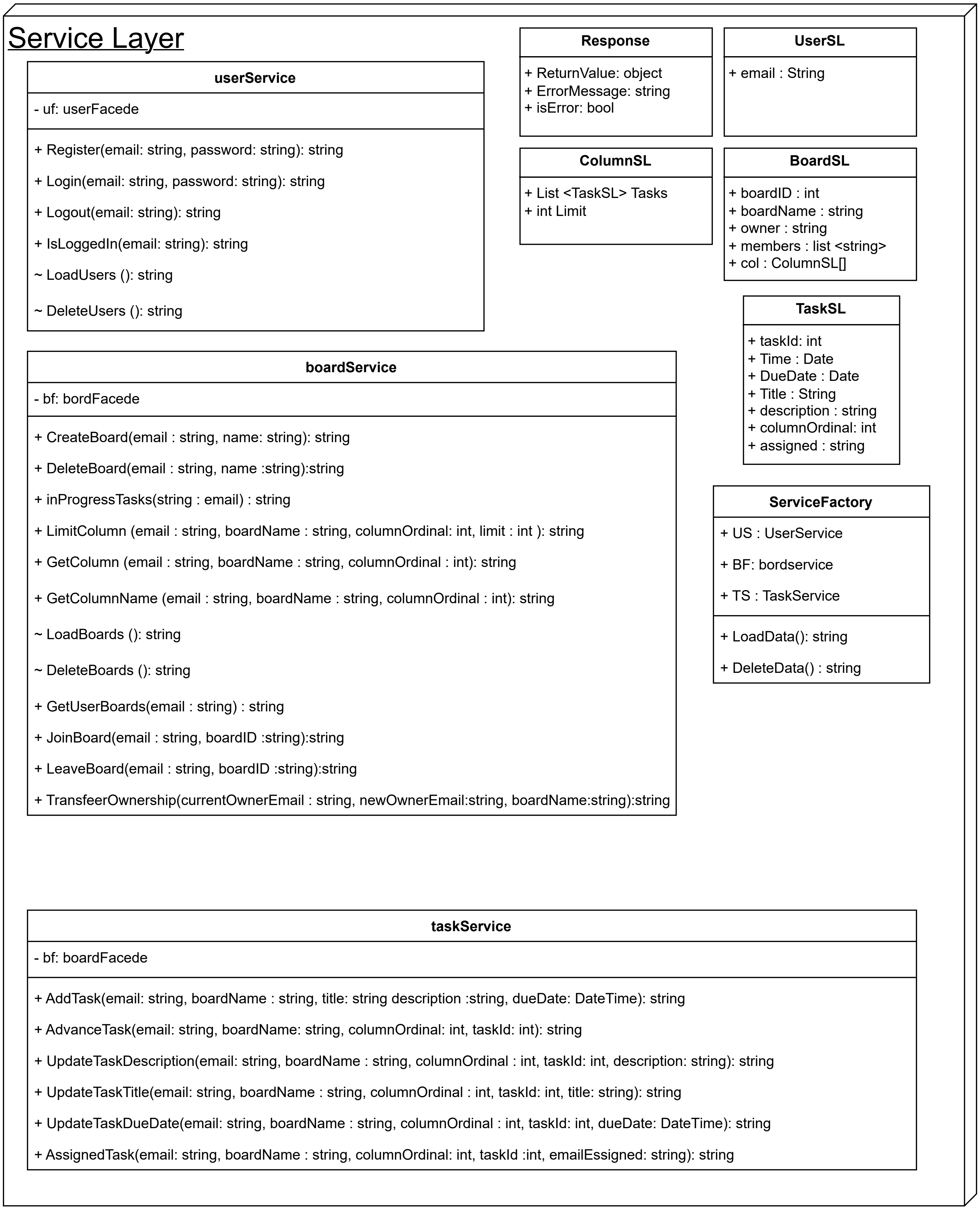

Service Layer
userService
- uf: userFacede
+ Register(email: string, password: string): string
+ Login(email: string, password: string): string
+ Logout(email: string): string
+ IsLoggedIn(email: string): string
~ LoadUsers (): string
~ DeleteUsers (): string
Response
+ ReturnValue: object
+ ErrorMessage: string
+ isError: bool
UserSL
+ email : String
ColumnSL
+ List <TaskSL> Tasks
+ int Limit
BoardSL
+ boardID : int
+ boardName : string
+ owner : string
+ members : list <string>
+ col : ColumnSL[]
TaskSL
+ taskId: int
+ Time : Date
+ DueDate : Date
+ Title : String
+ description : string
+ columnOrdinal: int
+ assigned : string
boardService
- bf: bordFacede
+ CreateBoard(email : string, name: string): string
+ DeleteBoard(email : string, name :string):string
+ inProgressTasks(string : email) : string
+ LimitColumn (email : string, boardName : string, columnOrdinal: int, limit : int ): string
+ GetColumn (email : string, boardName : string, columnOrdinal : int): string
+ GetColumnName (email : string, boardName : string, columnOrdinal : int): string
~ LoadBoards (): string
~ DeleteBoards (): string
+ GetUserBoards(email : string) : string
+ JoinBoard(email : string, boardID :string):string
+ LeaveBoard(email : string, boardID :string):string
+ TransfeerOwnership(currentOwnerEmail : string, newOwnerEmail:string, boardName:string):string
ServiceFactory
+ US : UserService
+ BF: bordservice
+ TS : TaskService
+ LoadData(): string
+ DeleteData() : string
taskService
- bf: boardFacede
+ AddTask(email: string, boardName : string, title: string description :string, dueDate: DateTime): string
+ AdvanceTask(email: string, boardName: string, columnOrdinal: int, taskId: int): string
+ UpdateTaskDescription(email: string, boardName : string, columnOrdinal : int, taskId: int, description: string): string
+ UpdateTaskTitle(email: string, boardName : string, columnOrdinal : int, taskId: int, title: string): string
+ UpdateTaskDueDate(email: string, boardName : string, columnOrdinal : int, taskId: int, dueDate: DateTime): string
+ AssignedTask(email: string, boardName : string, columnOrdinal: int, taskId :int, emailEssigned: string): string

## Service Layer Changes

**userService**

**new function -**
LoadUser, DeleteUser - in order to fetch the data from the db

**boardService**

**new function (due to new requierments( -**
LoadBoards,DeleteBoards- in order to fetch the data from the db.
JoinBoard,LeaveBoard- in order to let the user the ability join and leave other boards.
TransferOwnership - let an owner to assign other member in the ownership.

**UserSL**

no changes

**taskService**

**new function -**
AssignTask -due to new requirement. let user the ability to assign task to a member.

**BordSL**

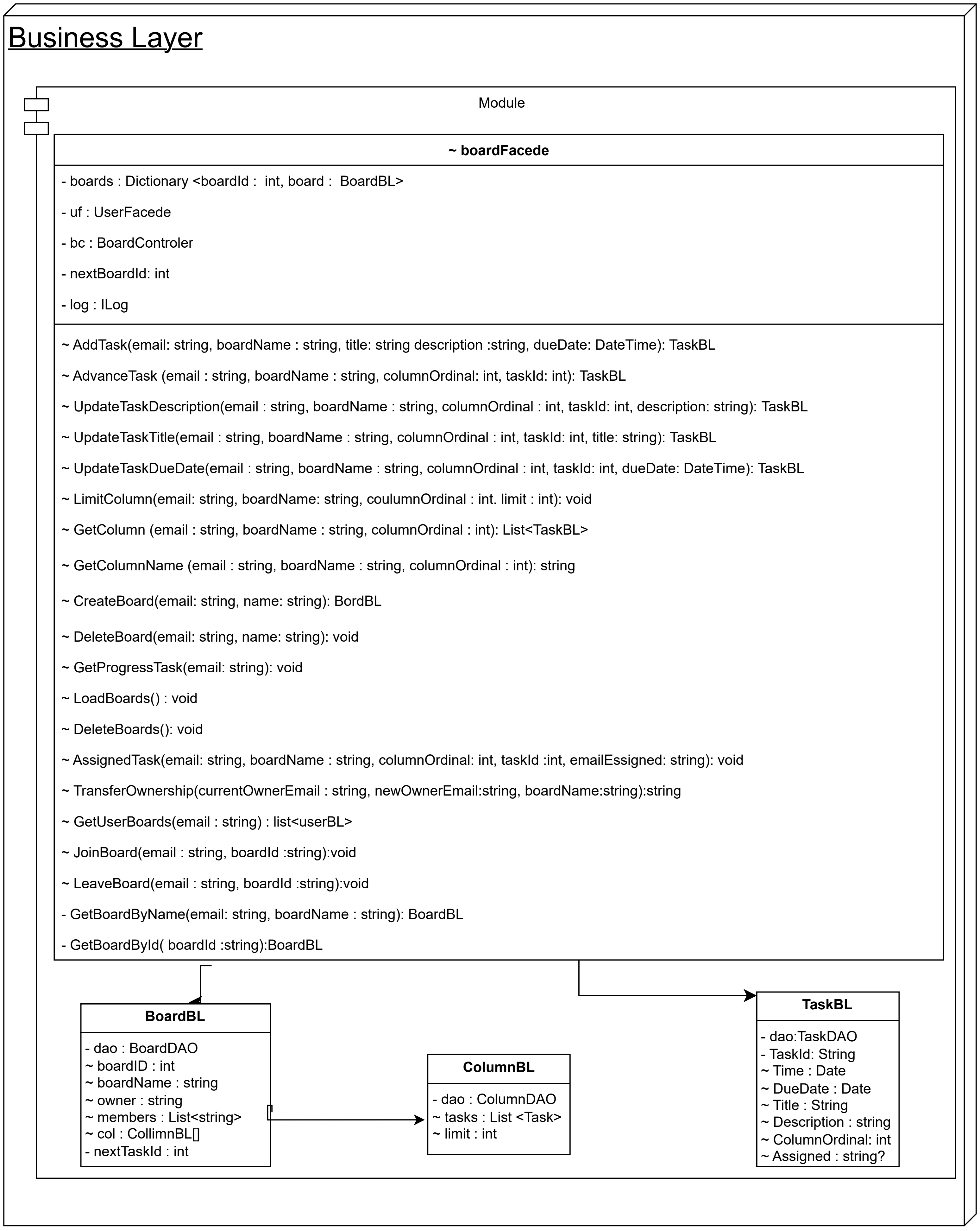
**new varible-**
onwer, members, boardId - due to new requierments.

**Response**

no changes

**ServiceFactory**

**new functions**
LoadData, DeleteData - in order to fetch the data from the db



Business Layer

Module

**~ boardFacede**

- boards : Dictionary <boardId :  int, board :  BoardBL>
- uf : UserFacede
- bc : BoardControler
- nextBoardId: int
- log : ILog

~ AddTask(email: string, boardName : string, title: string description :string, dueDate: DateTime): TaskBL
~ AdvanceTask (email : string, boardName : string, columnOrdinal: int, taskId: int): TaskBL
~ UpdateTaskDescription(email : string, boardName : string, columnOrdinal : int, taskId: int, description: string): TaskBL
~ UpdateTaskTitle(email : string, boardName : string, columnOrdinal : int, taskId: int, title: string): TaskBL
~ UpdateTaskDueDate(email : string, boardName : string, columnOrdinal : int, taskId: int, dueDate: DateTime): TaskBL
~ LimitColumn(email: string, boardName: string, coulumnOrdinal : int. limit : int): void
~ GetColumn (email : string, boardName : string, columnOrdinal : int): List<TaskBL>
~ GetColumnName (email : string, boardName : string, columnOrdinal : int): string
~ CreateBoard(email: string, name: string): BordBL
~ DeleteBoard(email: string, name: string): void
~ GetProgressTask(email: string): void
~ LoadBoards() : void
~ DeleteBoards(): void
~ AssignedTask(email: string, boardName : string, columnOrdinal: int, taskId :int, emailEssigned: string): void
~ TransferOwnership(currentOwnerEmail : string, newOwnerEmail:string, boardName:string):string
~ GetUserBoards(email : string) : list<userBL>
~ JoinBoard(email : string, boardId :string):void
~ LeaveBoard(email : string, boardId :string):void
- GetBoardByName(email: string, boardName : string): BoardBL
- GetBoardById( boardId :string):BoardBL

**BoardBL**

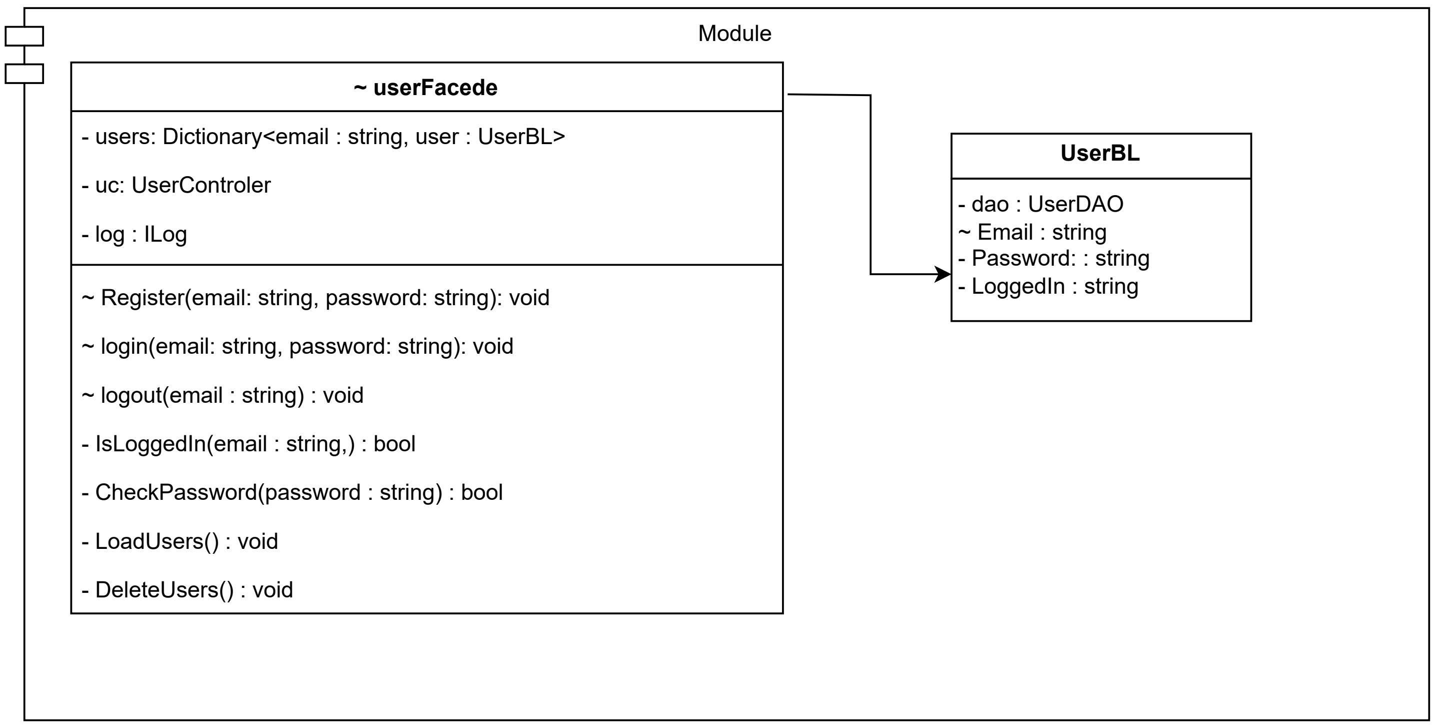
- dao : BoardDAO
~ boardID : int
~ boardName : string
~ owner : string
~ members : List<string>
~ col : CollimnBL[]
- nextTaskId : int

**ColumnBL**

- dao : ColumnDAO
~ tasks : List <Task>
~ limit : int

**TaskBL**

- dao:TaskDAO
- TaskId: String
~ Time : Date
~ DueDate : Date
~ Title : String
~ Description : string
~ ColumnOrdinal: int
~ Assigned : string?


Module
~ userFacede
- users: Dictionary<email : string, user : UserBL>
- uc: UserControler
- log : ILog
~ Register(email: string, password: string): void
~ login(email: string, password: string): void
~ logout(email : string) : void
- IsLoggedIn(email : string,) : bool
- CheckPassword(password : string) : bool
- LoadUsers() : void
- DeleteUsers() : void
UserBL
- dao : UserDAO
~ Email : string
- Password: : string
- LoggedIn : string

# Business Layer Changes

### boardFacede

**new function -**

getBoardById
loadBoards - fetch the data from the db layer.
deleteAllBoards - clear the related data in the db layer.
transferOwnership,assignTask - due to new requirements.
getUserBoards - able the user get all it's related boards.
joinBoard,LeaveBoard - let the user the ability to join and leave each board.

nextBoardId - in order to keep track and give a unique id for each board.

### userFacede

**new function -**

Logout - there was not any possibility to log out
IsLoggedIn - in order to check before users action

**new variable -**

log - in order to log important information like exceptions and object creation.

### UserBL

**deleted variable -**

boards- to lower the capling between the user module and boards and task module, we change the way we store the boards in the board facede to in order to not be needed to use the user module for board related actions.

**new variable -**

dao - in order to connect with the data access layer

### BoardBL

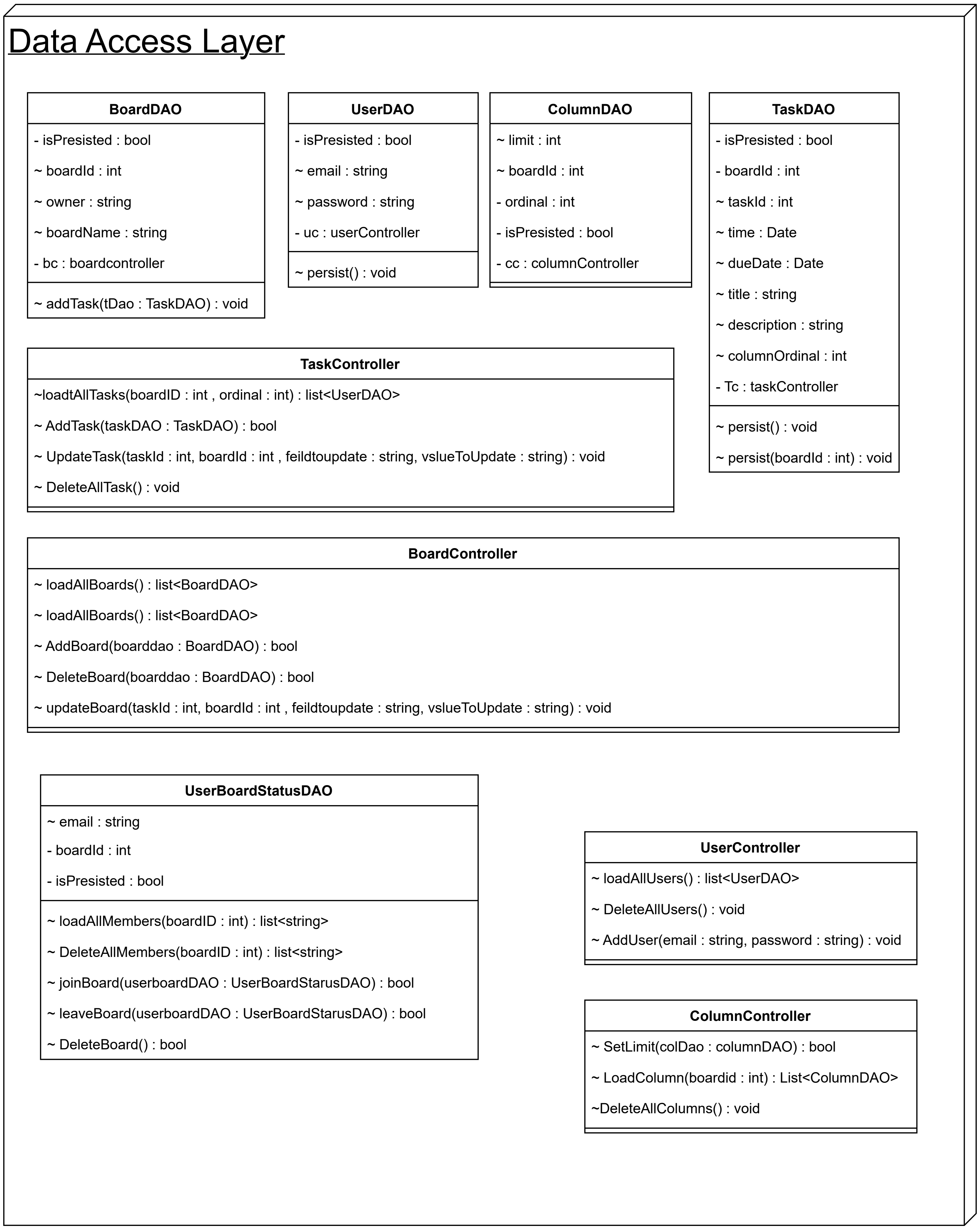
**new variable -**

dao - in order to connect with the data access layer
owner - due to new requierment
boardId - in order to manage the boards and have uniqe key.

### TaskBL

**new variable -**

dao - in order to connect with the data access layer
assign - due to new requierment



## Data Access Layer

**BoardDAO**
- isPresisted : bool
~ boardId : int
~ owner : string
~ boardName : string
- bc : boardcontroller

~ addTask(tDao : TaskDAO) : void

**UserDAO**
- isPresisted : bool
~ email : string
~ password : string
- uc : userController

~ persist() : void

**ColumnDAO**
~ limit : int
~ boardId : int
- ordinal : int
- isPresisted : bool
- cc : columnController

**TaskDAO**
- isPresisted : bool
- boardId : int
~ taskId : int
~ time : Date
~ dueDate : Date
~ title : string
~ description : string
~ columnOrdinal : int
- Tc : taskController

~ persist() : void
~ persist(boardId : int) : void

**TaskController**
~loadtAllTasks(boardID : int , ordinal : int) : list<UserDAO>
~ AddTask(taskDAO : TaskDAO) : bool
~ UpdateTask(taskId : int, boardId : int , feildtoupdate : string, vslueToUpdate : string) : void
~ DeleteAllTask() : void

**BoardController**
~ loadAllBoards() : list<BoardDAO>
~ loadAllBoards() : list<BoardDAO>
~ AddBoard(boarddao : BoardDAO) : bool
~ DeleteBoard(boarddao : BoardDAO) : bool
~ updateBoard(taskId : int, boardId : int , feildtoupdate : string, vslueToUpdate : string) : void

**UserBoardStatusDAO**
~ email : string
- boardId : int
- isPresisted : bool

~ loadAllMembers(boardID : int) : list<string>
~ DeleteAllMembers(boardID : int) : list<string>
~ joinBoard(userboardDAO : UserBoardStarusDAO) : bool
~ leaveBoard(userboardDAO : UserBoardStarusDAO) : bool
~ DeleteBoard() : bool

**UserController**
~ loadAllUsers() : list<UserDAO>
~ DeleteAllUsers() : void
~ AddUser(email : string, password : string) : void

**ColumnController**
~ SetLimit(colDao : columnDAO) : bool
~ LoadColumn(boardid : int) : List<ColumnDAO>
~DeleteAllColumns() : void