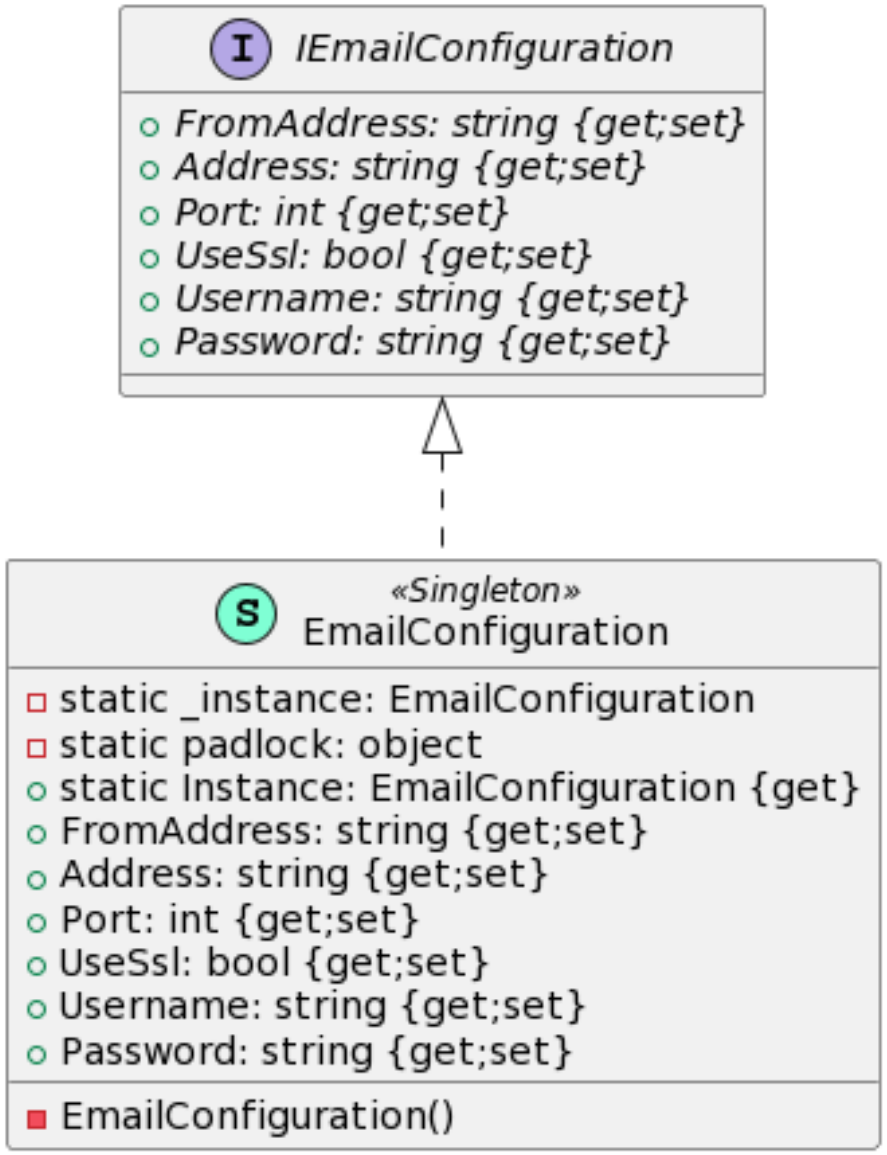
Design patterns

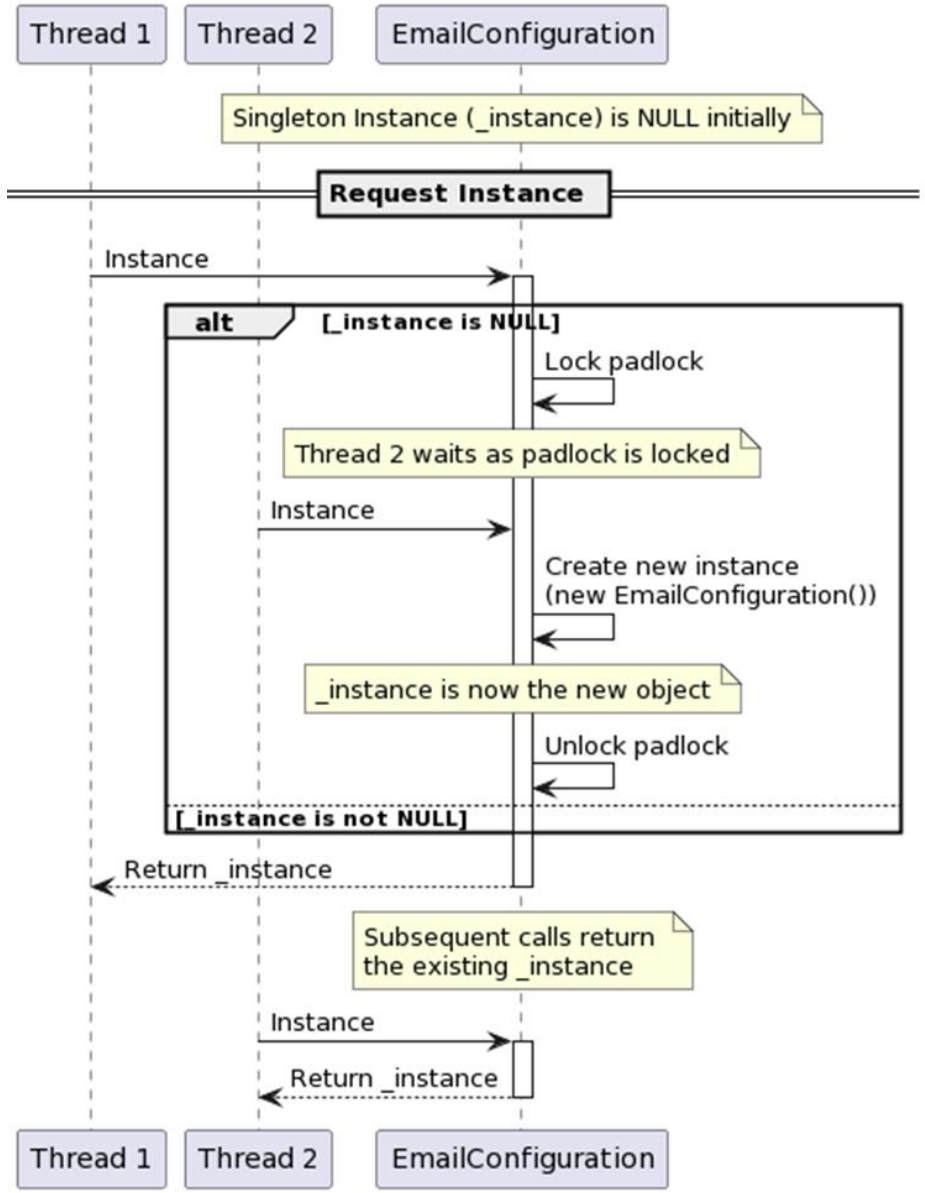
By Bodarev Victor

TI-202

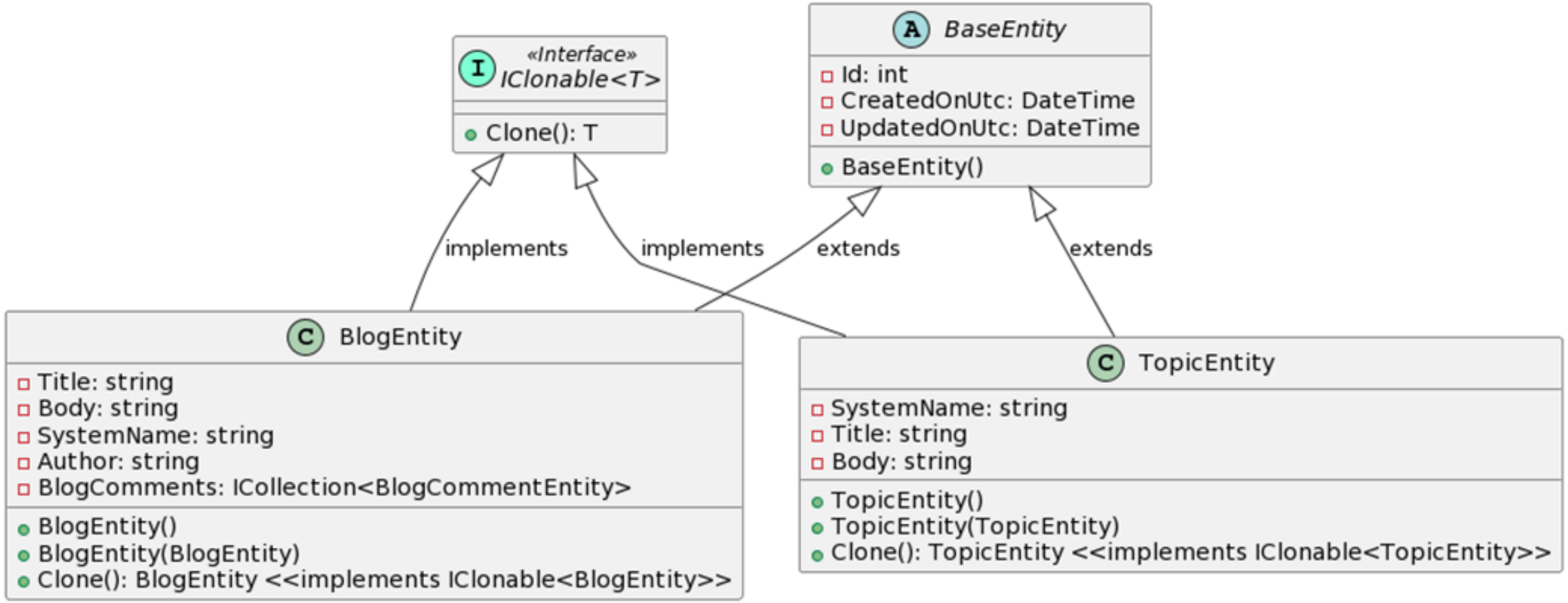
Creational patterns


IEmailConfiguration
FromAddress: string {get;set}
Address: string {get;set}
Port: int {get;set}
UseSsl: bool {get;set}
Username: string {get;set}
Password: string {get;set}
«Singleton»
EmailConfiguration
static _instance: EmailConfiguration
static padlock: object
static Instance: EmailConfiguration {get}
FromAddress: string {get;set}
Address: string {get;set}
Port: int {get;set}
UseSsl: bool {get;set}
Username: string {get;set}
Password: string {get;set}
EmailConfiguration()

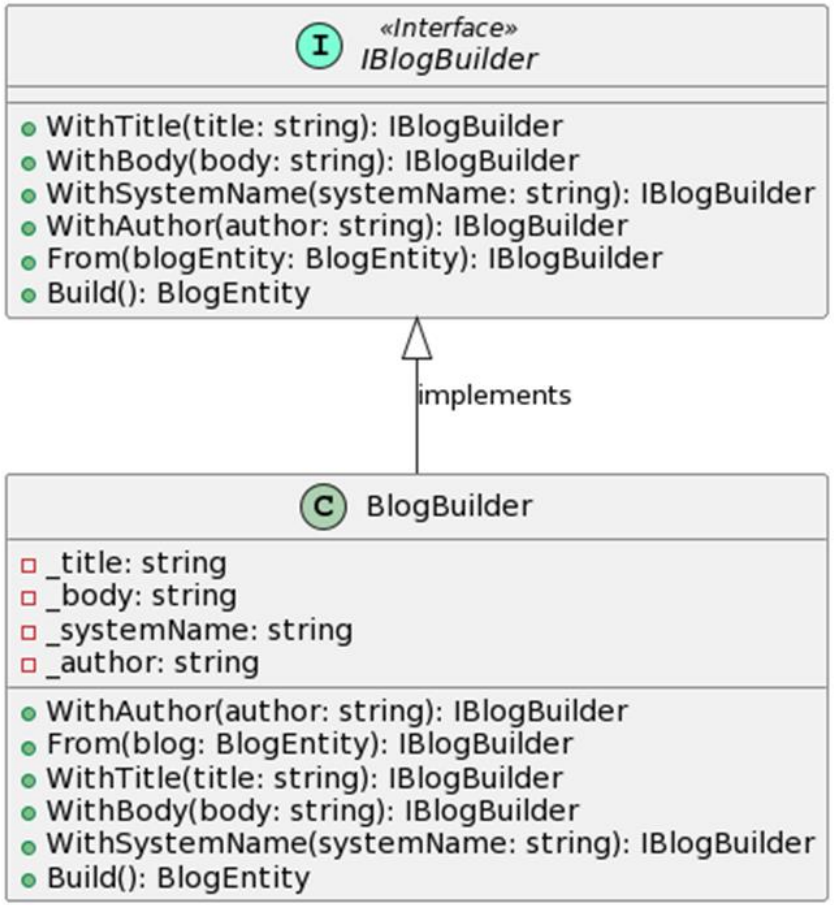
Singletone

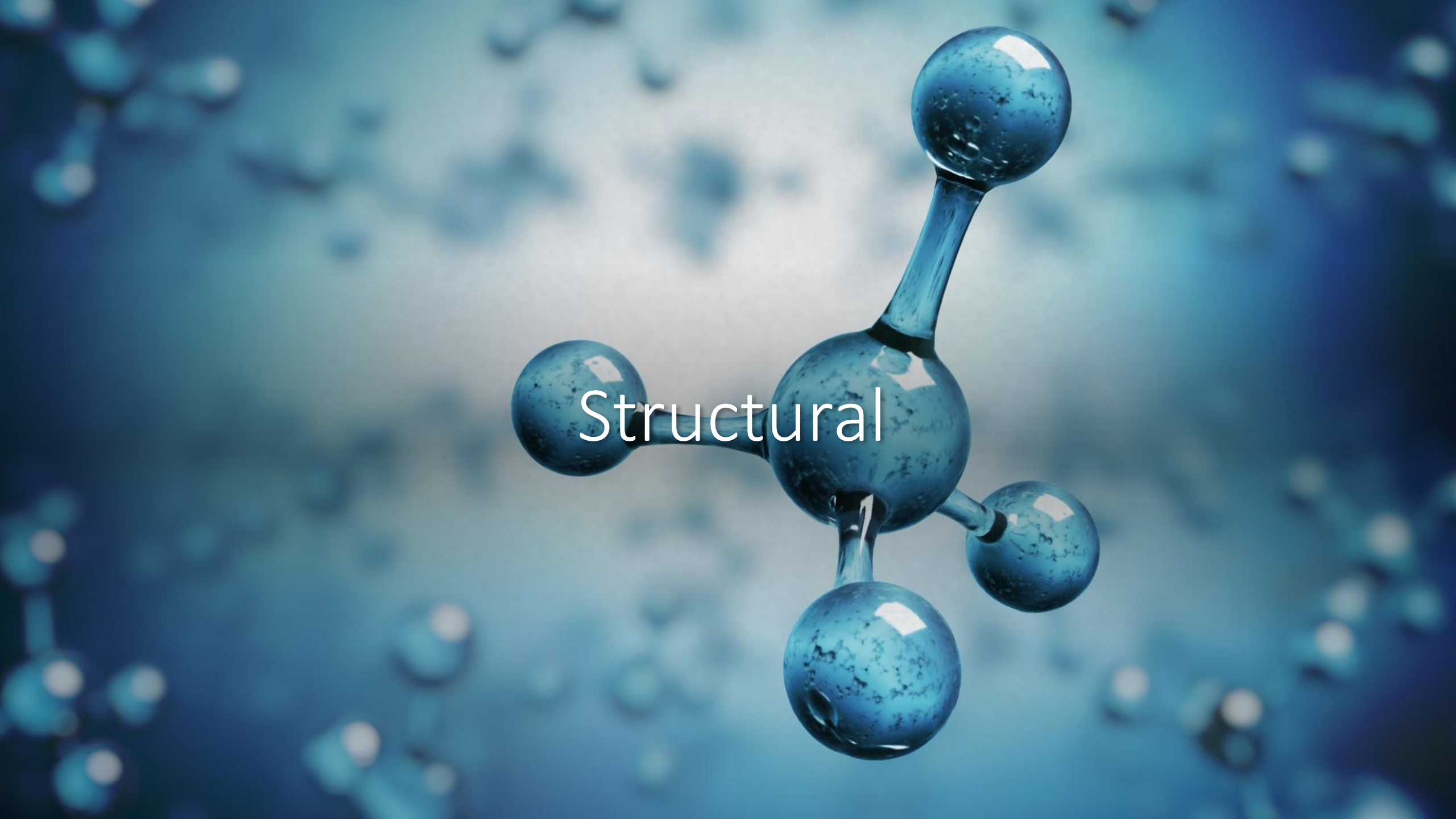

Thread 1
Thread 2
EmailConfiguration
Singleton Instance (_instance) is NULL initially
Request Instance
Instance
alt
[_instance is NULL]
Lock padlock
Thread 2 waits as padlock is locked
Instance
Create new instance
(new EmailConfiguration())
_instance is now the new object
Unlock padlock
[_instance is not NULL]
Return _instance
Subsequent calls return
the existing _instance
Instance
Return _instance
Thread 1
Thread 2
EmailConfiguration

Prototype

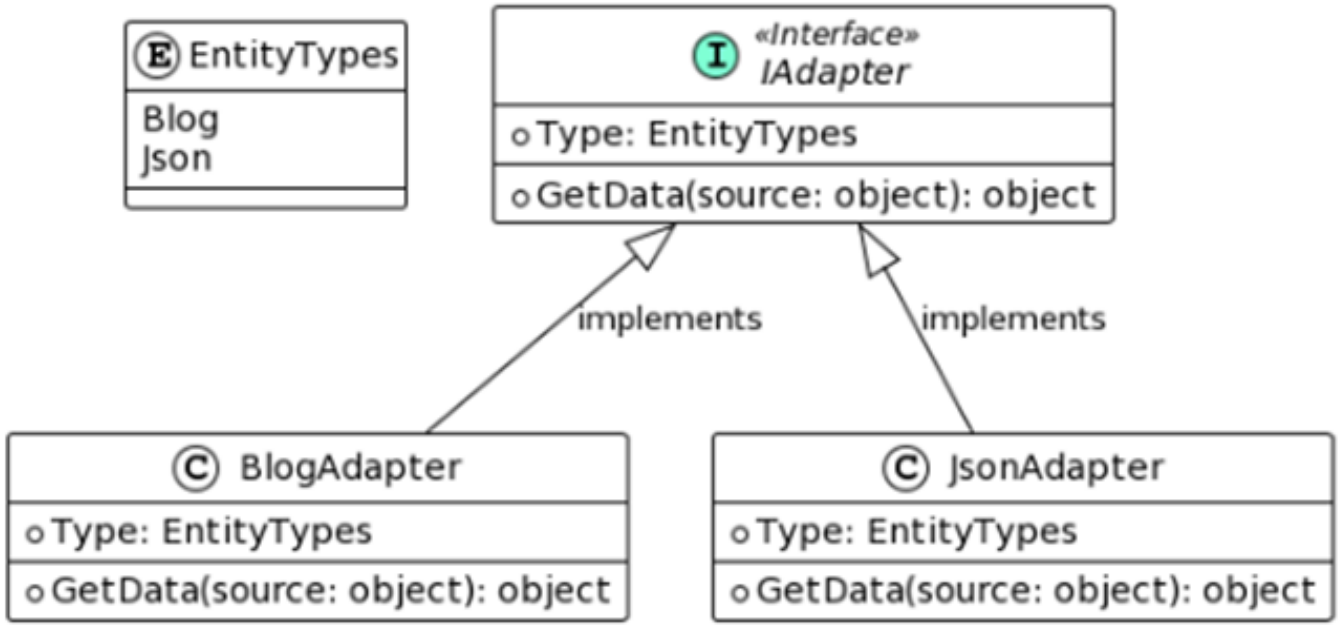

«Interface»
IClonable<T>
Clone(): T
BaseEntity
Id: int
CreatedOnUtc: DateTime
UpdatedOnUtc: DateTime
BaseEntity()
implements
implements
extends
extends
BlogEntity
Title: string
Body: string
SystemName: string
Author: string
BlogComments: ICollection<BlogCommentEntity>
BlogEntity()
BlogEntity(BlogEntity)
Clone(): BlogEntity <<implements IClonable<BlogEntity>>
TopicEntity
SystemName: string
Title: string
Body: string
TopicEntity()
TopicEntity(TopicEntity)
Clone(): TopicEntity <<implements IClonable<TopicEntity>>

Builder

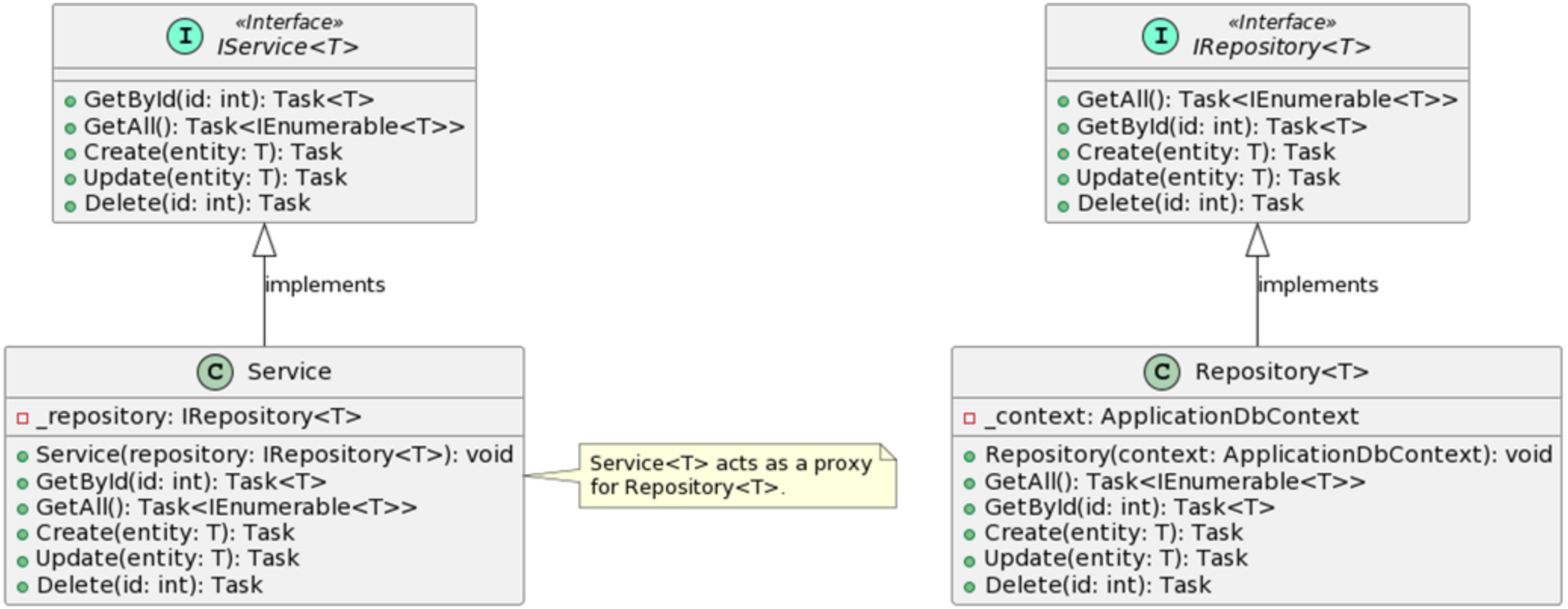

«Interface»
IBlogBuilder
WithTitle(title: string): IBlogBuilder
WithBody(body: string): IBlogBuilder
WithSystemName(systemName: string): IBlogBuilder
WithAuthor(author: string): IBlogBuilder
From(blogEntity: BlogEntity): IBlogBuilder
Build(): BlogEntity
implements
BlogBuilder
_title: string
_body: string
_systemName: string
_author: string
WithAuthor(author: string): IBlogBuilder
From(blog: BlogEntity): IBlogBuilder
WithTitle(title: string): IBlogBuilder
WithBody(body: string): IBlogBuilder
WithSystemName(systemName: string): IBlogBuilder
Build(): BlogEntity

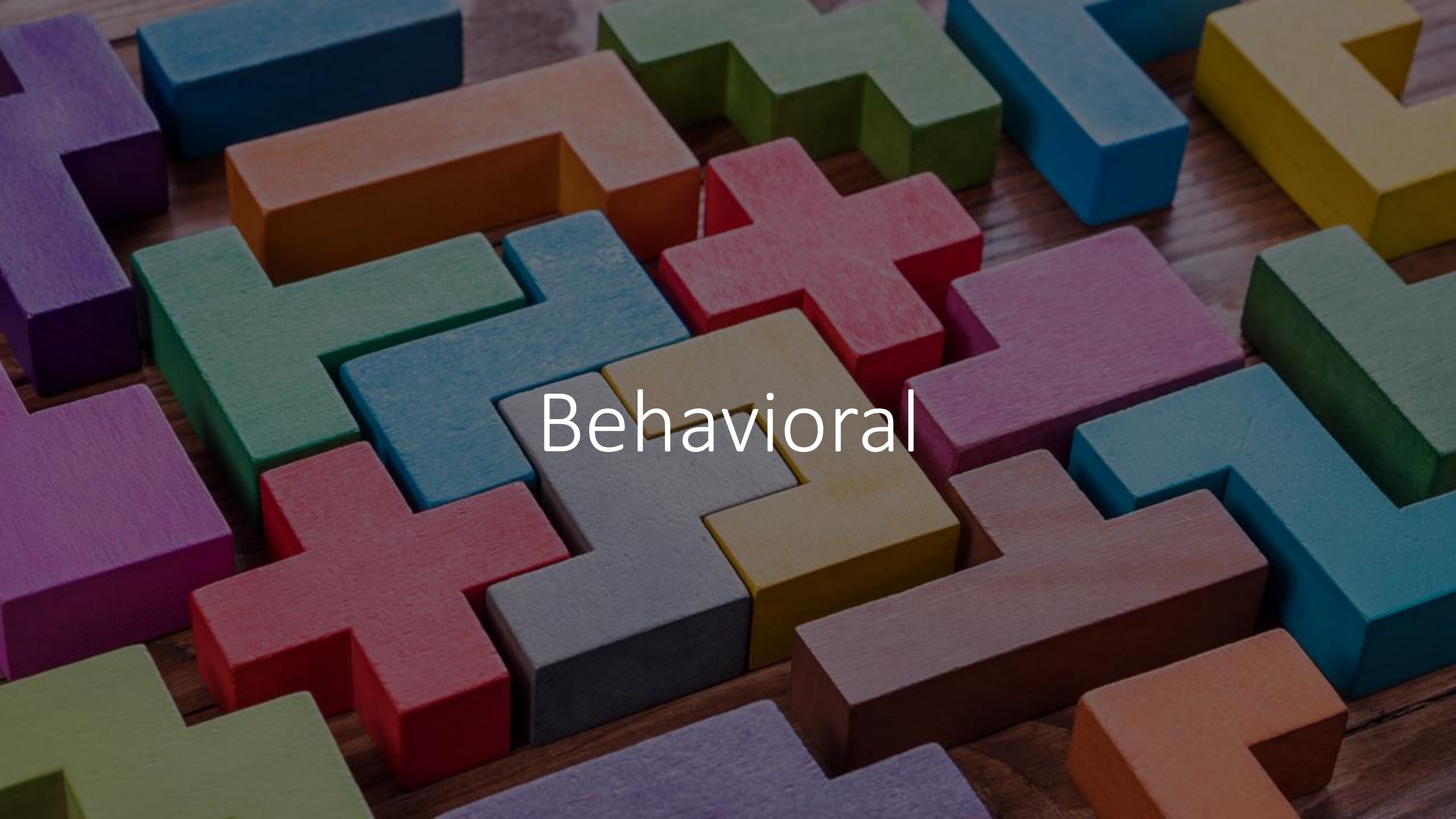

Structural

Adapter

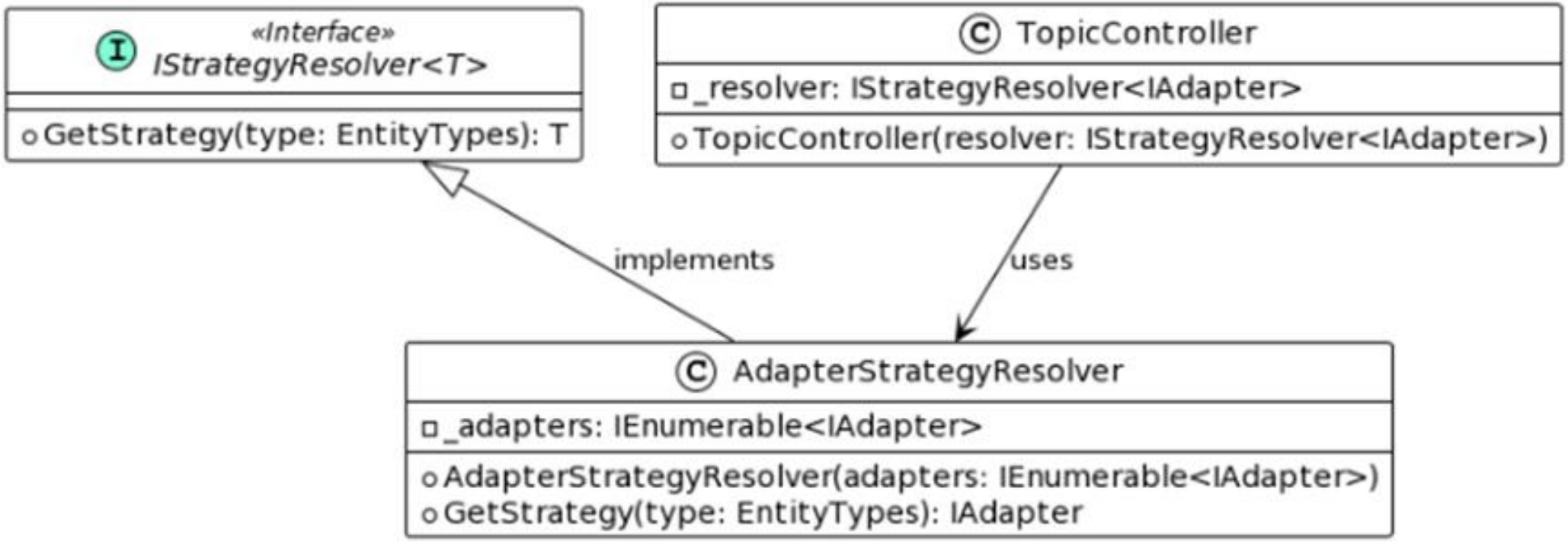

EntityTypes
Blog
Json
«Interface»
IAdapter
Type: EntityTypes
GetData(source: object): object
implements
implements
BlogAdapter
Type: EntityTypes
GetData(source: object): object
JsonAdapter
Type: EntityTypes
GetData(source: object): object

Proxy

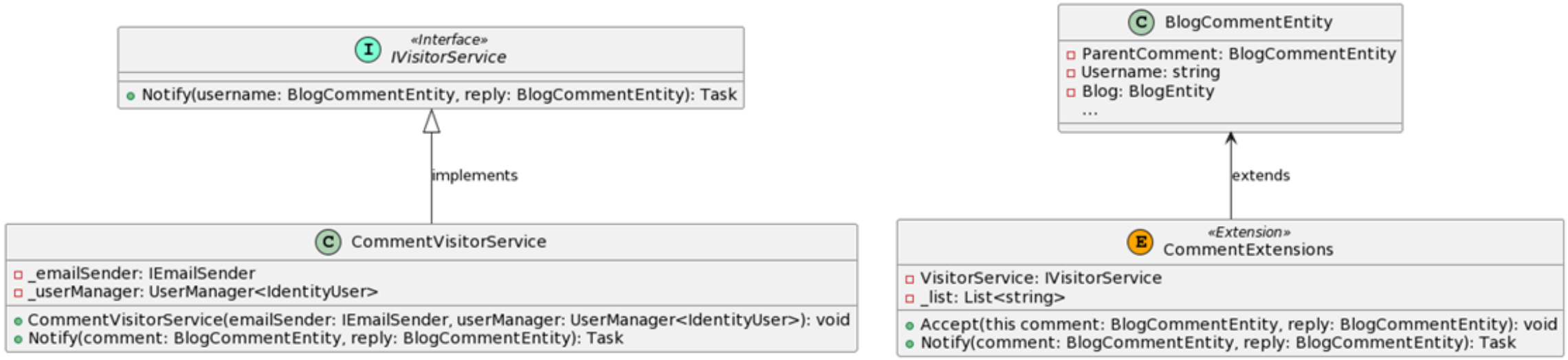

«Interface»
IService<T>
GetById(id: int): Task<T>
GetAll(): Task<IEnumerable<T>>
Create(entity: T): Task
Update(entity: T): Task
Delete(id: int): Task
implements
Service
_repository: IRepository<T>
Service(repository: IRepository<T>): void
GetById(id: int): Task<T>
GetAll(): Task<IEnumerable<T>>
Create(entity: T): Task
Update(entity: T): Task
Delete(id: int): Task
Service<T> acts as a proxy for Repository<T>.
«Interface»
IRepository<T>
GetAll(): Task<IEnumerable<T>>
GetById(id: int): Task<T>
Create(entity: T): Task
Update(entity: T): Task
Delete(id: int): Task
implements
Repository<T>
_context: ApplicationDbContext
Repository(context: ApplicationDbContext): void
GetAll(): Task<IEnumerable<T>>
GetById(id: int): Task<T>
Create(entity: T): Task
Update(entity: T): Task
Delete(id: int): Task

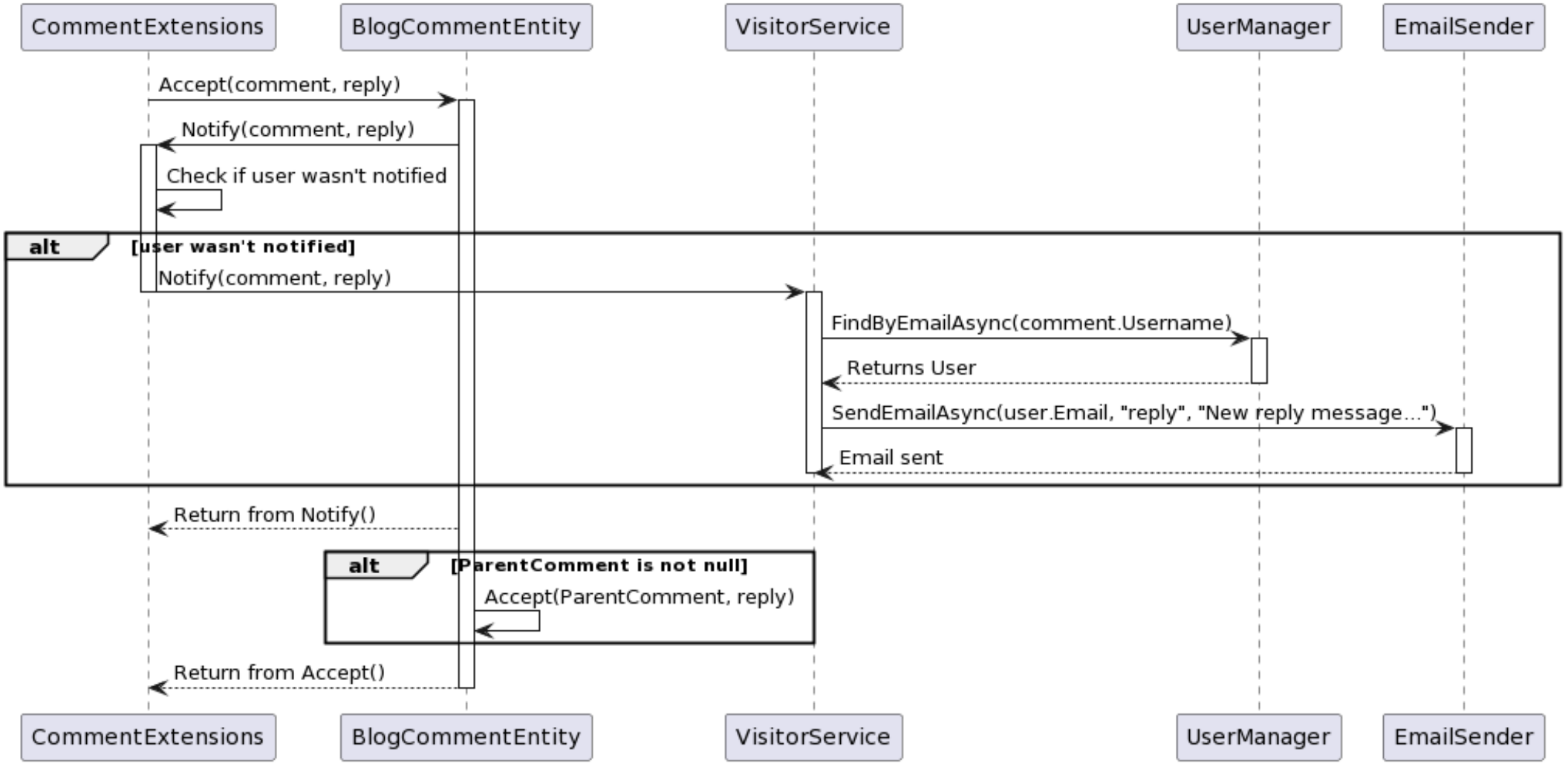

Behavioral

Strategy


«Interface»
IStrategyResolver<T>
GetStrategy(type: EntityTypes): T
TopicController
_resolver: IStrategyResolver<IAdapter>
TopicController(resolver: IStrategyResolver<IAdapter>)
implements
uses
AdapterStrategyResolver
_adapters: IEnumerable<IAdapter>
AdapterStrategyResolver(adapters: IEnumerable<IAdapter>)
GetStrategy(type: EntityTypes): IAdapter

Visitor


«Interface»
IVisitorService
Notify(username: BlogCommentEntity, reply: BlogCommentEntity): Task
implements
CommentVisitorService
_emailSender: IEmailSender
_userManager: UserManager<IdentityUser>
CommentVisitorService(emailSender: IEmailSender, userManager: UserManager<IdentityUser>): void
Notify(comment: BlogCommentEntity, reply: BlogCommentEntity): Task
BlogCommentEntity
ParentComment: BlogCommentEntity
Username: string
Blog: BlogEntity
...
extends
«Extension»
CommentExtensions
VisitorService: IVisitorService
_list: List<string>
Accept(this comment: BlogCommentEntity, reply: BlogCommentEntity): void
Notify(comment: BlogCommentEntity, reply: BlogCommentEntity): Task


CommentExtensions
BlogCommentEntity
VisitorService
UserManager
EmailSender
Accept(comment, reply)
Notify(comment, reply)
Check if user wasn't notified
alt
[user wasn't notified]
Notify(comment, reply)
FindByEmailAsync(comment.Username)
Returns User
SendEmailAsync(user.Email, "reply", "New reply message...")
Email sent
Return from Notify()
alt
[ParentComment is not null]
Accept(ParentComment, reply)
Return from Accept()
CommentExtensions
BlogCommentEntity
VisitorService
UserManager
EmailSender