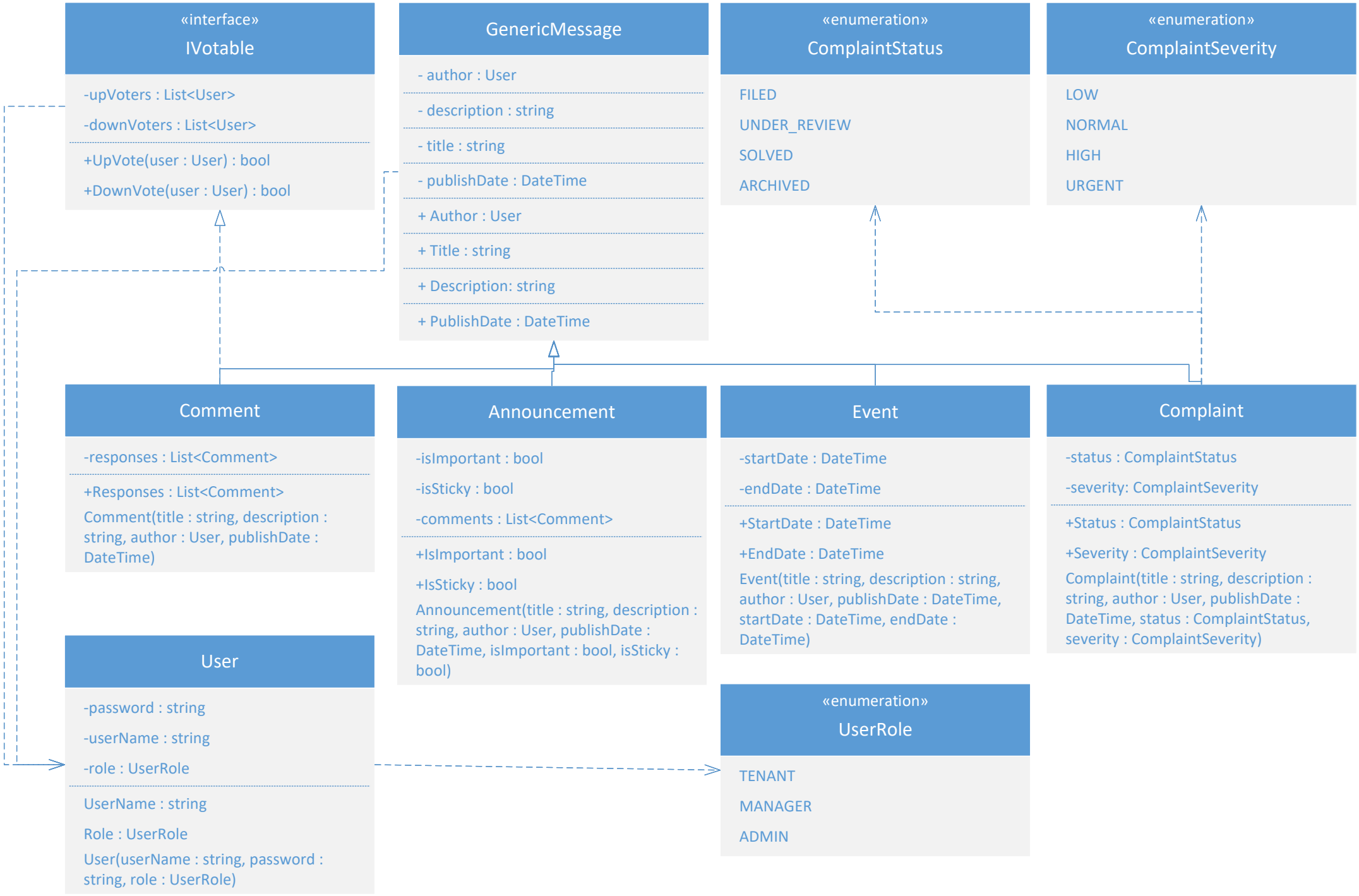
«interface»
IVotable
-upVoters : List<User>
-downVoters : List<User>
+UpVote(user : User) : bool
+DownVote(user : User) : bool
GenericMessage
- author : User
- description : string
- title : string
- publishDate : DateTime
+ Author : User
+ Title : string
+ Description: string
+ PublishDate : DateTime
«enumeration»
ComplaintStatus
FILED
UNDER_REVIEW
SOLVED
ARCHIVED
«enumeration»
ComplaintSeverity
LOW
NORMAL
HIGH
URGENT
Comment
-responses : List<Comment>
+Responses : List<Comment>
Comment(title : string, description : string, author : User, publishDate : DateTime)
Announcement
-isImportant : bool
-isSticky : bool
-comments : List<Comment>
+IsImportant : bool
+IsSticky : bool
Announcement(title : string, description : string, author : User, publishDate : DateTime, isImportant : bool, isSticky : bool)
Event
-startDate : DateTime
-endDate : DateTime
+StartDate : DateTime
+EndDate : DateTime
Event(title : string, description : string, author : User, publishDate : DateTime, startDate : DateTime, endDate : DateTime)
Complaint
-status : ComplaintStatus
-severity: ComplaintSeverity
+Status : ComplaintStatus
+Severity : ComplaintSeverity
Complaint(title : string, description : string, author : User, publishDate : DateTime, status : ComplaintStatus, severity : ComplaintSeverity)
User
-password : string
-userName : string
-role : UserRole
UserName : string
Role : UserRole
User(userName : string, password : string, role : UserRole)
«enumeration»
UserRole
TENANT
MANAGER
ADMIN

UserManager

CreateUser(userName : string, password : string, role : UserRole)
UpdateUser(id : int, userName : string, password : string, role : UserRole)

DeleteUser(id : int)

GetAllUsers()

GetUserId(userName : string)

GetUsersByPage(p : int, c : int)

AuthenticatedUser(name : string, password : string)

ComplaintManager

CreateComplaint(title : string, description : string, author : User, publishDate : DateTime, status : ComplaintStatus, severity : ComplaintServerity)
UpdateComplaint(id: int, title : string, description : string, author : User, publishDate : DateTime, status : ComplaintStatus, severity : ComplaintServerity, currentUser : User)

GetAllComplaintsByUser(user: User)

GetAllComplaintsByStatusOrSeverity(status : ComplaintStatus?, severity : ComplaintSeverity?)

GetComplaint(id : int)

GetAllComplaintsByPage(page : int, count : int)
AddResponse(id : int, comment : Comment)

AnnouncementManager

CreateAnnouncement(title : string, description : string, author : User, publishDate : DateTime, isImportant : bool, isSticky : bool)
UpdateAnnouncement(id: int, title : string, description : string, author : User, publishDate : DateTime, isImportant : bool, isSticky : bool, currentUser : User)
DeleteAnnouncement(id : int, currentUser: User)
GetAnnouncementsByPage(page : int, count : int)

GetAnnouncementById(id : int)

AddComment(id: int, title : string, description : string, author : User, publishDate : DateTime)

UpVote(id : int, currentUser : User)

DownVote(id : int, currentUser : User)

CommentManager

CreateComment(title : string, description : string, author : User, publishDate : DateTime)
UpdateComment(title : string, description : string, author : User, publishDate : DateTime, currentUser : User)
DeleteAnnouncement(id : int, currentUser : User)

GetComment(id : int)

AddResponse(id: int, title : string, description : string, author : User, publishDate : DateTime)

UpVote(id : int, currentUser : User)

DownVote(id : int, currentUser : User)

EventManager

CreateEvent(id: int, title : string, description : string, author : User, publishDate : DateTime, startDate : DateTime, endDate : DateTime)
UpdateEvent(id: int, title : string, description : string, author : User, publishDate : DateTime, startDate : DateTime, endDate : DateTime, currentUser : User)

DeleteEvent(id : int, currentUser : User)

GetAllEventsByMonth(queryDate : DateTime)

GetEvent(id : int)

UserRepository

CreateConnection()

CreateUser(userName : string, password : string, role : UserRole)

UpdateUser(id : int, userName : string, password : string, role : UserRole)

DeleteUser(id : int)

GetAllUsers()

GetUserId(userName : string)

GetUsersByPage(p : int, c : int)

ComplaintRepository

CreateConnection()

CreateComplaint(title : string, description : string, author : User, publishDate : DateTime, status : ComplaintStatus, severity : ComplaintServerity)

UpdateComplaint(id: int, title : string, description : string, author : User, publishDate : DateTime, status : ComplaintStatus, severity : ComplaintServerity, currentUser : User)

GetAllComplaintsByUser(user: User)

GetAllComplaintsByStatusOrSeverity(status : ComplaintStatus?, severity : ComplaintSeverity?)

GetComplaint(id : int)

GetAllComplaintsByPage(page : int, count : int)

AddResponse(id : int, comment : Comment)

AnnouncementRepository

CreateConnection()

CreateAnnouncement(title : string, description : string, author : User, publishDate : DateTime, isImportant : bool, isSticky : bool)

UpdateAnnouncement(id: int, title : string, description : string, author : User, publishDate : DateTime, isImportant : bool, isSticky : bool, currentUser : User)

DeleteAnnouncement(id : int, currentUser: User)

GetAnnouncementsByPage(page : int, count : int)

GetAnnouncementById(id : int)

AddComment(id: int, title : string, description : string, author : User, publishDate : DateTime)

UpVote(id : int, currentUser : User)

DownVote(id : int, currentUser : User)

CommentRepository

CreateConnection()

CreateComment(title : string, description : string, author : User, publishDate : DateTime)

UpdateComment(title : string, description : string, author : User, publishDate : DateTime, currentUser : User)

DeleteAnnouncement(id : int, currentUser : User)

GetComment(id : int)

AddResponse(id: int, title : string, description : string, author : User, publishDate : DateTime)

UpVote(id : int, currentUser : User)

DownVote(id : int, currentUser : User)

EventRepository

CreateConnection()

CreateEvent(id: int, title : string, description : string, author : User, publishDate : DateTime, startDate : DateTime, endDate : DateTime)

UpdateEvent(id: int, title : string, description : string, author : User, publishDate : DateTime, startDate : DateTime, endDate : DateTime, currentUser : User)

DeleteEvent(id : int, currentUser : User)

GetAllEventsByMonth(queryDate : DateTime)

GetEvent(id : int)