RESPONSE_JSON
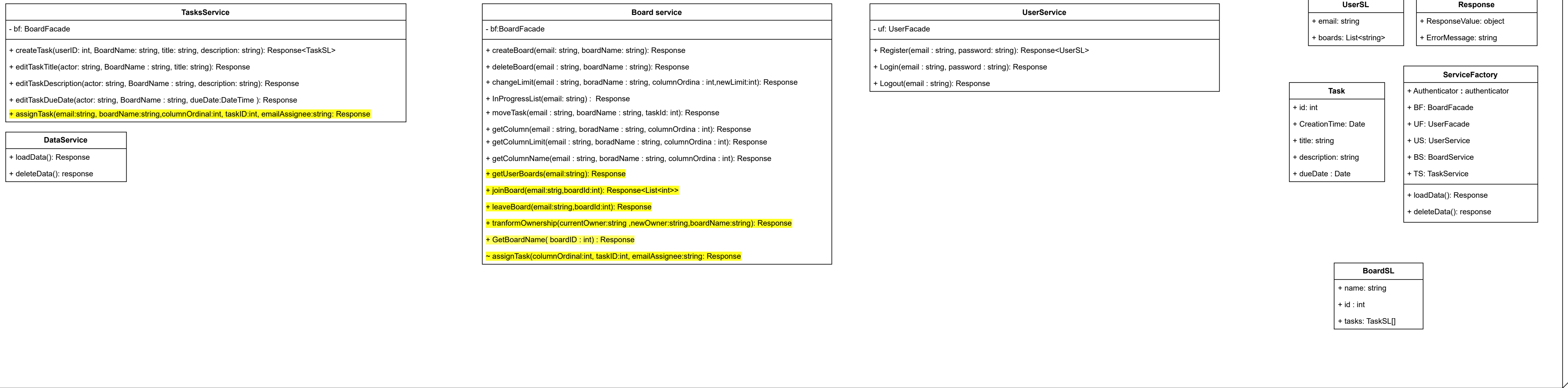{
  ErrorMessage: string,
  ReturnValue: string|number|object
}

Visibility

+ public

- private

protected

~ internal

Service Layer

TasksService

- bf: BoardFacade

+ createTask(userID: int, BoardName: string, title: string, description: string): Response<TaskSL>
+ editTaskTitle(actor: string, BoardName : string, title: string): Response
+ editTaskDescription(actor: string, BoardName : string, description: string): Response
+ editTaskDueDate(actor: string, BoardName : string, dueDate:DateTime): Response
+ assignTask(email:string, boardName:string,columnOrdinal:int, taskID:int, emailAssignee:string: Response

DataService

+ loadData(): Response
+ deleteData(): response

Board service

- bf:BoardFacade

+ createBoard(email: string, boardName: string): Response
+ deleteBoard(email: string, boardName : string): Response
+ changeLimit(email : string, boradName : string, columnOrdina : int,newLimit:int): Response
+ InProgressList(email: string): Response
+ moveTask(email : string, boardName : string, taskId: int): Response
+ getColumn(email : string, boradName : string, columnOrdina : int): Response
+ getColumnLimit(email : string, boradName : string, columnOrdina : int): Response
+ getColumnName(email : string, boradName : string, columnOrdina : int): Response
+ getUserBoards(email:string): Response
+ joinBoard(email:strig,boardId:int): Response<List<int>>
+ leaveBoard(email:string,boardId:int): Response
+ tranformOwnership(currentOwner:string ,newOwner:string,boardName:string): Response
+ GetBoardName(boardID : int) : Response
~ assignTask(columnOrdinal:int, taskId:int, emailAssignee:string: Response

UserService

- uf: UserFacade

+ Register(email : string, password: string): Response<UserSL>
+ Login(email : string, password : string): Response
+ Logout(email : string): Response

UserSL

+ email: string

+ boards: List<string>

Response

+ ResponseValue: object
+ ErrorMessage: string

Task

+ id: int

+ CreationTime: Date

+ title: string

+ description: string

+ dueDate : Date

ServiceFactory

+ Authenticator : authenticator

+ BF: BoardFacade

+ UF: UserFacade

+ US: UserService

+ BS: BoardService

+ TS: TaskService

+ loadData(): Response
+ deleteData(): response

BoardSL

+ name: string

+ id : int

+ tasks: TaskSL[]

Business Layer

Board Module

~ BoardFacade

- boards: Dictionary<int, BoardBL>
- at: Authenticator
- taskIDCuonter: int
- boardIDCuonter: int
- bc: BoardController
- buc: BoardUserControler
- userToBoards: Dictionary<string, List<int>>

~createBoard(email: string, boardName: string): BoardBL
~ deleteBoard(email: string, boardName : string): void
~ changeLimit(email : string, boradName : string, columnOrdina : int,newLimit:int): void
~ moveTask(email : string, boardName : string, taskId: int): void
~ createTask(email: string, BoardName: string, title: string, body: string, dueDate): TaskBL
~ editTaskTitle(email: string, BoardName: string, title: string): void
~ editTaskDescription(email: string, BoardName: string, newDescription: string): void
~ editTaskDueDate(email: string, BoardName: string, newDueDate): void
~ getColumn(email : string, boradName : string, columnOrdina : int): List<TasksSLS>
~ getColumnLimit(email : string, boradName : string, columnOrdina : int): int
~ InProgressList(email : string) : List<TaskBL>
~ getColumnName(email : string, boradName : string, columnOrdina : int): string
~ getUserBoards(email:string): List<int>
~ joinBoard(email:strig,boardId:int): void
~ leaveBoard(email:string,boardId:int): void
~ getBoardName(boardId:int) : string
+ tranformOwnership(currentOwner:string ,newOwner:string,boardName:string): void
~ assignTask(email:string, boardName:string,columnOrdinal:int, taskID:int, emailAssignee:string: void
- GetBoardId(email :string, boardName : string) : int
~ deleteData(): void
~ loadData(): void

~ BoardBL

- dao: BoardDAO
- TC : TaskController
- owner: string
- name: string
- id : int
- backlog: List<TaskBL>
- backlogLimit: int
- inProgress : List<TaskBL>
- inProgressLimit: int
- done : List<TaskBL>
- doneLimite : int
- members: List<string>

~ createTask(email: string, BoardName: string, title: string, body: string, dueDate): TaskBL
~ moveTask(email : string, boardName : string, taskId: int): void
- getTask(id:int): TaskBL
~ editTaskTitle(email: string, BoardName: string, title: string): void
~ editTaskDescription(email: string, BoardName: string, newDescription: string): void
~ editTaskDueDate(email: string, BoardName: string, newDueDate): void
~ changeLimit(columnOrdina : int,newLimit:int): void
~ GetLimit(col:int): int
~ GetColumn(col: int) : LinkedList<TaskBL>
~ DeleteBoard(email : string) : void
~ DeleteData(): void
~ joinBoard(email:strig): void
~ leaveBoard(email:string): void
~ getBoardName(boardId:int) : string
~ tranformOwnership(currentOwner:string ,newOwner:string): void
~ assignTask(columnOrdinal:int, taskId:int, emailAssignee:string: void

~ TaskBL

~ dao: TaskDAO
- id: int
- title: string
- description: string
- creationDate: Date
- dueDate : Date
- assignTo: string

~ assignTask(emailAssignee:string): void
~ MoveTask(email: string, col: int): void

UserModule

~ UserFacade

- uc: UserController
- users: List<UserBL>
- at: Authenticator

~ register(email : string, password: string): UserBL
~ Login(email : string, password : string): void
~ Logout(email : string): UserBL
~ loadData(): void
~ deleteData(): void

~ UserBL

~ dao: UserDAO
- email: string
- password: string

~ ChackPasswordMatch(password: string): bool
~ cheackValidPassword(password: string): bool

Module

Authenticator

- usersList: List<usersID>

~ disconect(email : string): void
~ isConect(email : string): boolean
~ connect(email : string): void

Data Access Layer

~ BoardDAO

- BoardID: int
- BoardName: string
- owner: string
- DoneLimit: int
- BacklogLimit: int
- InprogressLimit: int
- BC: BoardController
- BUC: BoardUserController
- isPersisted: bool = false

~ joinBoard(email:string): void
~ GetMember() : List<string>
~ leaveBoard(email:string): void
~ DeletBoard: void
~ Persist(): void

~ TaskDAO

- id: int
- BoardID: int
- title: string
- description: string
- state : int
- creationDate: Date
- dueDate : Date
- asignTo : string
- TC: TaskController
- isPresisted:bool = False

~ Persist(): void

~ BoardController

- _connectionString : string
- _tableName : string
- TableName : string = "Boards"

~ GetAllBoards(): List<BoardDAO>
~ Insert(board: BoardDAO) : bool
~ Update(idL int , attribuateName: string, attribuateValue: string): bool
~ Delete(id:int): bool
~ Clear() : bool

BoardUserController

- _connectionString : string
- _tableName : string
- TableName : string = "UserToBoards"

~ GetAllMembers(BoardID: int):list<string>
~ GetBoardByUser(email : string): list<int>
~ Insert(BoardId: int, emial: string) : bool
~ Delete(BoardID: int, emial: string): bool
~ Clear() : bool

~ TaskController

- _connectionString : string
- _tableName : string
- TableName : string = "Tasks"

~ GetAllTask(BoardID: int): List<TaskDAO>
~ Insert(Task: TaskDAO) : bool
~ Update(idL int , attribuateName: string, attribuateValue: string): bool
~ Delete(BoardID:int): bool
~ Clear() : bool

~ UserDAO

~ email: string
~ password: string
- isPersisted: bool = false
~ UserEmailColumnName : string = "Email"
- UC: UserController

~ Persist(): void
~ register(email:string,password:string): void
~ UserPassColumnName : string = "Password"

~ UserController

- _connectionString : string
- _tableName : string
- TableName : string = "User"

~ GetAllUsers(): List<UserDAO>
~ Insert(user: UserDAO) : bool
~ Clear() : bool

changes

all the added requirements are marked in yellow.

Additionally we move the list of boards in the userBL to a Dictionary at the Boardfacad for better cohesin