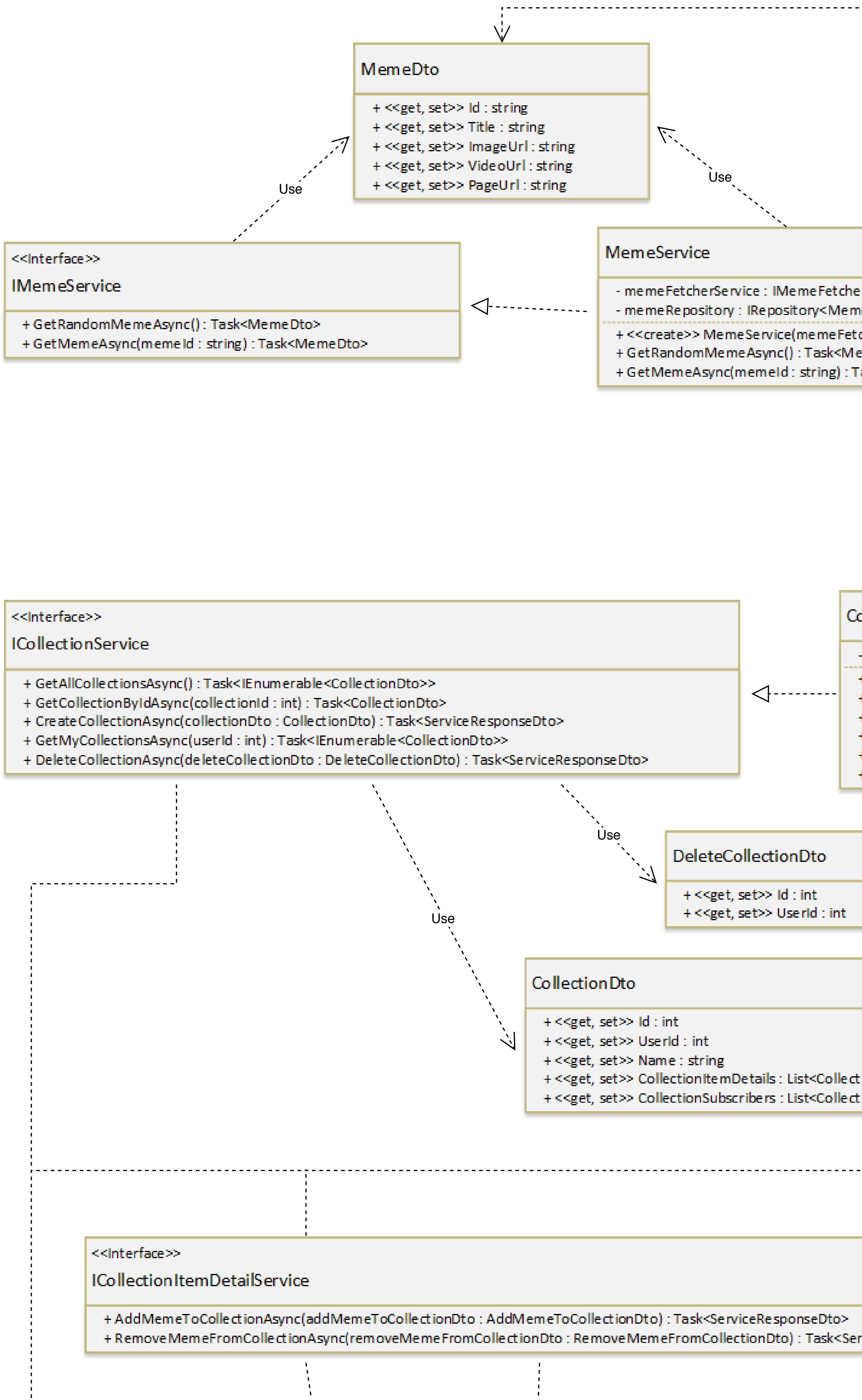
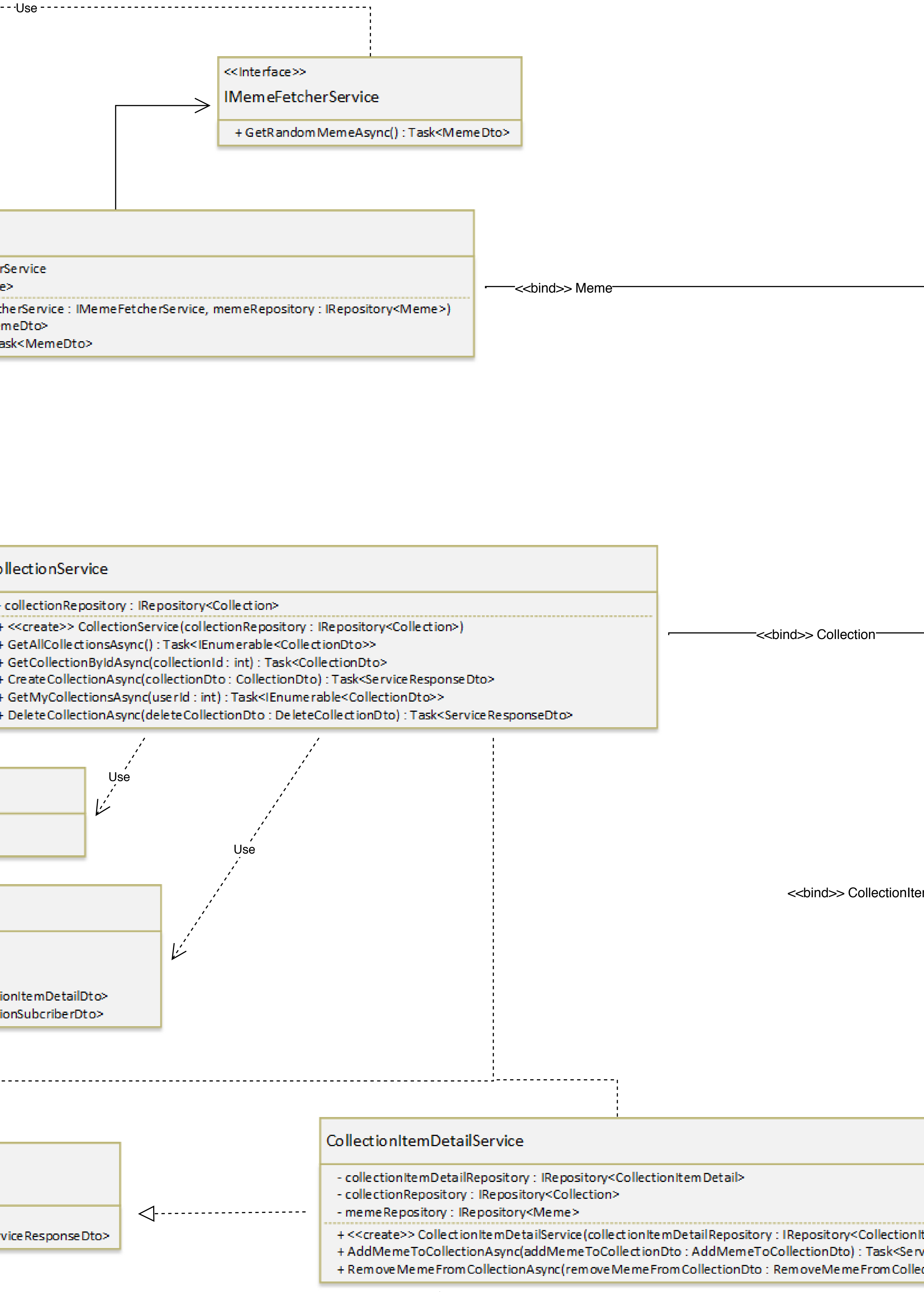
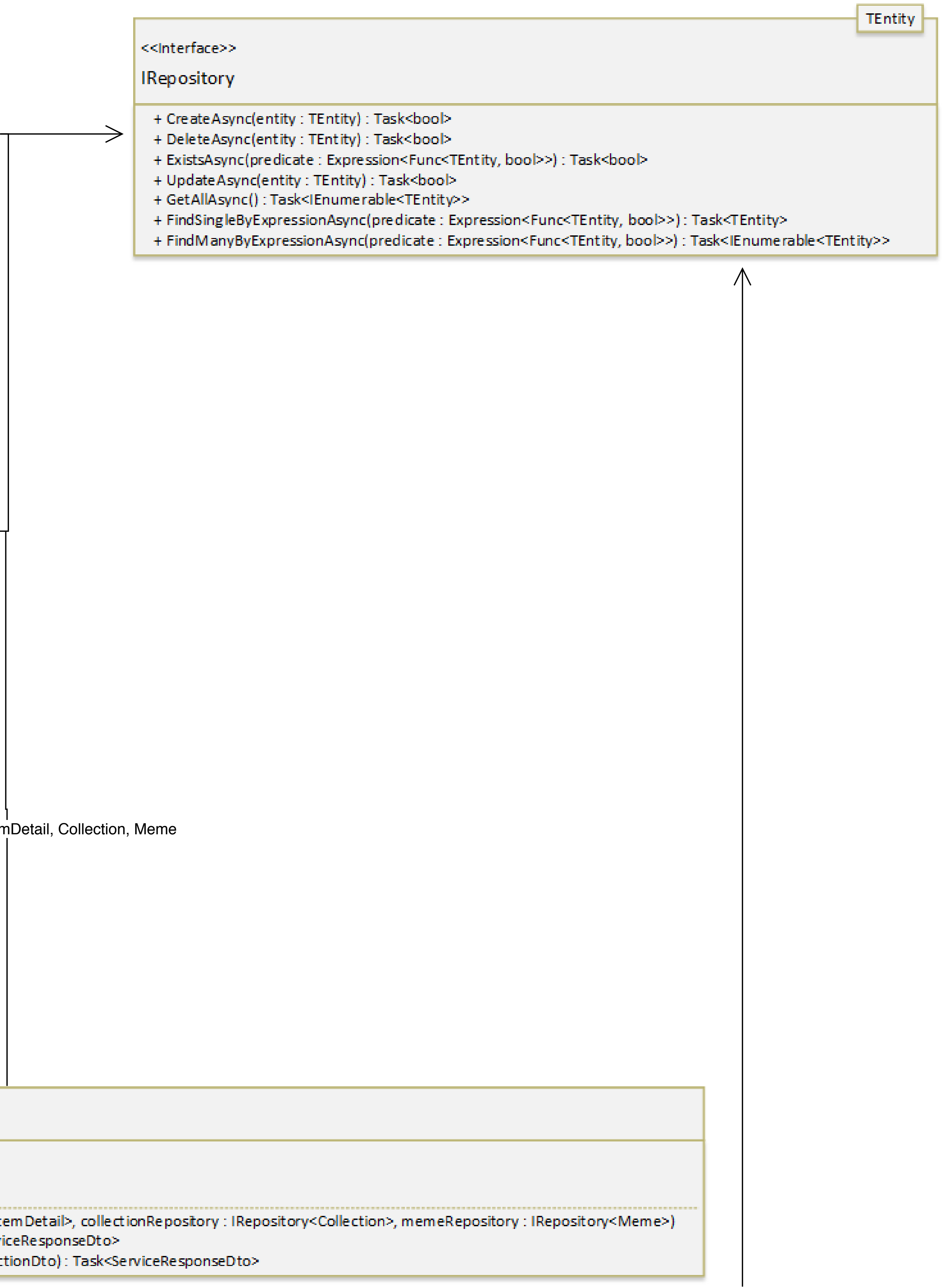
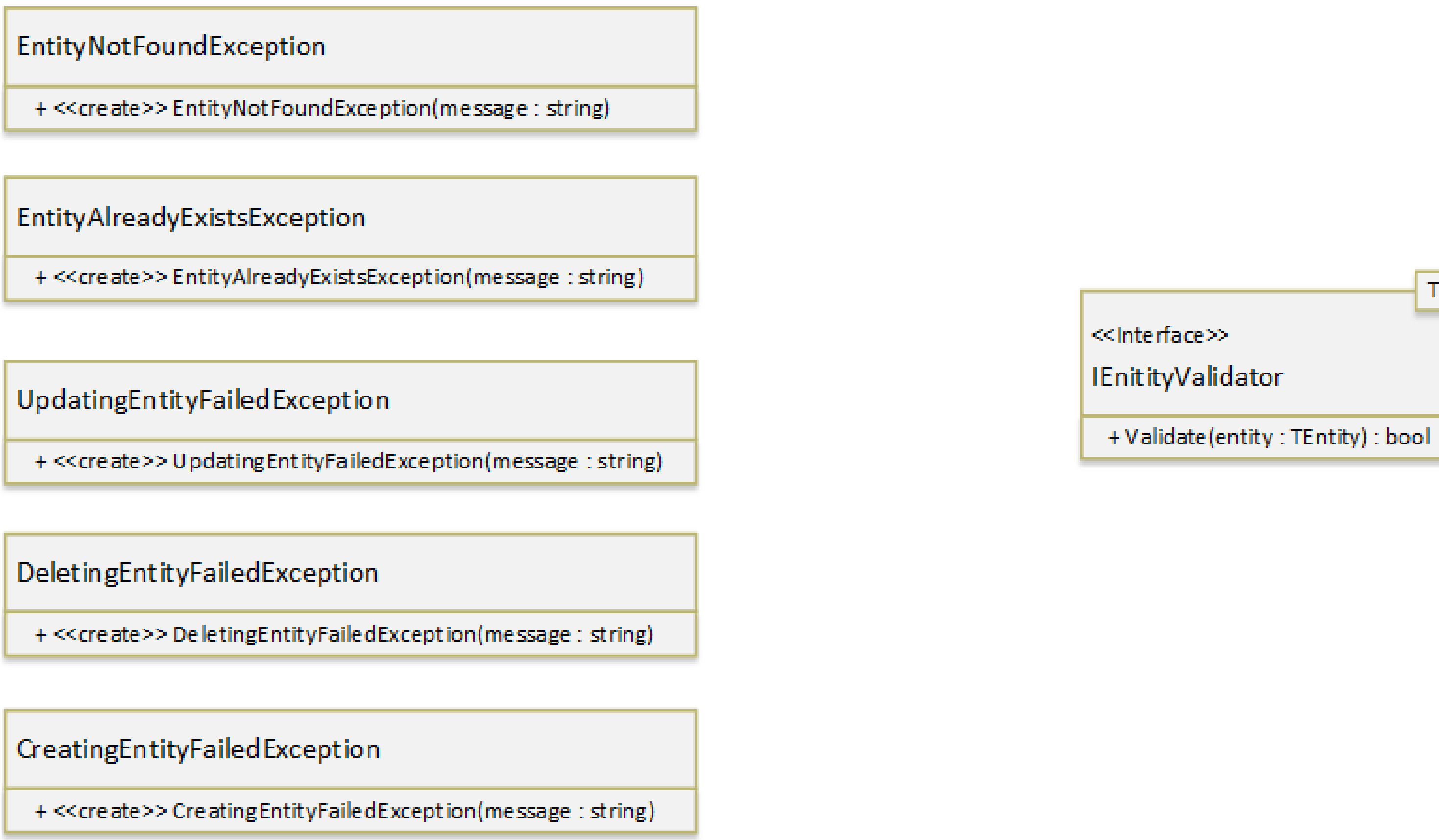
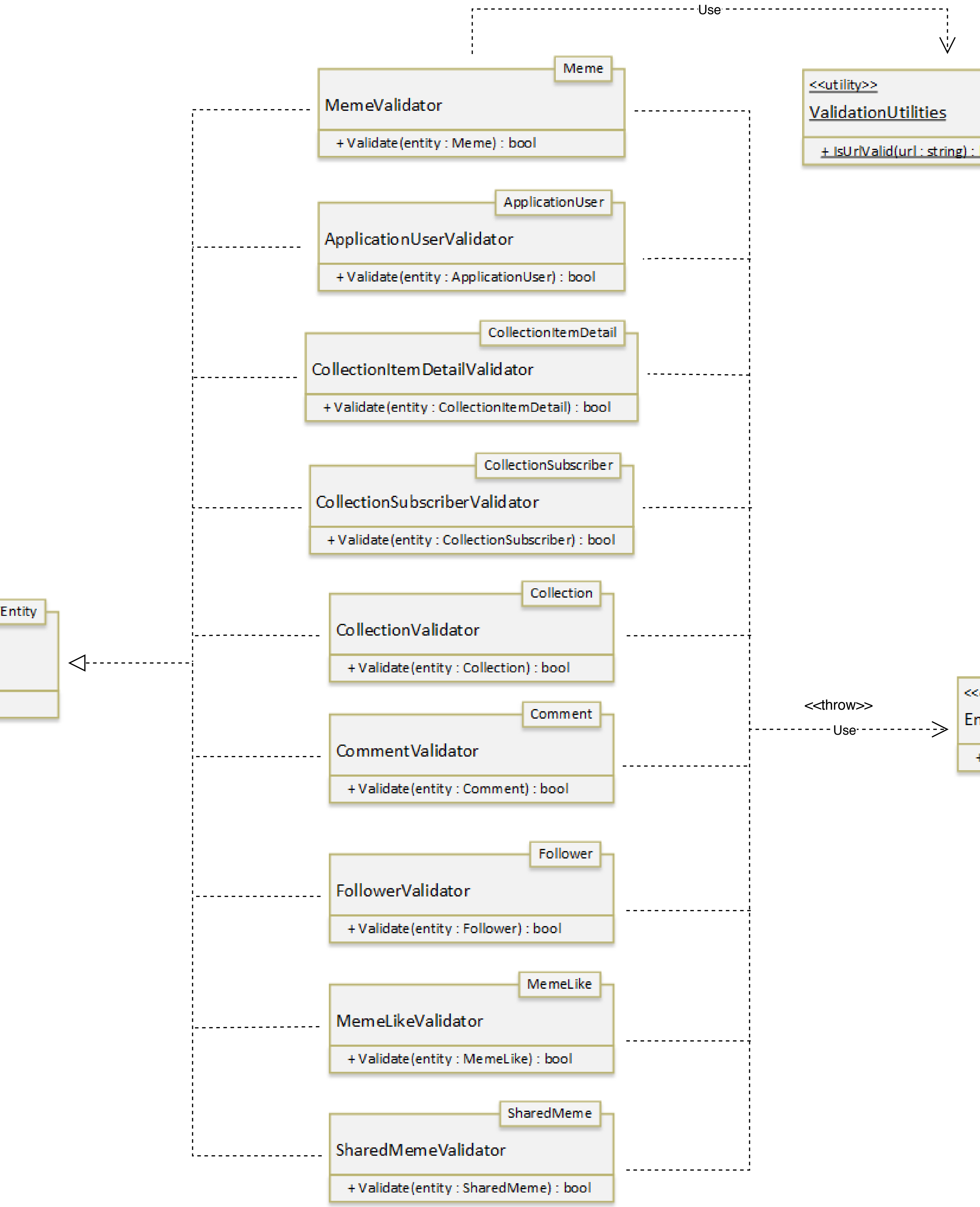

MemeDto
+ <<get, set>> Id : string
+ <<get, set>> Title : string
+ <<get, set>> ImageUrl : string
+ <<get, set>> VideoUrl : string
+ <<get, set>> PageUrl : string
Use
Use
<<Interface>>
IMemeService
+ GetRandomMemeAsync() : Task<MemeDto>
+ GetMemeAsync(memeId : string) : Task<MemeDto>
MemeService
- memeFetcherService : IMemeFetche
- memeRepository : IRepository<Mem
+ <<create>> MemeService(memeFetc
+ GetRandomMemeAsync() : Task<Me
+ GetMemeAsync(memeId : string) : Ta
<<Interface>>
ICollectionService
+ GetAllCollectionsAsync() : Task<IEnumerable<CollectionDto>>
+ GetCollectionByIdAsync(collectionId : int) : Task<CollectionDto>
+ CreateCollectionAsync(collectionDto : CollectionDto) : Task<ServiceResponseDto>
+ GetMyCollectionsAsync(userId : int) : Task<IEnumerable<CollectionDto>>
+ DeleteCollectionAsync(deleteCollectionDto : DeleteCollectionDto) : Task<ServiceResponseDto>
Co
Use
DeleteCollectionDto
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
Use
CollectionDto
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
+ <<get, set>> Name : string
+ <<get, set>> CollectionItemDetails : List<Collect
+ <<get, set>> CollectionSubscribers : List<Collect
<<Interface>>
ICollectionItemDetailService
+ AddMemeToCollectionAsync(addMemeToCollectionDto : AddMemeToCollectionDto) : Task<ServiceResponseDto>
+ RemoveMemeFromCollectionAsync(removeMemeFromCollectionDto : RemoveMemeFromCollectionDto) : Task<Ser

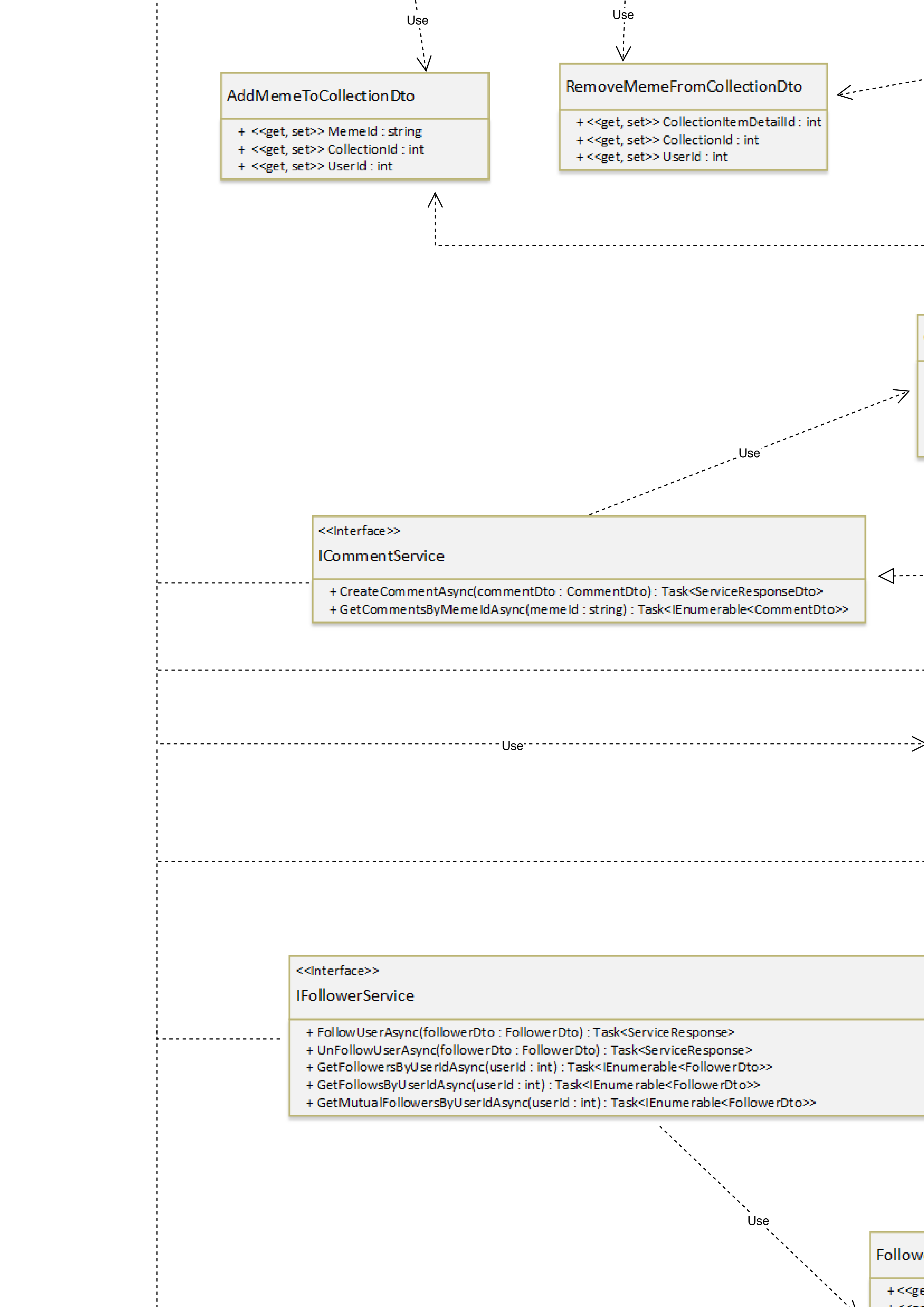

Use
<<Interface>>
IMemeFetcherService
+ GetRandomMemeAsync() : Task<MemeDto>
rService
e>
cherService : IMemeFetcherService, memeRepository : IRepository<Meme>)
meDto>
ask<MemeDto>
<<bind>> Meme
llectionService
- collectionRepository : IRepository<Collection>
+ <<create>> CollectionService(collectionRepository : IRepository<Collection>)
+ GetAllCollectionsAsync() : Task<IEnumerable<CollectionDto>>
+ GetCollectionByIdAsync(collectionId : int) : Task<CollectionDto>
+ CreateCollectionAsync(collectionDto : CollectionDto) : Task<ServiceResponseDto>
+ GetMyCollectionsAsync(userId : int) : Task<IEnumerable<CollectionDto>>
+ DeleteCollectionAsync(deleteCollectionDto : DeleteCollectionDto) : Task<ServiceResponseDto>
<<bind>> Collection
Use
Use
<<bind>> CollectionIter
ionItemDetailDto>
ionSubcriberDto>
viceResponseDto>
CollectionItemDetailService
- collectionItemDetailRepository : IRepository<CollectionItemDetail>
- collectionRepository : IRepository<Collection>
- memeRepository : IRepository<Meme>
+ <<create>> CollectionItemDetailService(collectionItemDetailRepository : IRepository<CollectionIt
+ AddMemeToCollectionAsync(addMemeToCollectionDto : AddMemeToCollectionDto) : Task<Serv
+ RemoveMemeFromCollectionAsync(removeMemeFromCollectionDto : RemoveMemeFromCollec

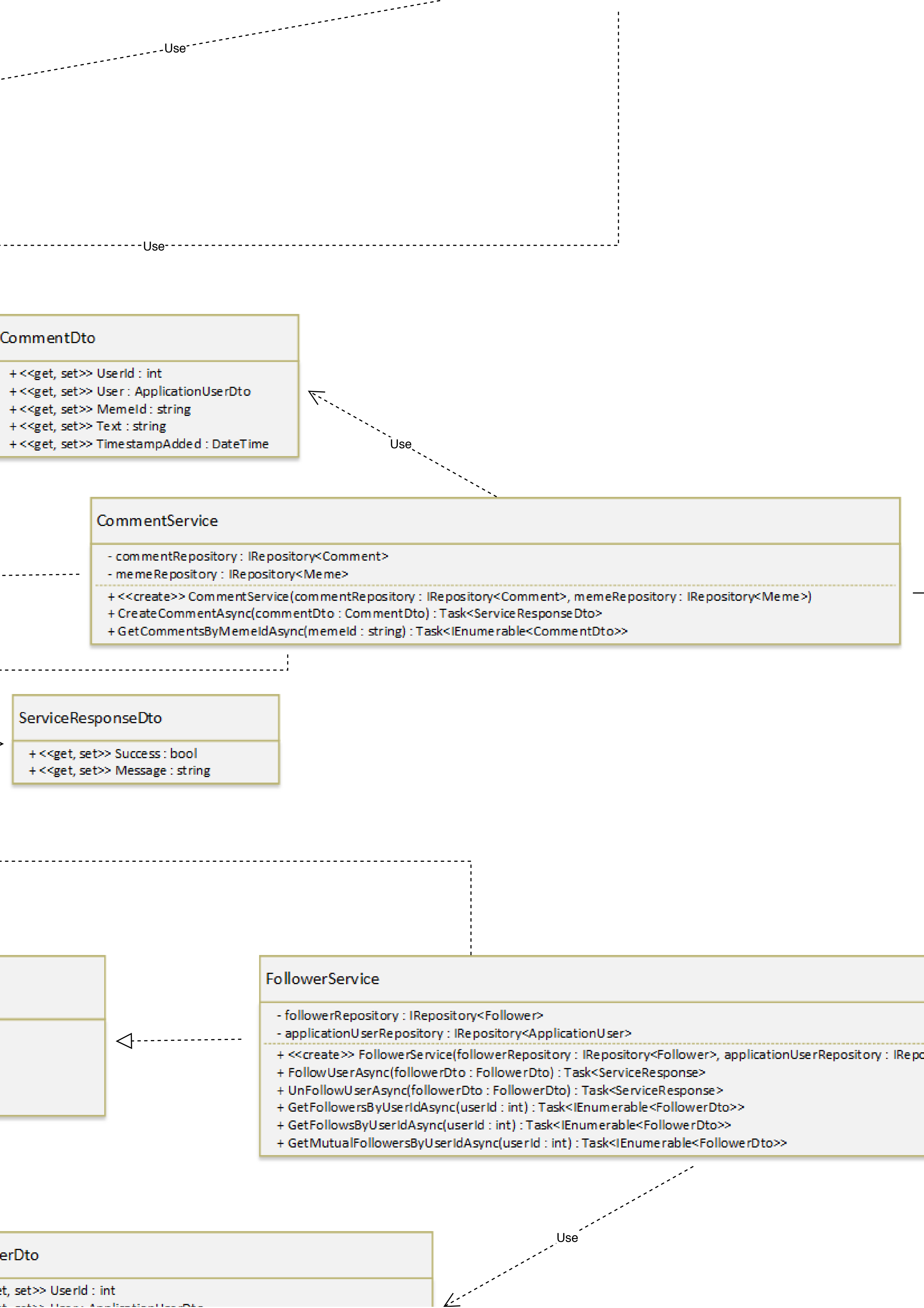

TEntity
<<Interface>>
IRepository
+ CreateAsync(entity : TEntity) : Task<bool>
+ DeleteAsync(entity : TEntity) : Task<bool>
+ ExistsAsync(predicate : Expression<Func<TEntity, bool>>) : Task<bool>
+ UpdateAsync(entity : TEntity) : Task<bool>
+ GetAllAsync() : Task<IEnumerable<TEntity>>
+ FindSingleByExpressionAsync(predicate : Expression<Func<TEntity, bool>>) : Task<TEntity>
+ FindManyByExpressionAsync(predicate : Expression<Func<TEntity, bool>>) : Task<IEnumerable<TEntity>>
mDetail, Collection, Meme
temDetail>, collectionRepository : IRepository<Collection>, memeRepository : IRepository<Meme>)
viceResponseDto>
ctionDto) : Task<ServiceResponseDto>

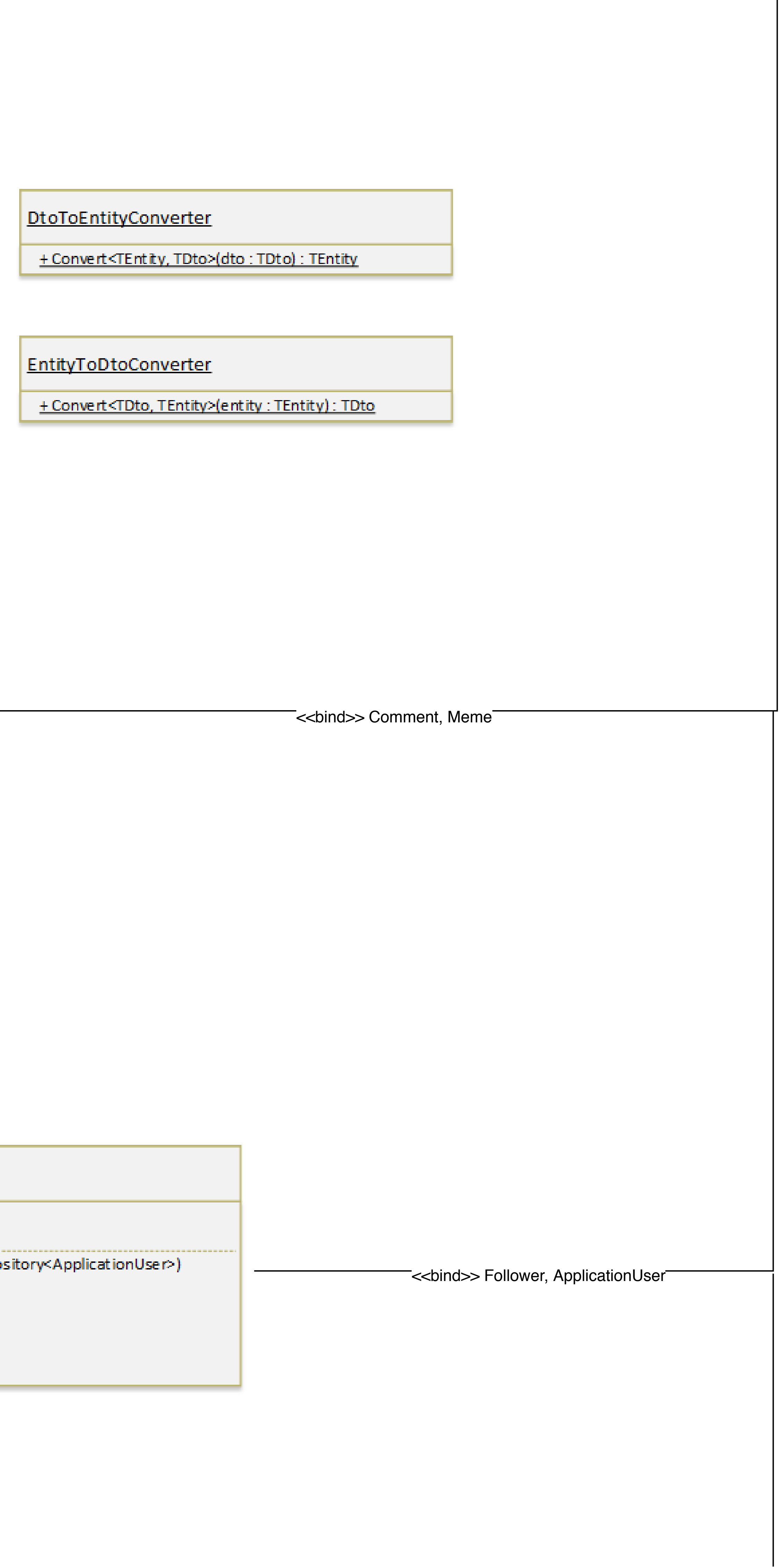

EntityNotFoundException
+ <<create>> EntityNotFoundException(message : string)
EntityAlreadyExistsException
+ <<create>> EntityAlreadyExistsException(message : string)
UpdatingEntityFailedException
+ <<create>> UpdatingEntityFailedException(message : string)
DeletingEntityFailedException
+ <<create>> DeletingEntityFailedException(message : string)
CreatingEntityFailedException
+ <<create>> CreatingEntityFailedException(message : string)
T
<<Interface>>
IEnitityValidator
+ Validate(entity : TEntity) : bool

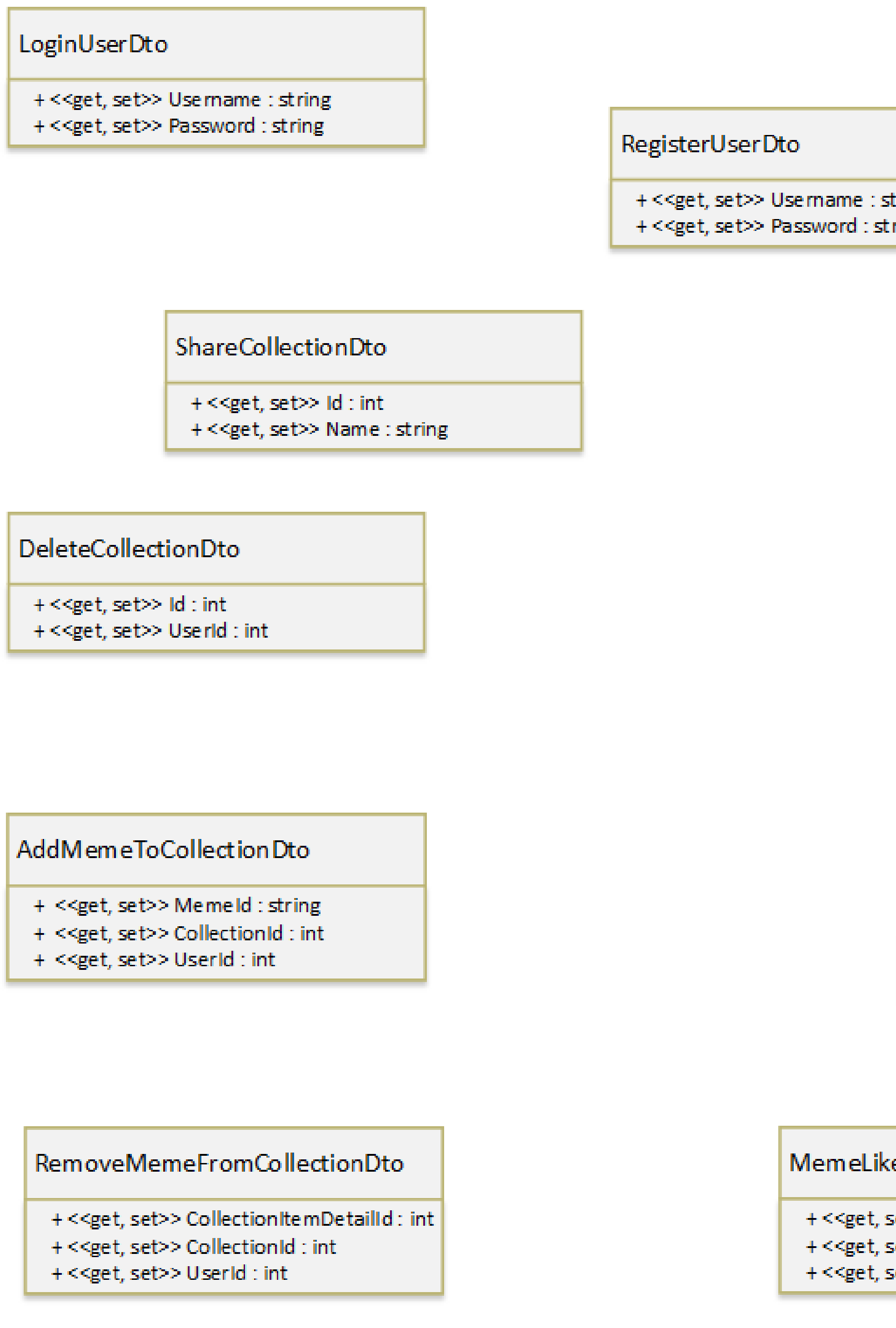

Use
Meme
MemeValidator
+ Validate(entity : Meme) : bool
<<utility>>
ValidationUtilities
+ IsUrlValid(url : string) :
ApplicationUser
ApplicationUserValidator
+ Validate(entity : ApplicationUser) : bool
CollectionItemDetail
CollectionItemDetailValidator
+ Validate(entity : CollectionItemDetail) : bool
CollectionSubscriber
CollectionSubscriberValidator
+ Validate(entity : CollectionSubscriber) : bool
Entity
Collection
CollectionValidator
+ Validate(entity : Collection) : bool
<<throw>>
Use
<<
En
+
Comment
CommentValidator
+ Validate(entity : Comment) : bool
Follower
FollowerValidator
+ Validate(entity : Follower) : bool
MemeLike
MemeLikeValidator
+ Validate(entity : MemeLike) : bool
SharedMeme
SharedMemeValidator
+ Validate(entity : SharedMeme) : bool

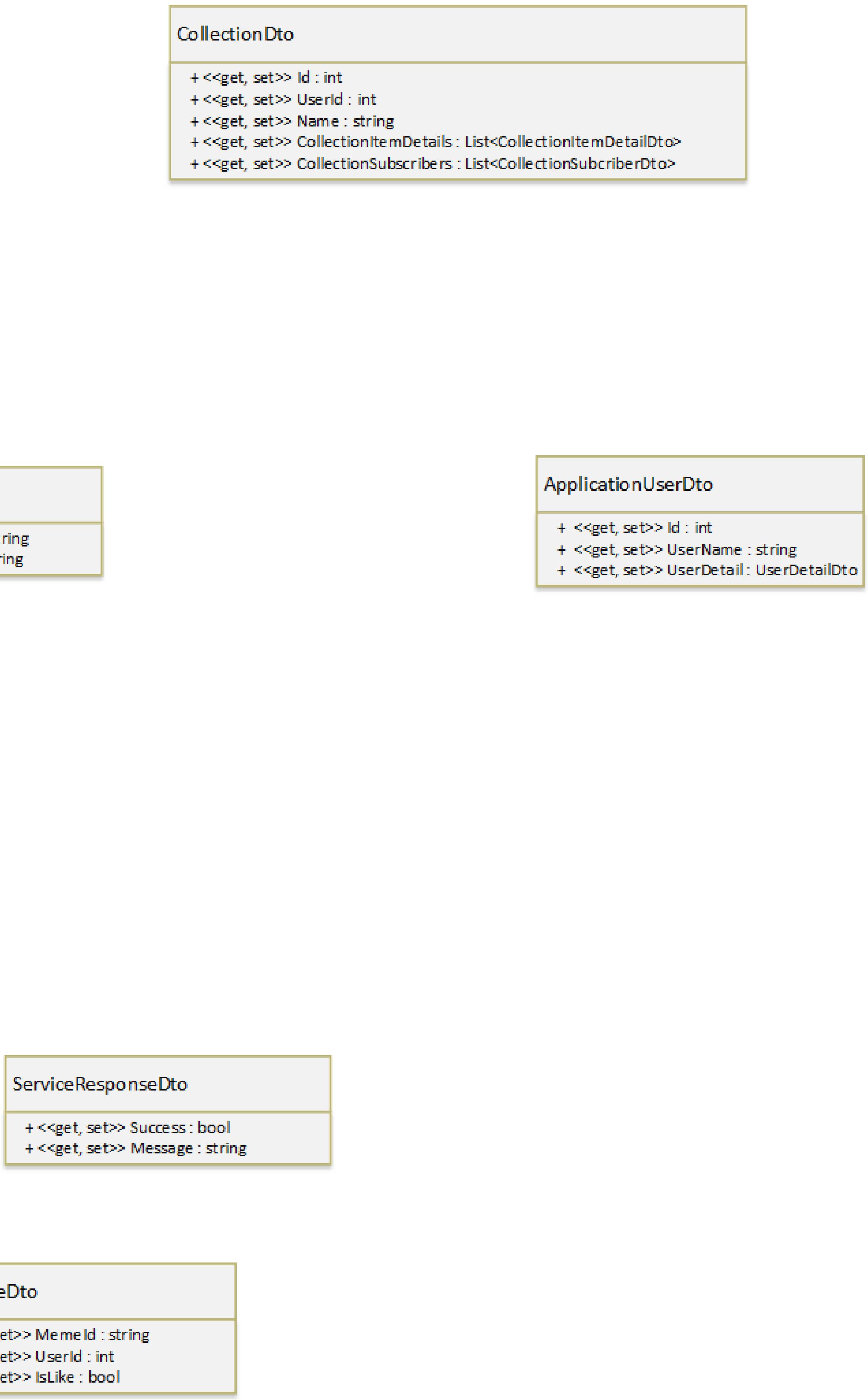
bool
exception>>
ntityValidationException
+ <<create>> EntityValidationException(message : string)


Use
Use
AddMemeToCollectionDto
+ <<get, set>> MemeId : string
+ <<get, set>> CollectionId : int
+ <<get, set>> UserId : int
RemoveMemeFromCollectionDto
+ <<get, set>> CollectionItemDetailId : int
+ <<get, set>> CollectionId : int
+ <<get, set>> UserId : int
Use
<<Interface>>
ICommentService
+ CreateCommentAsync(commentDto : CommentDto) : Task<ServiceResponseDto>
+ GetCommentsByMemeIdAsync(memeId : string) : Task<IEnumerable<CommentDto>>
Use
<<Interface>>
IFollowerService
+ FollowUserAsync(followerDto : FollowerDto) : Task<ServiceResponse>
+ UnFollowUserAsync(followerDto : FollowerDto) : Task<ServiceResponse>
+ GetFollowersByUserIdAsync(userId : int) : Task<IEnumerable<FollowerDto>>
+ GetFollowsByUserIdAsync(userId : int) : Task<IEnumerable<FollowerDto>>
+ GetMutualFollowersByUserIdAsync(userId : int) : Task<IEnumerable<FollowerDto>>
Use


Use
Use
CommentDto
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUserDto
+ <<get, set>> MemeId : string
+ <<get, set>> Text : string
+ <<get, set>> TimestampAdded : DateTime
Use
CommentService
- commentRepository : IRepository<Comment>
- memeRepository : IRepository<Meme>
+ <<create>> CommentService(commentRepository : IRepository<Comment>, memeRepository : IRepository<Meme>)
+ CreateCommentAsync(commentDto : CommentDto) : Task<ServiceResponseDto>
+ GetCommentsByMemeIdAsync(memeId : string) : Task<IEnumerable<CommentDto>>
ServiceResponseDto
+ <<get, set>> Success : bool
+ <<get, set>> Message : string
FollowerService
- followerRepository : IRepository<Follower>
- applicationUserRepository : IRepository<ApplicationUser>
+ <<create>> FollowerService(followerRepository : IRepository<Follower>, applicationUserRepository : IRepo
+ FollowUserAsync(followerDto : FollowerDto) : Task<ServiceResponse>
+ UnFollowUserAsync(followerDto : FollowerDto) : Task<ServiceResponse>
+ GetFollowersByUserIdAsync(userId : int) : Task<IEnumerable<FollowerDto>>
+ GetFollowsByUserIdAsync(userId : int) : Task<IEnumerable<FollowerDto>>
+ GetMutualFollowersByUserIdAsync(userId : int) : Task<IEnumerable<FollowerDto>>
Use
erDto
t, set>> UserId : int


DtoToEntityConverter
+ Convert<TEntity, TDto>(dto : TDto) : TEntity
EntityToDtoConverter
+ Convert<TDto, TEntity>(entity : TEntity) : TDto
<<bind>> Comment, Meme
sitory<ApplicationUser>)
<<bind>> Follower, ApplicationUser


LoginUserDto
+ <<get, set>> Username : string
+ <<get, set>> Password : string
RegisterUserDto
+ <<get, set>> Username : st
+ <<get, set>> Password : str
ShareCollectionDto
+ <<get, set>> Id : int
+ <<get, set>> Name : string
DeleteCollectionDto
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
AddMemeToCollectionDto
+ <<get, set>> MemeId : string
+ <<get, set>> CollectionId : int
+ <<get, set>> UserId : int
RemoveMemeFromCollectionDto
+ <<get, set>> CollectionItemDetailId : int
+ <<get, set>> CollectionId : int
+ <<get, set>> UserId : int
MemeLike
+ <<get, s
+ <<get, s
+ <<get, s



CollectionDto

```
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
+ <<get, set>> Name : string
+ <<get, set>> CollectionItemDetails : List<CollectionItemDetailDto>
+ <<get, set>> CollectionSubscribers : List<CollectionSubcriberDto>
```

```
ring
ring
```

ApplicationUserDto

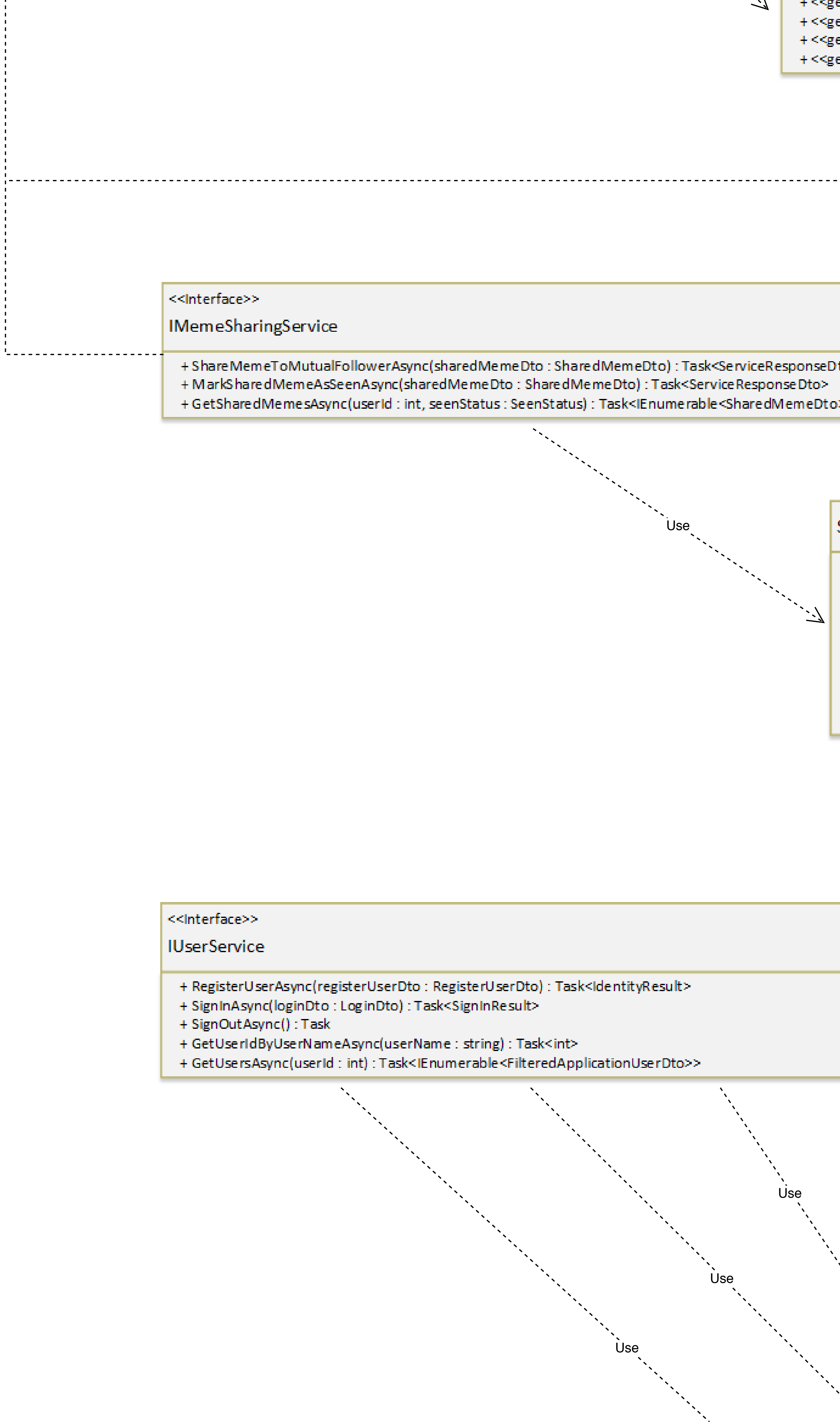
```
+ <<get, set>> Id : int
+ <<get, set>> UserName : string
+ <<get, set>> UserDetail : UserDetailDto
```

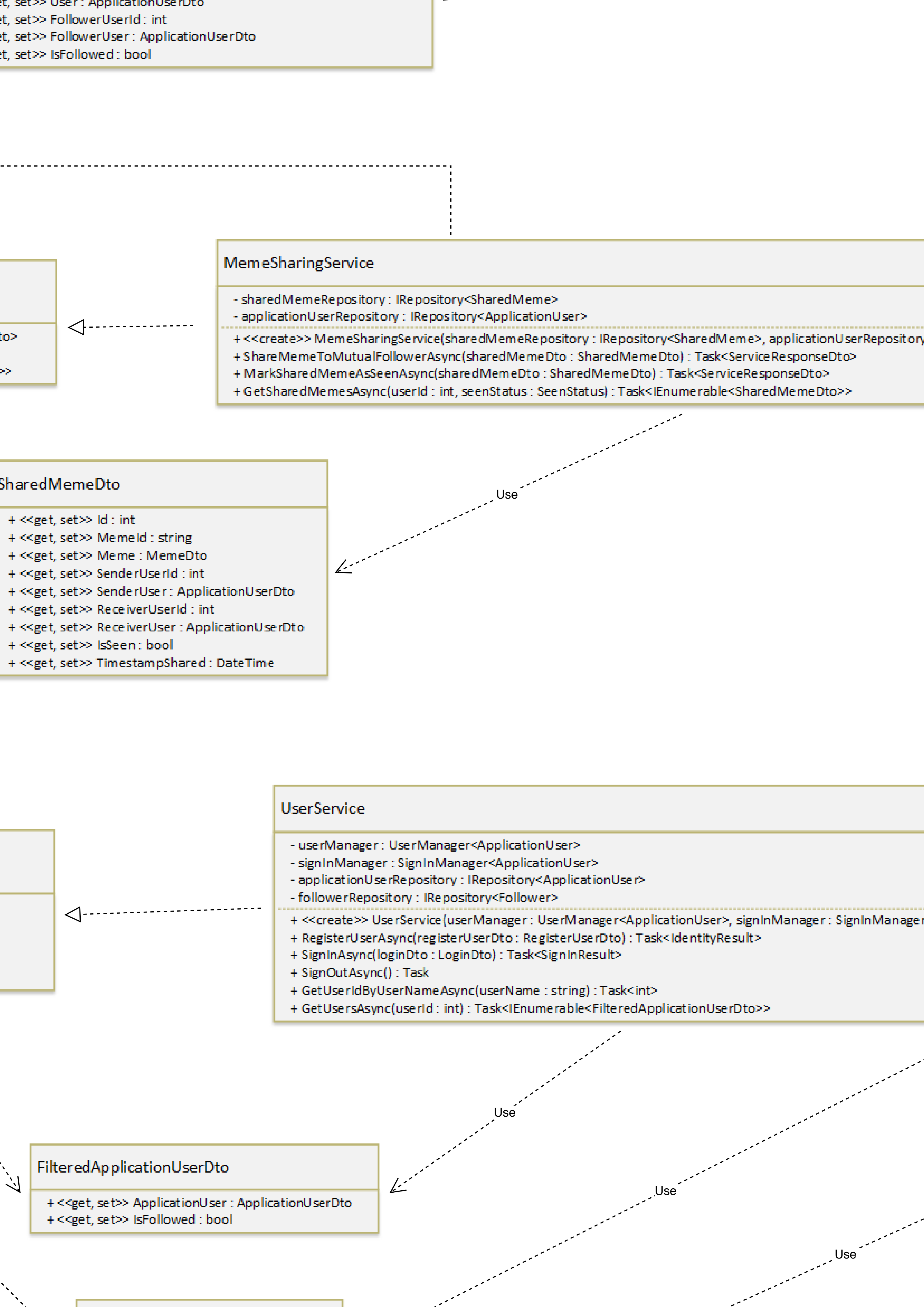
ServiceResponseDto

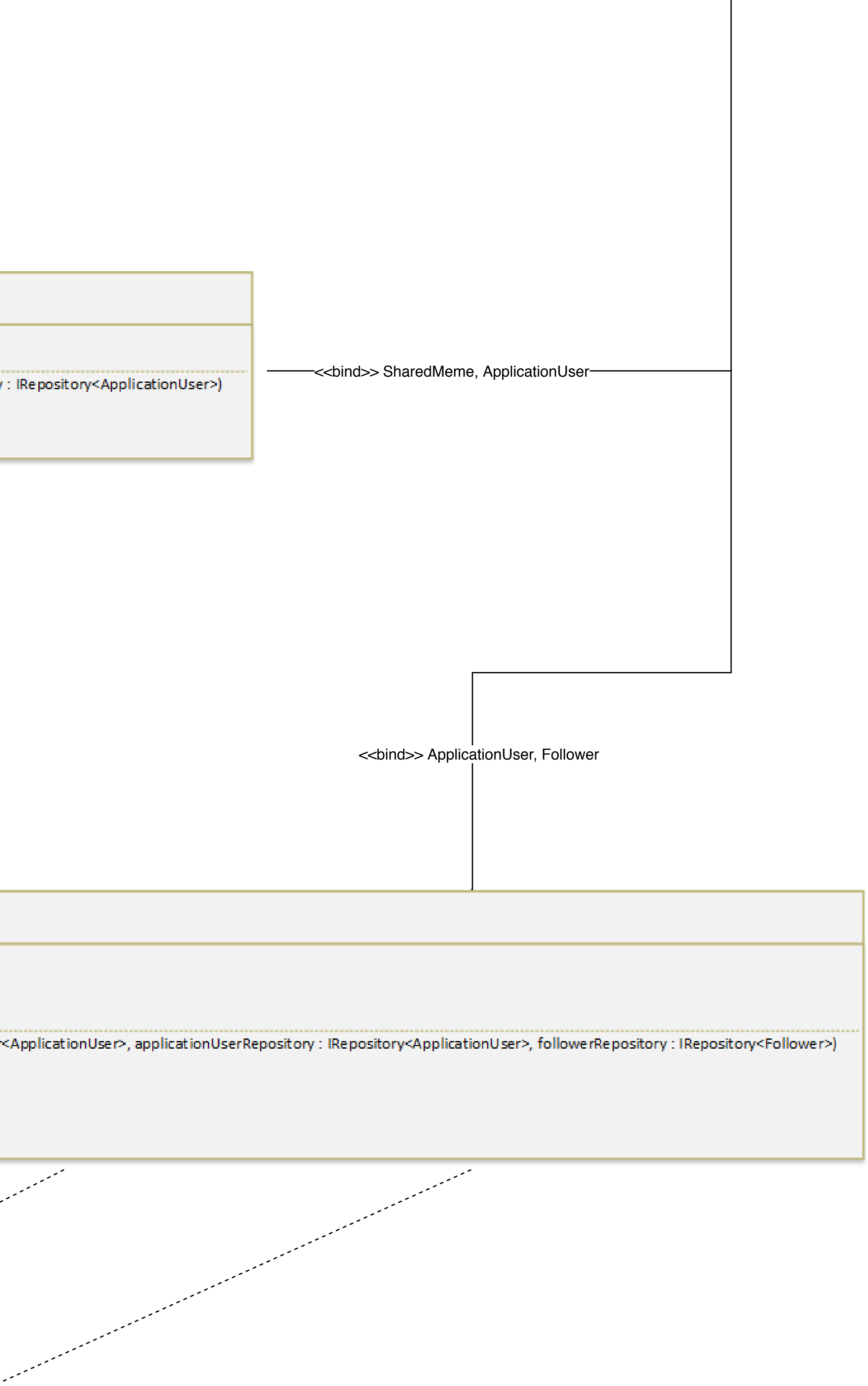
```
+ <<get, set>> Success : bool
+ <<get, set>> Message : string
```

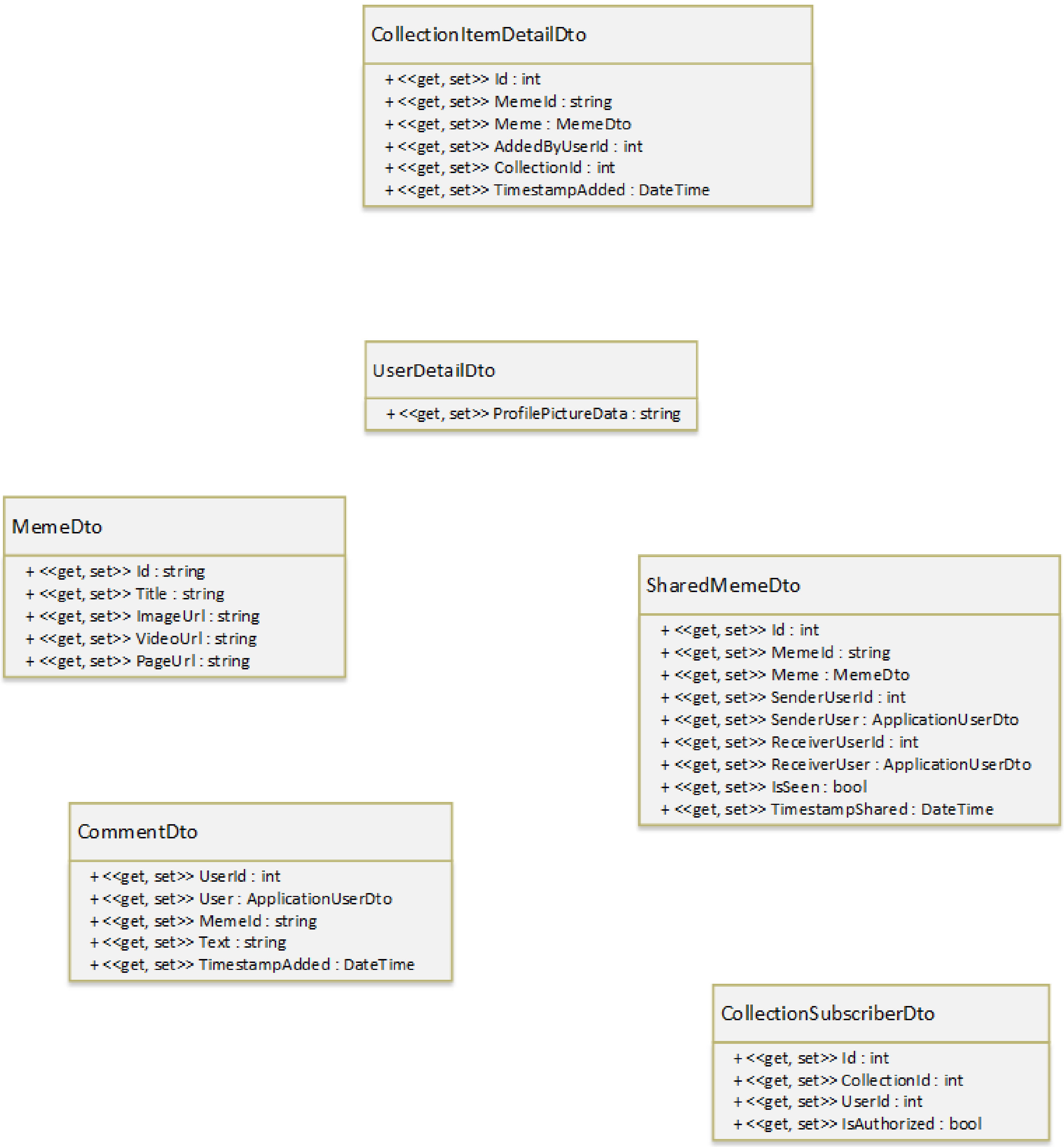
eDto

```
et>> MemeId : string
et>> UserId : int
et>> IsLike : bool
```


+ <<ge
+ <<ge
+ <<ge
+ <<ge
<<Interface>>
IMemeSharingService
+ ShareMemeToMutualFollowerAsync(sharedMemeDto : SharedMemeDto) : Task<ServiceResponseDt
+ MarkSharedMemeAsSeenAsync(sharedMemeDto : SharedMemeDto) : Task<ServiceResponseDto>
+ GetSharedMemesAsync(userId : int, seenStatus : SeenStatus) : Task<IEnumerable<SharedMemeDto
Use
S
<<Interface>>
IUserService
+ RegisterUserAsync(registerUserDto : RegisterUserDto) : Task<IdentityResult>
+ SignInAsync(loginDto : LoginDto) : Task<SignInResult>
+ SignOutAsync() : Task
+ GetUserIdByUserNameAsync(userName : string) : Task<int>
+ GetUsersAsync(userId : int) : Task<IEnumerable<FilteredApplicationUserDto>>
Use
Use
Use


et, set>> User : ApplicationUserDto
et, set>> FollowerUserId : int
et, set>> FollowerUser : ApplicationUserDto
et, set>> IsFollowed : bool
MemeSharingService
- sharedMemeRepository : IRepository<SharedMeme>
- applicationUserRepository : IRepository<ApplicationUser>
+ <<create>> MemeSharingService(sharedMemeRepository : IRepository<SharedMeme>, applicationUserRepository
+ ShareMemeToMutualFollowerAsync(sharedMemeDto : SharedMemeDto) : Task<ServiceResponseDto>
+ MarkSharedMemeAsSeenAsync(sharedMemeDto : SharedMemeDto) : Task<ServiceResponseDto>
+ GetSharedMemesAsync(userId : int, seenStatus : SeenStatus) : Task<IEnumerable<SharedMemeDto>>
to>
>>
Use
SharedMemeDto
+ <<get, set>> Id : int
+ <<get, set>> MemeId : string
+ <<get, set>> Meme : MemeDto
+ <<get, set>> SenderUserId : int
+ <<get, set>> SenderUser : ApplicationUserDto
+ <<get, set>> ReceiverUserId : int
+ <<get, set>> ReceiverUser : ApplicationUserDto
+ <<get, set>> IsSeen : bool
+ <<get, set>> TimestampShared : DateTime
UserService
- userManager : UserManager<ApplicationUser>
- signInManager : SignInManager<ApplicationUser>
- applicationUserRepository : IRepository<ApplicationUser>
- followerRepository : IRepository<Follower>
+ <<create>> UserService(userManager : UserManager<ApplicationUser>, signInManager : SignInManager
+ RegisterUserAsync(registerUserDto : RegisterUserDto) : Task<IdentityResult>
+ SignInAsync(loginDto : LoginDto) : Task<SignInResult>
+ SignOutAsync() : Task
+ GetUserIdByUserNameAsync(userName : string) : Task<int>
+ GetUsersAsync(userId : int) : Task<IEnumerable<FilteredApplicationUserDto>>
Use
FilteredApplicationUserDto
+ <<get, set>> ApplicationUser : ApplicationUserDto
+ <<get, set>> IsFollowed : bool
Use
Use


y : IRepository<ApplicationUser>)
<<bind>> SharedMeme, ApplicationUser
<<bind>> ApplicationUser, Follower
<ApplicationUser>, applicationUserRepository : IRepository<ApplicationUser>, followerRepository : IRepository<Follower>)



```
CollectionItemDetailDto
+ <<get, set>> Id : int
+ <<get, set>> MemeId : string
+ <<get, set>> Meme : MemeDto
+ <<get, set>> AddedByUserId : int
+ <<get, set>> CollectionId : int
+ <<get, set>> TimestampAdded : DateTime
```

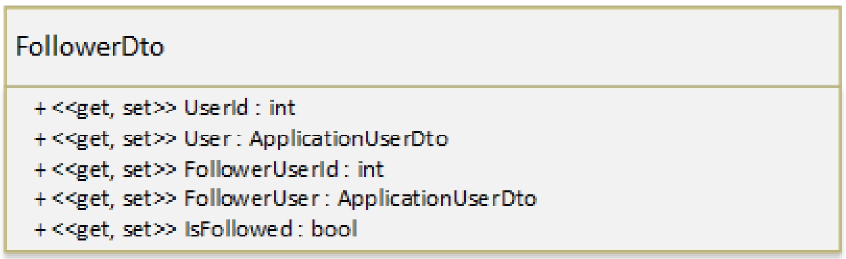
```
UserDetailDto
+ <<get, set>> ProfilePictureData : string
```

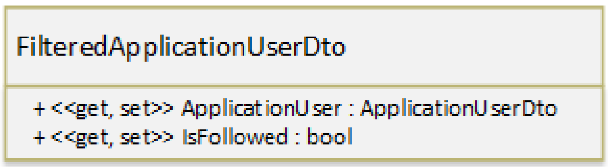
```
MemeDto
+ <<get, set>> Id : string
+ <<get, set>> Title : string
+ <<get, set>> ImageUrl : string
+ <<get, set>> VideoUrl : string
+ <<get, set>> PageUrl : string
```

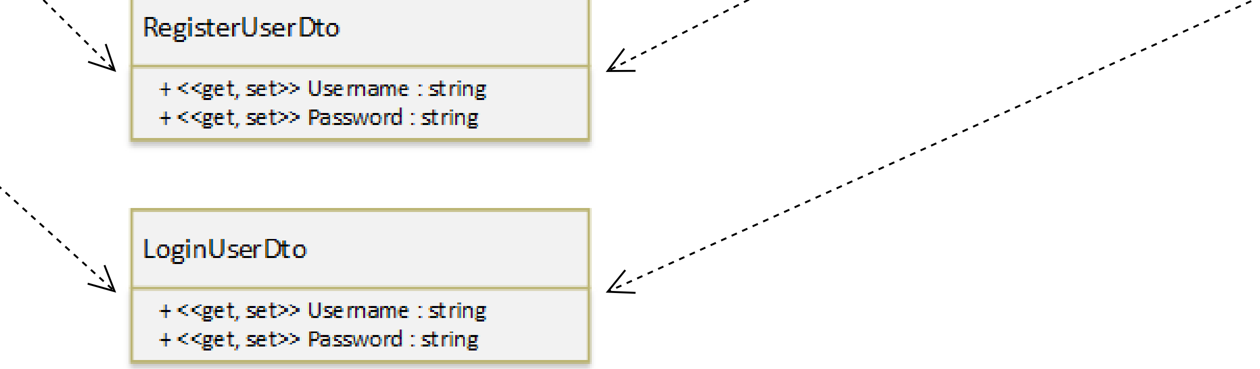
```
SharedMemeDto
+ <<get, set>> Id : int
+ <<get, set>> MemeId : string
+ <<get, set>> Meme : MemeDto
+ <<get, set>> SenderUserId : int
+ <<get, set>> SenderUser : ApplicationUserDto
+ <<get, set>> ReceiverUserId : int
+ <<get, set>> ReceiverUser : ApplicationUserDto
+ <<get, set>> IsSeen : bool
+ <<get, set>> TimestampShared : DateTime
```

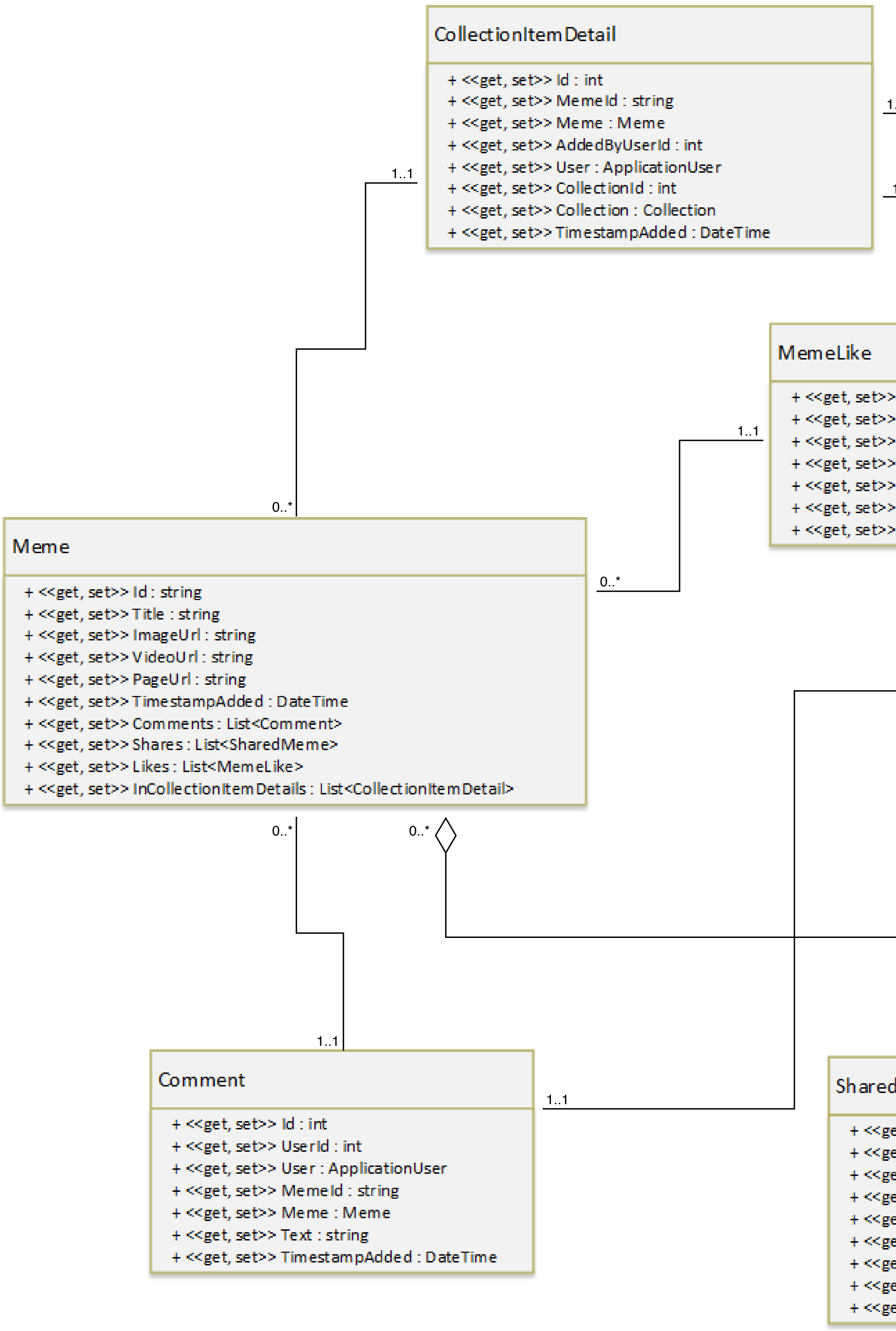
```
CommentDto
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUserDto
+ <<get, set>> MemeId : string
+ <<get, set>> Text : string
+ <<get, set>> TimestampAdded : DateTime
```

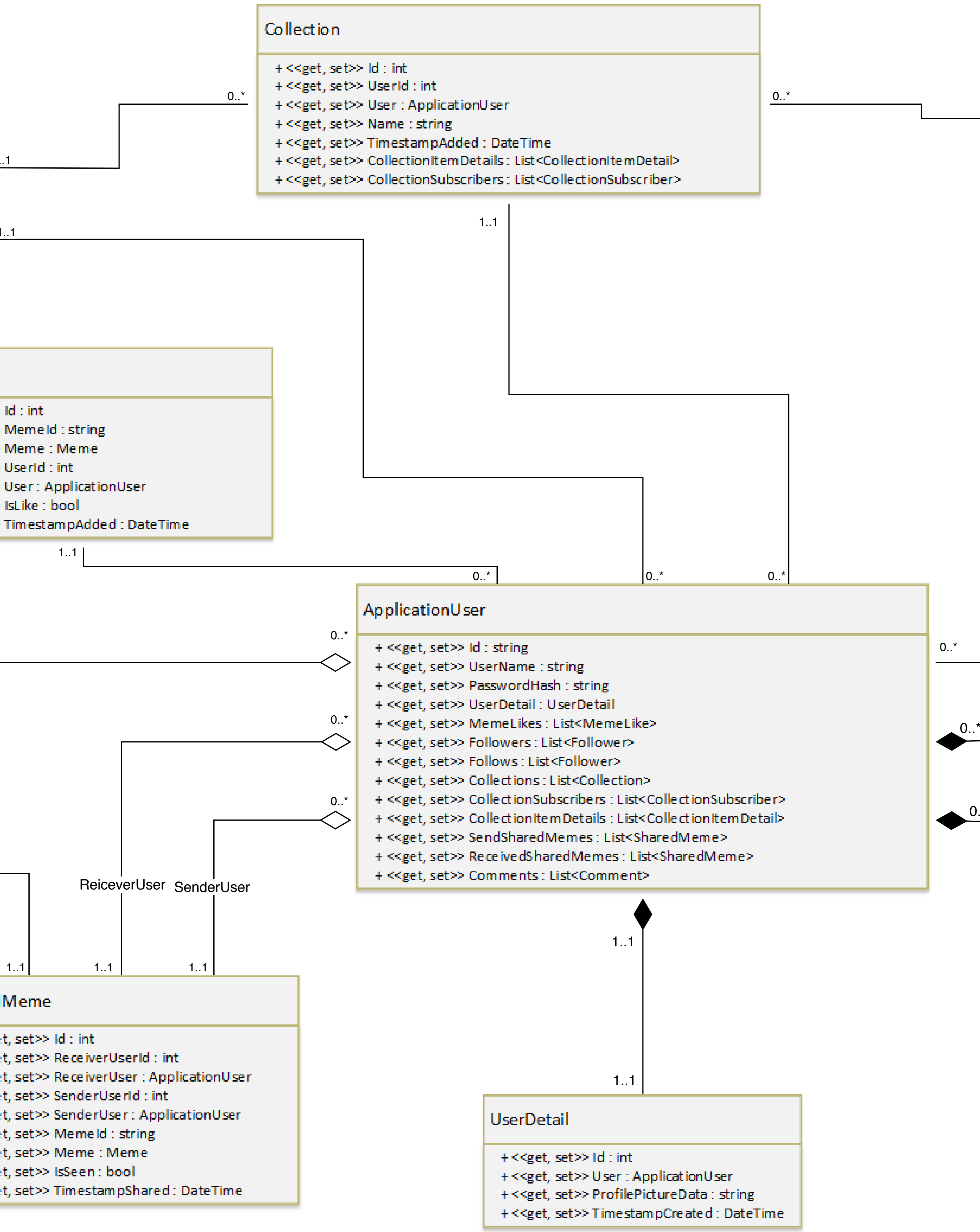
```
CollectionSubscriberDto
+ <<get, set>> Id : int
+ <<get, set>> CollectionId : int
+ <<get, set>> UserId : int
+ <<get, set>> IsAuthorized : bool
```

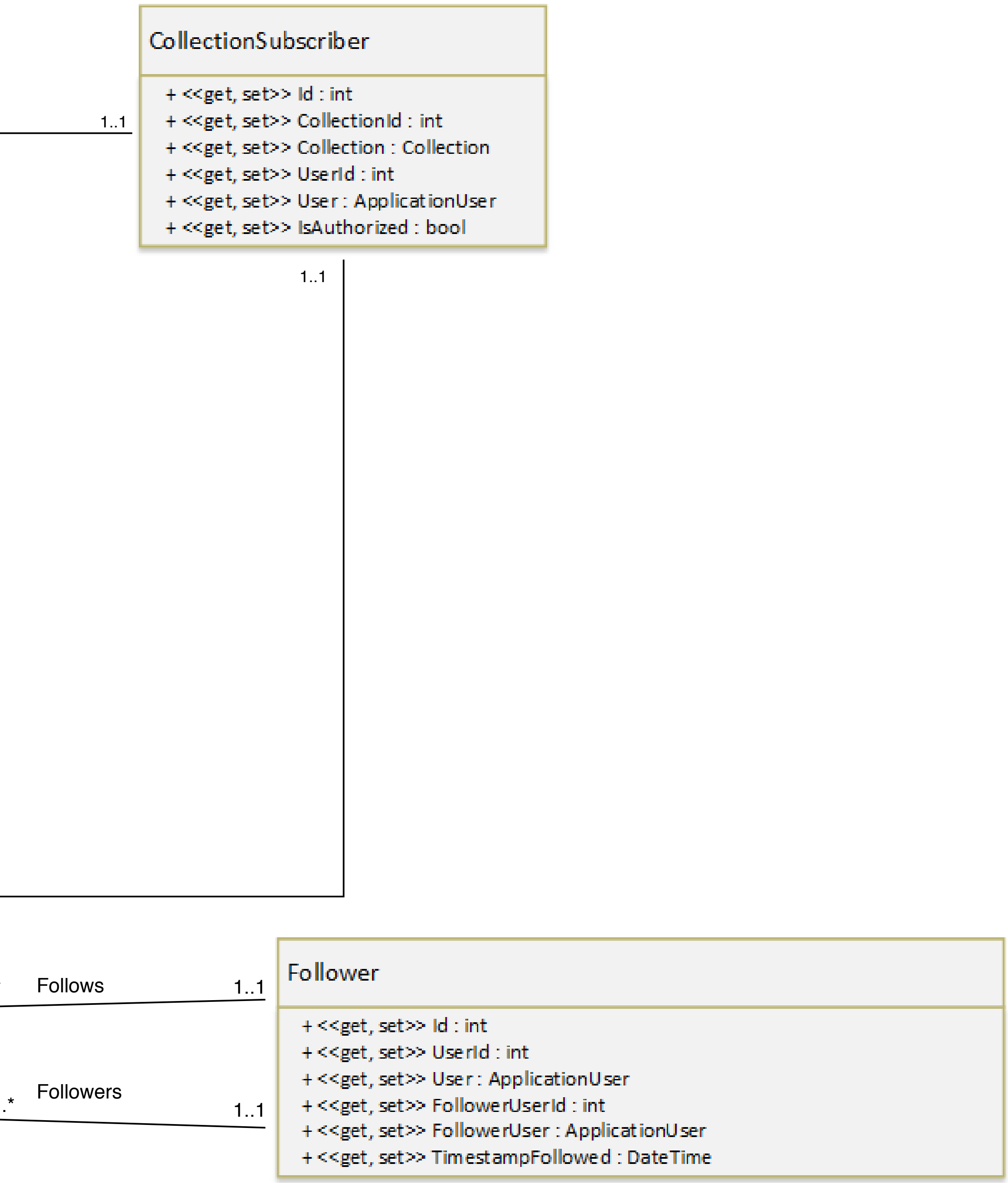

FollowerDto
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUserDto
+ <<get, set>> FollowerUserId : int
+ <<get, set>> FollowerUser : ApplicationUserDto
+ <<get, set>> IsFollowed : bool

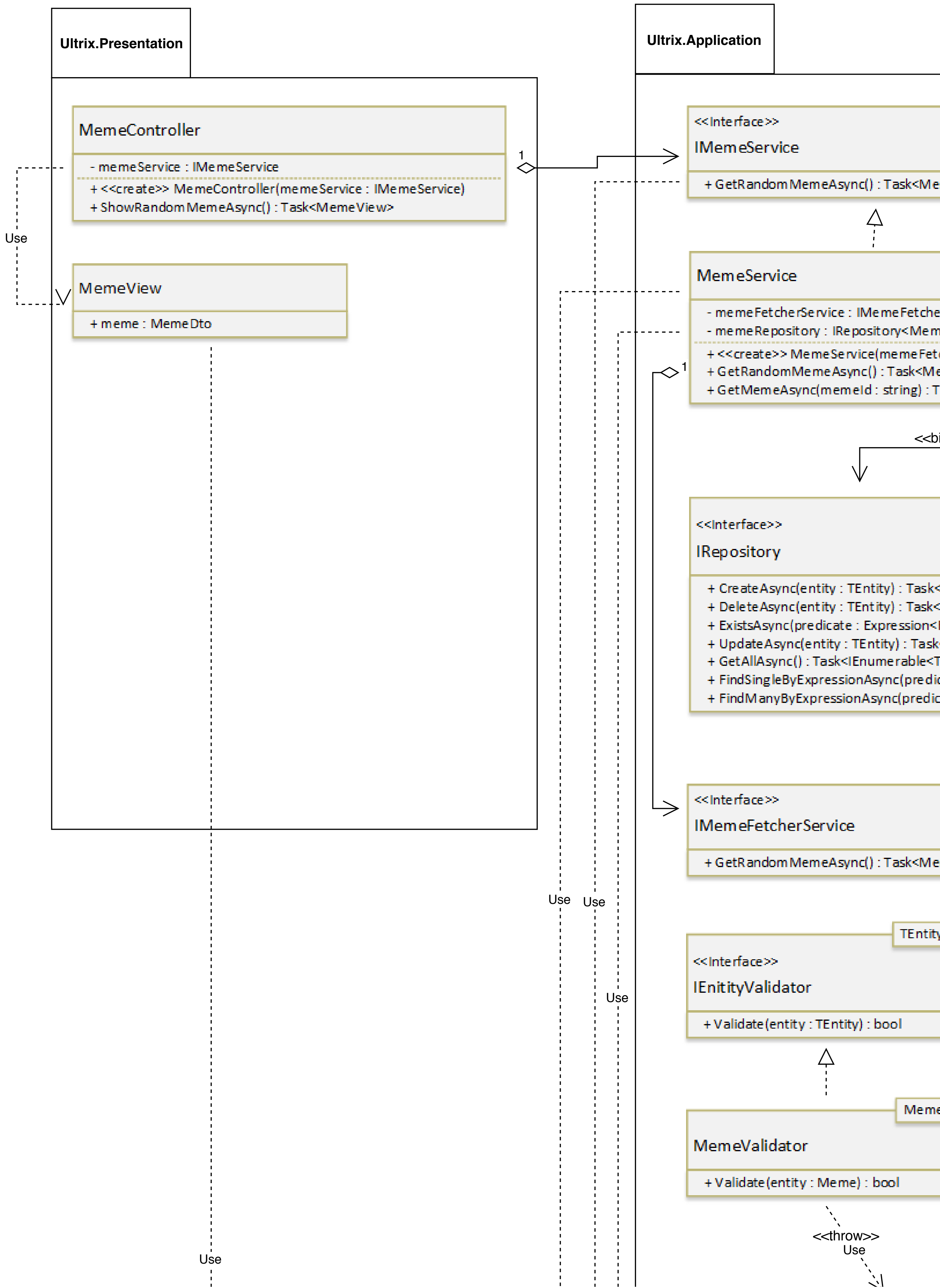

FilteredApplicationUserDto
+ <<get, set>> ApplicationUser : ApplicationUserDto
+ <<get, set>> IsFollowed : bool

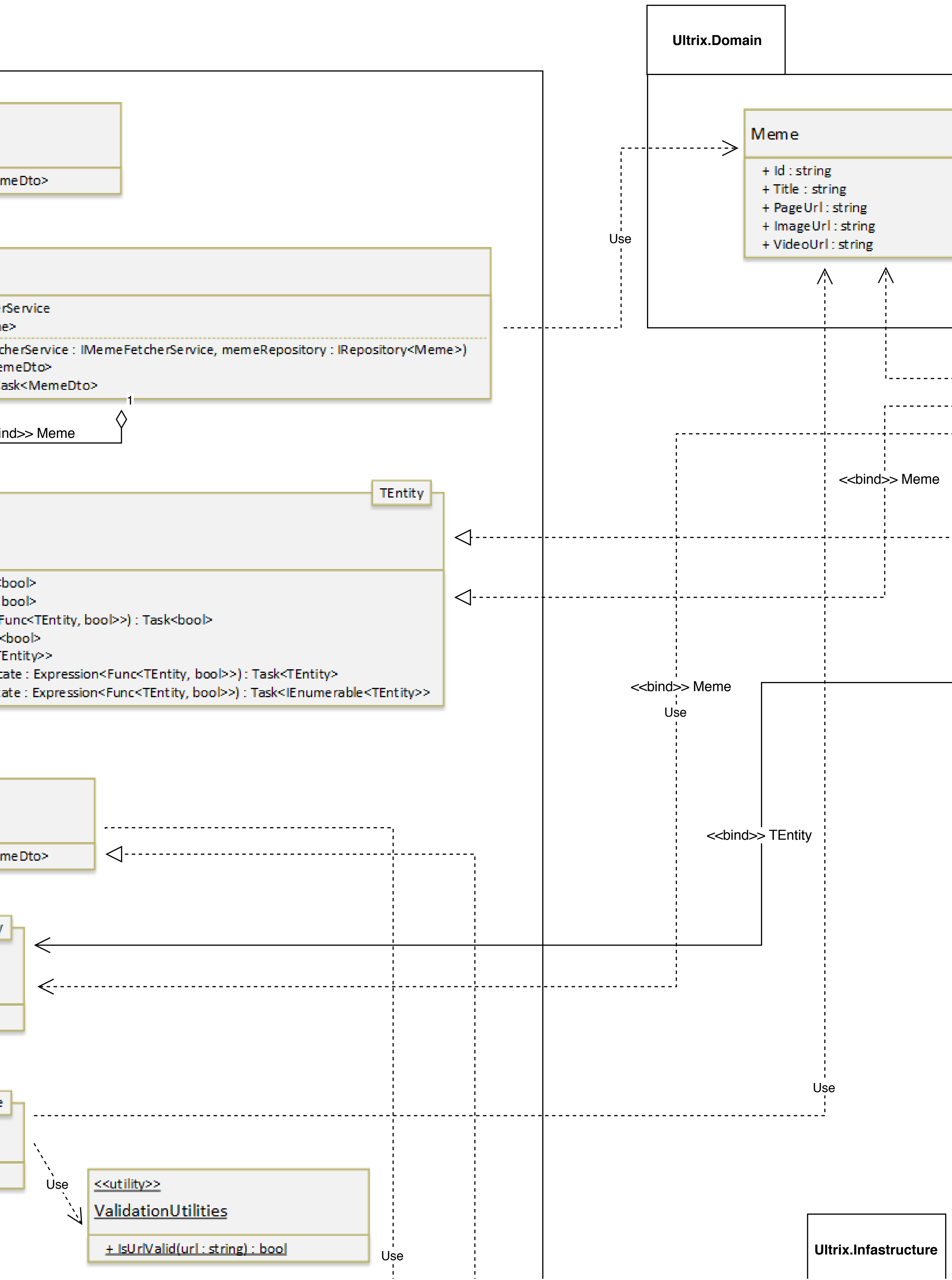

RegisterUserDto
+ <<get, set>> Username : string
+ <<get, set>> Password : string
LoginUserDto
+ <<get, set>> Username : string
+ <<get, set>> Password : string

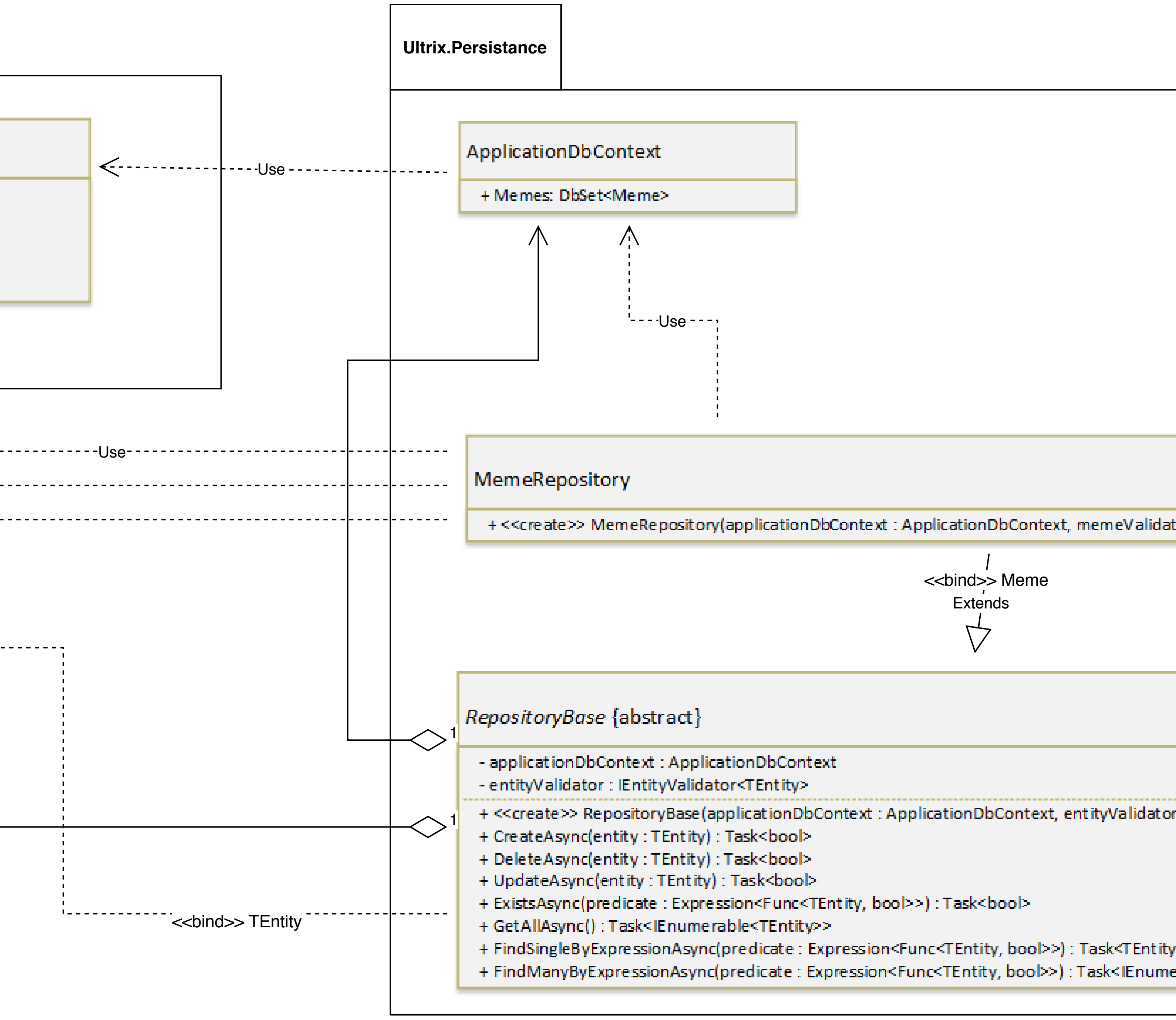

CollectionItemDetail
+ <<get, set>> Id : int
+ <<get, set>> MemeId : string
+ <<get, set>> Meme : Meme
+ <<get, set>> AddedByUserId : int
+ <<get, set>> User : ApplicationUser
+ <<get, set>> CollectionId : int
+ <<get, set>> Collection : Collection
+ <<get, set>> TimestampAdded : DateTime
1..1
0..*
MemeLike
1..1
0..*
Meme
+ <<get, set>> Id : string
+ <<get, set>> Title : string
+ <<get, set>> ImageUrl : string
+ <<get, set>> VideoUrl : string
+ <<get, set>> PageUrl : string
+ <<get, set>> TimestampAdded : DateTime
+ <<get, set>> Comments : List<Comment>
+ <<get, set>> Shares : List<SharedMeme>
+ <<get, set>> Likes : List<MemeLike>
+ <<get, set>> InCollectionItemDetails : List<CollectionItemDetail>
0..*
0..*
1..1
Comment
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUser
+ <<get, set>> MemeId : string
+ <<get, set>> Meme : Meme
+ <<get, set>> Text : string
+ <<get, set>> TimestampAdded : DateTime
1..1
Shared

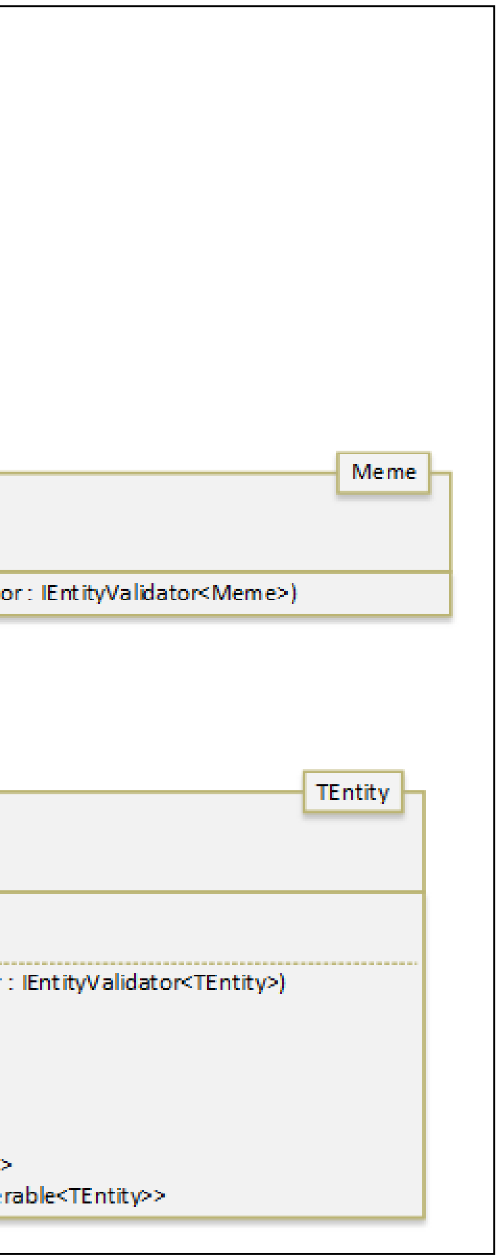

Collection
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUser
+ <<get, set>> Name : string
+ <<get, set>> TimestampAdded : DateTime
+ <<get, set>> CollectionItemDetails : List<CollectionItemDetail>
+ <<get, set>> CollectionSubscribers : List<CollectionSubscriber>
0..*
..1
0..*
1..1
1..1
Id : int
MemeId : string
Meme : Meme
UserId : int
User : ApplicationUser
IsLike : bool
TimestampAdded : DateTime
1..1
0..*
0..*
0..*
ApplicationUser
+ <<get, set>> Id : string
+ <<get, set>> UserName : string
+ <<get, set>> PasswordHash : string
+ <<get, set>> UserDetail : UserDetail
+ <<get, set>> MemeLikes : List<MemeLike>
+ <<get, set>> Followers : List<Follower>
+ <<get, set>> Follows : List<Follower>
+ <<get, set>> Collections : List<Collection>
+ <<get, set>> CollectionSubscribers : List<CollectionSubscriber>
+ <<get, set>> CollectionItemDetails : List<CollectionItemDetail>
+ <<get, set>> SendSharedMemes : List<SharedMeme>
+ <<get, set>> ReceivedSharedMemes : List<SharedMeme>
+ <<get, set>> Comments : List<Comment>
0..*
0..*
0..*
0..*
0..*
0.
ReiceverUser
SenderUser
1..1
1..1
1..1
1..1
1..1
lMeme
et, set>> Id : int
et, set>> ReceiverUserId : int
et, set>> ReceiverUser : ApplicationUser
et, set>> SenderUserId : int
et, set>> SenderUser : ApplicationUser
et, set>> MemeId : string
et, set>> Meme : Meme
et, set>> IsSeen : bool
et, set>> TimestampShared : DateTime
1..1
UserDetail
+ <<get, set>> Id : int
+ <<get, set>> User : ApplicationUser
+ <<get, set>> ProfilePictureData : string
+ <<get, set>> TimestampCreated : DateTime

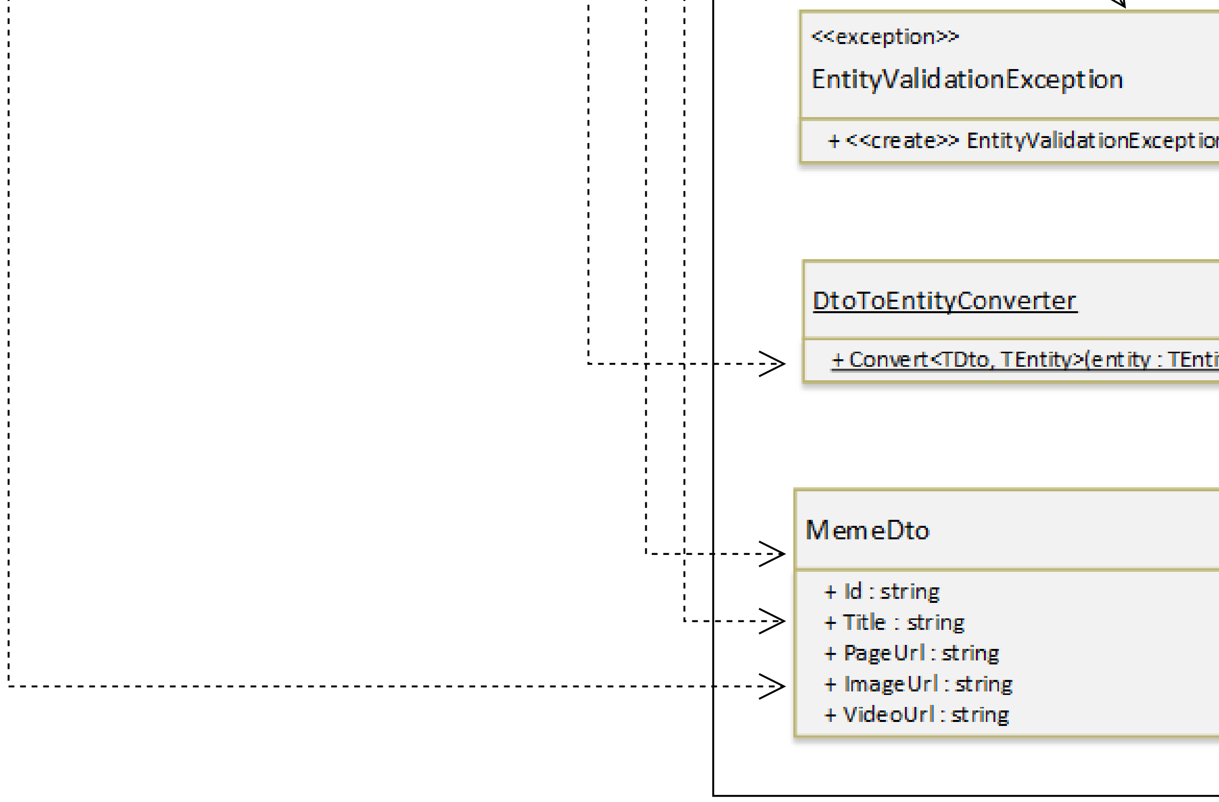

CollectionSubscriber
+ <<get, set>> Id : int
+ <<get, set>> CollectionId : int
+ <<get, set>> Collection : Collection
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUser
+ <<get, set>> IsAuthorized : bool
1..1
1..1
Follows
1..1
Followers
1..1
Follower
+ <<get, set>> Id : int
+ <<get, set>> UserId : int
+ <<get, set>> User : ApplicationUser
+ <<get, set>> FollowerUserId : int
+ <<get, set>> FollowerUser : ApplicationUser
+ <<get, set>> TimestampFollowed : DateTime

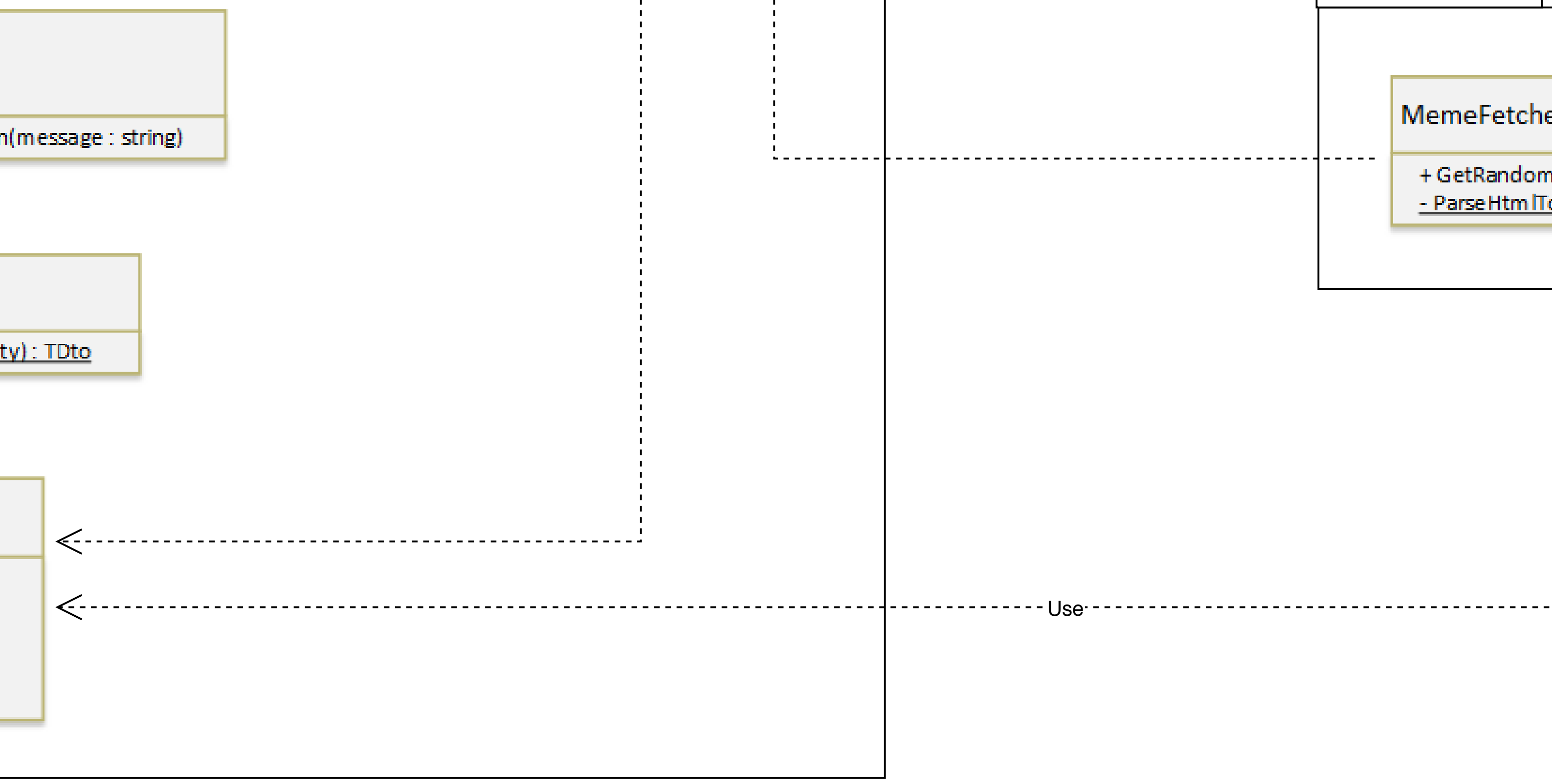

Ultrix.Presentation
MemeController
- memeService : IMemeService
+ <<create>> MemeController(memeService : IMemeService)
+ ShowRandomMemeAsync() : Task<MemeView>
Use
MemeView
+ meme : MemeDto
1
Ultrix.Application
<<Interface>>
IMemeService
+ GetRandomMemeAsync() : Task<Me
MemeService
- memeFetcherService : IMemeFetche
- memeRepository : IRepository<Mem
+ <<create>> MemeService(memeFet
+ GetRandomMemeAsync() : Task<Me
+ GetMemeAsync(memeId : string) : T
1
<<bi
<<Interface>>
IRepository
+ CreateAsync(entity : TEntity) : Task<
+ DeleteAsync(entity : TEntity) : Task<
+ ExistsAsync(predicate : Expression<
+ UpdateAsync(entity : TEntity) : Task
+ GetAllAsync() : Task<IEnumerable<T
+ FindSingleByExpressionAsync(predic
+ FindManyByExpressionAsync(predic
<<Interface>>
IMemeFetcherService
+ GetRandomMemeAsync() : Task<Me
Use
Use
Use
TEntity
<<Interface>>
IEnitityValidator
+ Validate(entity : TEntity) : bool
Meme
MemeValidator
+ Validate(entity : Meme) : bool
<<throw>>
Use
Use

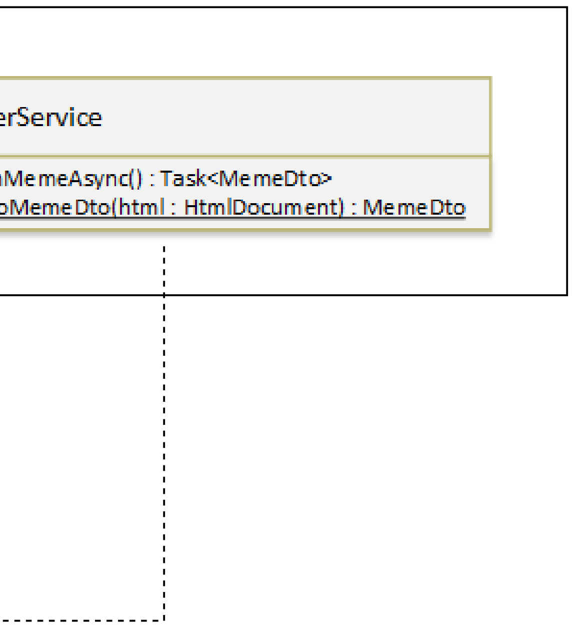

Ultrix.Domain
Meme
+ Id : string
+ Title : string
+ PageUrl : string
+ ImageUrl : string
+ VideoUrl : string
meDto>
erService
ne>
cherService : IMemeFetcherService, memeRepository : IRepository<Meme>)
emeDto>
ask<MemeDto>
1
ind>> Meme
Use
<<bind>> Meme
TEntity
bool>
bool>
Func<TEntity, bool>>) : Task<bool>
<bool>
TEntity>>
cate : Expression<Func<TEntity, bool>>) : Task<TEntity>
cate : Expression<Func<TEntity, bool>>) : Task<IEnumerable<TEntity>>
<<bind>> Meme
Use
<<bind>> TEntity
meDto>
Use
Use
<<utility>>
ValidationUtilities
+ IsUrlValid(url : string) : bool
Use
Ultrix.Infastructure


Ultrix.Persistance
ApplicationDbContext
+ Memes: DbSet<Meme>
Use
Use
Use
MemeRepository
+ <<create>> MemeRepository(applicationDbContext : ApplicationDbContext, memeValidat
<<bind>> Meme
Extends
RepositoryBase {abstract}
- applicationDbContext : ApplicationDbContext
- entityValidator : IEntityValidator<TEntity>
+ <<create>> RepositoryBase(applicationDbContext : ApplicationDbContext, entityValidator
+ CreateAsync(entity : TEntity) : Task<bool>
+ DeleteAsync(entity : TEntity) : Task<bool>
+ UpdateAsync(entity : TEntity) : Task<bool>
+ ExistsAsync(predicate : Expression<Func<TEntity, bool>>) : Task<bool>
+ GetAllAsync() : Task<IEnumerable<TEntity>>
+ FindSingleByExpressionAsync(predicate : Expression<Func<TEntity, bool>>) : Task<TEntity
+ FindManyByExpressionAsync(predicate : Expression<Func<TEntity, bool>>) : Task<IEnume
1
1
<<bind>> TEntity


Meme
or : IEntityValidator<Meme>)
TEntity
: IEntityValidator<TEntity>)
>
erable<TEntity>>


<<exception>>
EntityValidationException
+ <<create>> EntityValidationExceptio
DtoToEntityConverter
+ Convert<TDto, TEntity>(entity : TEnti
MemeDto
+ Id : string
+ Title : string
+ PageUrl : string
+ ImageUrl : string
+ VideoUrl : string


n(message : string)
ty) : TDto
Use
MemeFetche
+ GetRandom
- ParseHtmlTo


erService
nMemeAsync() : Task<MemeDto>
oMemeDto(html : HtmlDocument) : MemeDto