLendItems(): Lists items available for borrowing.
- filterByDate(startDate: date, endDate: date): Filters the lendable items by the available date range.
- createNewItem(): Opens the Lend Item form on the page.

Donations Page

- viewDonations(): Shows the items that have been listed for donation.
- filterByDate(startDate: date, endDate: date): Filters the donation items based on the specified date range.
- createNewItem(): Enables the user to list a new item for donation.

Lost Items Page

- createNewLostItem(title: string, description: string, photos: file[], lostDate: date): Allows users to create a listing for an item they have lost, including the date it was lost.

Found Items Page

- createNewFoundItem(title: string, description: string, photos: file[], foundDate: date): Allows users to list an item they have found, along with the date it was found.

Your Profile Page

- editProfile(): Enables a user to edit their personal profile information.
- retrieveInfo(): Retrieves the user's profile information for viewing or editing.

Edit Profile Page

- uploadProfPic(): Allows the user to upload a new profile picture.
- editInfo(): Lets the user edit their personal information.
- editContactInfo(): Allows the user to update their contact information.

Other's Profile Page

- contact(): May enable a user to contact the owner of the profile.
- retrieveInfo(): Likely used to retrieve and display the information of another user's profile.

Navigation Bar

- goToProfile(): Redirects the user to their Profile Page.
- goToNotifications(): Opens the Notifications Page.
- goToSettings(): Navigates to the Settings Page.

Messages Page

- sendMessage(): Allows a user to send a message to another user.
- receiveMessage(): Receives messages sent to the user.
- blockContact(): Blocks communication from a specific contact.
- unblockContact(): Removes a block previously placed on a contact.
- viewContactProf(): Allows viewing the profile of the contact.

Settings Page

- changePassword(): Enables the user to change their account password.
- changeNotificationMethod(): Allows the user to change how they receive notifications.

- viewUserAgreements(): Provides access to the user agreements or terms of service.
- logOut(): Signs the user out of their account.
- deleteUser(): Allows the user to delete their account.

All Found Items Page

- viewFoundItems(): Displays all items that have been found and listed by users.
- filterByDate(foundDate: date): Filters found items by the date they were found.
- claimItem(): Could be a method to initiate a claim process for an item that a user has lost.

2.3.6. Web Server Layer

SalesItemService

- **Attributes:**

  -_repository : ItemRepository: Abstract general repository.

  -_fileService : IFileService: File service used to save and delete images.

  -_mapper : IMapper: Auto mapper used to map DTO's.

- **Operations:**

  +ItemService(repository: IItemRepository<SalesItem>, fileService: IFileService, mapper: IMapper):

  +GetAllItems(): Retrieves a list of all sales items from the repository.

  +GetItemById(id: int): Fetches a specific sales item by its identifier (id).

  +AddNewItem(newSalesItem: AddSalesItemDto): Adds a new sales item to the repository, using the data provided in newSalesItem.

  +UpdateItem(updateSalesItem: UpdateSalesItemDto): Updates an existing sales item with the new data provided in updateSalesItem.

  +DeleteItemById(id: int): Removes a sales item from the repository based on its identifier (id).

LendItemService

- **Attributes:**

-_repository: IItemRepository<LendItem>: Abstract general repository.

-_fileService: IFileService: File service used to save and delete images.

-_mapper: IMapper: Auto mapper used to map DTO's.

- **Operations:**

+ItemService(repository: IItemRepository<LendItem>, fileService: IFileService, mapper: IMapper):

+GetAllItems(): Retrieves a list of all lending items from the repository.

+GetItemById(id: int): Fetches a specific lending item by its identifier (id).

+AddNewItem(newLendItem: AddLendItemDto): Adds a new lend item to the repository, using the data provided in newLendItem.

+UpdateItem(updateLendItem: UpdateLendItemDto): Updates an existing lend item with the new data provided in updateLendItem.

+DeleteItemById(id: int): Removes a lending item from the repository based on its identifier (id).

DonateItemService

- **Attributes:**

-_repository: IItemRepository<DonateItem>: Abstract general repository.

-_fileService: IFileService: File service used to save and delete images.

-_mapper: IMapper: Auto mapper used to map DTO's.

- **Operations:**

+ItemService(repository: IItemRepository<DonateItem>, fileService: IFileService, mapper: IMapper):

+GetAllItems(): Retrieves a list of all donation items from the repository.

+GetItemById(id: int): Fetches a specific donation item by its identifier (id).

+AddNewItem(newDonateItem: AddDonateItemDto): Adds a new donation item to the repository, using the data provided in newDonationItem.

+UpdateItem(updateDonateItem: UpdateDonateItemDto): Updates an existing donation item with the new data provided in updateDonationItem.

+DeleteItemById(id: int): Removes a donation item from the repository based on its identifier (id).

LAFItemService

- **Attributes:**

  -_repository: IItemRepository<LAFItem>: Abstract general repository.

  -_fileService: IFileService: File service used to save and delete images.

  -_mapper: IMapper: Auto mapper used to map DTO's

- **Operations:**

  +ItemService(repository: IItemRepository<LAFItem>, fileService: IFileService, mapper: IMapper):

  +GetAllItems(): Retrieves a list of all Lost and Found items from the repository.

  +GetItemById(id: int): Fetches a specific Lost and Found item by its identifier (id).

  +AddNewItem(newLAFItem: AddLAFItemDto): Adds a new donation item to the repository, using the data provided in newLAFItem.

  +UpdateItem(updateLAFItem: UpdateLAFItemDto): Updates an existing Lost and Found item with the new data provided in updateLAFItem.

  +DeleteItemById(id: int): Removes a Lost and Found item from the repository based on its identifier (id).

FileService

- **Attributes:**

-_environment: IWebHostEnvironment: Current hosting environment.

- **Operations:**

+FileService(env: IWebHostEnvironment):

+SaveImage(imageFile: IFormFile): Saves an image file to the server.

+GetImage(imageFileName: string): Retrieves an image file from the server based on its file name.

+DeleteImage(imageFileName: string): Deletes an image file from the server based on its file name.

AuthService

- **Attributes:**

- _userManager : private UserManager<User>: A userManager instance for the service.

- _config : private IConfiguration: Config object for the service.

- **Operations:**

+ RegisterUser(user : RegisterUser) : Registers user to database.

+ Login(user : LoginUser) : Checks user credentials and logs in user.

+ ChangePassword(user : ChangePasswordUser) : Changes the user password.

+ DeleteUser(user : DeleteUser) : Deletes the user given.

+ DeleteUserByPrincipal(token : string) : Deletes the user with the given token.

+ GenerateTokenString(user : LoginUser) : Changes the description of the user with the given token and string.

+ GenerateRefreshToken() : Generates JWT and refresh tokens.

+ UpdateRefresh(user : LoginUser, refreshToken : string) : Generates refresh token.

+ GetPrincipal(token : string?) : Updates the refresh token in the database.

+ GetUserEmailById(id : string) : Returns email of the user id.

SalesItemController

- **Attributes:**

-_itemService: IItemService: ItemService used in this controller.

- **Operations:**

  +SalesItemController(itemService: IItemService): Manages dependency injection for ItemService.

  +Get(): Returns all sales items in the database.

  +GetSingle(id: int): Returns the sales item with the given id from the database.

  +AddItem(newSalesItem: AddSalesItemDto): Adds a new sales item to the database with the values from AddSalesItemDto and the logged in user as the owner.

  +UpdateItem(updateSalesItem: UpdateSalesItemDto): Changes the values of the item that has the given id inside UpdateItemDto with the corresponding values from UpdateItemDto.

  +DeleteItemById(id: int): Deletes the sales item with the given Id.

LendItemController

- **Attributes:**

  -_itemService: IItemService: ItemService used in this controller.

- **Operations:**

  +LendItemController(itemService: IItemService): Manages dependency injection for ItemService.

  +Get(): Returns all lend items in the database.

  +GetSingle(id: int): Returns the lend item with the given id from the database.

  +AddItem(newLendItem: AddLendItemDto): Adds a new lend item to the database with the values from AddLendItemDto and the logged in user as the owner.

  +UpdateItem(updateLendItem: UpdateLendItemDto): Changes the values of the item that has the given id inside UpdateItemDto with the corresponding values from UpdateItemDto.

  +DeleteItemById(id: int): Deletes the lending item with the given Id.

DonateItemController

- **Attributes:**

  -_itemService: IItemService: ItemService used in this controller.

- **Operations:**

  +DonateItemController(itemService: IItemService): Manages dependency injection for ItemService.

  +Get(): Returns all donate items in the database

  +GetSingle(id: int): Returns the donate item with the given id from the database.

  +AddItem(newDonateItem: AddDonateItemDto): Adds a new donate item to the database.

  +UpdateItem(updateDonateItem: UpdateItemDto): Updates an existing donate item in the database.

  +DeleteItemById(id: int): Deletes a donate item from the database based on its ID.

AuthController

- **Attributes:**

  - _authService : private IAuthService: A authService instance for the controller.

  - _userManager : private UserManager<User>: A userManager instance for the controller.

- **Operations:**

  + RegisterUser(user : RegisterUser) : Registers the user into the database using the authService method.

  + Login(user : LoginUser) : Logs in the user if the user has previously registered using the authService method.

  + Refresh(model : RefreshModel) : Checks and refreshes the user's JWT token.

  + ChangePassword(user : ChangePasswordUser) : Allows the user to change password.

  + DeleteUser(user : DeleteUser) : Deletes the user from the database.

  + DeleteUserByPrincipal(token : string) : Deletes the user without needing an email.

  + GetById(id : string) : Allows users to change their description.

LAFItemController

- **Attributes:**

  -_itemService: IItemService:

- **Operations:**

  +LAFItemController(itemService: IItemService): Manages dependency injection for ItemService.

  +Get(): Returns all Lost and Found items in the database

  +GetSingle(id: int): Returns the donate item with the given id from the database.

  +AddItem(newLAFItem: AddLAFItemDto): Adds a new lost and found item to the database.

  +UpdateItem(updateLAFItem: UpdateItemDto): Updates an existing lost and found item in the database.

  +DeleteItemById(id: int): Deletes a lost and found item from the database based on its ID.

ItemRepository

- **Attributes:**

  -_context: ApplicationDbContext

- **Operations:**

  +ItemRepository(context: ApplicationDbContext):

  +AddNew(entity: TEntity): Adds a new entity to the repository.

  +GetAll(): Retrieves all entities from the repository.

  +GetById(id: int): Fetches a specific entity from the repository based on its ID.

  +Remove(entity: TEntity): Removes an entity from the repository.

  +UpdateDb(): Updates the database with any changes made in the repository.

ChatHub

- **Attributes:**

  -_dbContext : ApplicationDbContext: This database context holds all the messages that are sent in the app. Reserves the attributes of the data layer class Message's attributes (mId, senderId, receiverId, text).

- **Operations:**

  +ChatHub(_dbContext: ApplicationDbContext): This constructor initializes the data tables according to the context and receives the users previous messages and when needed (user opens their chat) updates accordingly to the database.

  +SendMessage(message: Message): This asynchronous method updates the database with the newly created Message. In addition to that, it returns a task and via SignalR sends the message object that is created with Message class and messaging frontend. SignalR displays this item in real-time.

ChatController

- **Operations:**

  +GetMessageById(id: int): This asynchronous method allows admin users to receive desired messages by using message id (mId).

2.3.7. Data Management Layer

User

- **Attributes:**

  - Name : string

  - RefreshToken : string

  - RefreshTokenExpiry : DateTime

  - Email : string

  - PasswordHash : string

  - Id : int

  - Description : string

  - Profile Picture : IFormFile

ChangePasswordUser

- **Attributes:**

  - Token : string

  - CurrenPassword : string

  - NewPassword : string

  - ConfirmPassword : string

DeleteUser

- **Attributes:**
    - **Email :** string
    - **Password :** string

LoginUser

- **Attributes:**
    - **Email :** string
    - **Password :** string

RegisterUser

- **Attributes:**
    - **Firstname :** string
    - **Lastname :** string
    - **Email :** string
    - **Password :** string
    - **ConfirmPassword :** string

LoginResponse

- **Attributes:**
    - **JwtToken :** string
    - **Expiration :** DateTime
    - **RefreshToken :** string
    - **ID :** string

ServiceResponce

- **Attributes:**
    - **Data :** T
    - **Success :** bool
    - **Message :** string
    - **Expiration :** DateTime
    - **ConfirmPassword :** string

RefreshModel

- **Attributes:**
    - **AccessToken :** string
    - **RefreshToken :** string
    - **Email :** string

Message

- **Attributes:**

- **MessageId :** int
- **SenderUser :** int
- **RecieverUser :** int
- **Text :** string

LendItem

- **Attributes:**
  - **id :** int
  - **header :** string
  - **description :** string
  - **image :** string
  - **owner :** string
  - **imageFile :** IFormfile
  - **listDate :** DateTime

SaleItem

- **Attributes:**
  - **id :** int
  - **header :** string
  - **description :** string
  - **price :** string
  - **image :** string
  - **owner :** string
  - **imageFile :** IFormfile
  - **listDate :** DateTime

DonateItem

- **Attributes:**
  - **id :** int
  - **header :** string
  - **description :** string
  - **image :** string
  - **owner :** string
  - **imageFile :** IFormfile
  - **listDate :** DateTime

LAFItem

- **Attributes:**

- **id :** int
- **header :** string
- **description :** string
- **price :** string
- **image :** string
- **owner :** string
- **imageFile :** IFormfile
- **listDate :** DateTime

GetLendItemDto

- **Attributes:**
  - **id :** int
  - **header :** string
  - **description :** string
  - **image :** string
  - **owner :** string
  - **imageUrl :** string
  - **listDate :** DateTime

AddLendItemDto

- **Attributes:**
  - **owner :** string
  - **header :** string
  - **description :** string
  - **imageFile :** IFormFile

UpdateLendItemDto

- **Attributes:**
  - **id :** int
  - **header :** string
  - **description :** string
  - **imageFile :** IFormFile

GetSaleItemDto

- **Attributes:**

- id : int

- header : string

- description : string

- price : string

- image : string

- owner : string

- imageUrl : string

- listDate : DateTime

AddSaleItemDto

- **Attributes:**

- owner : string

- header : string

- description : string

- price : string

- imageFile : IFormFile

UpdateSaleItemDto

- **Attributes:**

- id : int

- header : string

- description : string

- price : string

- imageFile : IFormFile

GetLAFItemDto

- **Attributes:**

- id : int

- header : string

- description : string

- price : string

- image : string

- owner : string

- imageUrl : string

- listDate : DateTime

AddLAFItemDto

- **Attributes:**
    - **owner : string**
    - **header : string**
    - **description : string**
    - **price : string**
    - **imageFile : IFormFile**

UpdateLAFItemDto

- **Attributes:**
    - **id : int**
    - **header : string**
    - **description : string**
    - **price : string**
    - **imageFile : IFormFile**

GetDonateItemDto

- **Attributes:**
    - **id : int**
    - **header : string**
    - **description : string**
    - **image : string**
    - **owner : string**
    - **imageUrl : string**
    - **listDate : DateTime**

AddDonateItemDto

- **Attributes:**
    - **owner : string**
    - **header : string**
    - **description : string**
    - **imageFile : IFormFile**

UpdateDonateItemDto

- **Attributes:**
    - **id : int**
    - **header : string**
    - **description : string**

- imageFile : IFormFile