Presentation Layer

Service Layer

UserService
+ changePassword(username: string , oldP: string, newP: string): json
+ createUser(username: string, password: string): json
+ login(username: string, password: string): json
+ getBoard(BoardName: String): json
+ getInProgress(): json
+ logout(userEmail: string):json

BoardService
+ createBoard(Name: string, userEmail: string)
+ addTask(title: String, Description: String, DueDate: String): json
+ nextState(email: string, boardName: string, State: int, TaskID: int): json
+ deleteBoard(Name: string, userEmail: string)
+ SetColumnLimit(ColName: string, BoardName: string, ColState: int)
+ SetColumnLimit(ColName: string, BoardName: string, ColState: int, NewLimit: int)

TaskService
+ createTask(Title: string, Description: string, DueDate: string): json
+ EditTitle(newTitle: string, user: User, board: string): json
+ EditDueDate(newDate: string, user: User, board: string): json
+ EditDescription(newBody: string, user: User, board: string): json
+ GetState(string: username, string: BoardName): json

Business Layer

UserController
- users: Dictionary<string, User>
- loggedIn: Collection
- TotalNumOfUsers: int

+ CreateUser(password: string, email: String): void
+ DeleteUser(email: string): void
+ UserExists(email: string): boolean
+ GetUser(email: string): User
+ validatePassword(user: User, password: String): boolean
+ login(email: String, password: String): void
+ logout(user: User): void

Board
- Backlog(Collection)
- InProgress(Collection)
- Done(Collection)
+ Name(String)
- maxTasks: int = -1
- numTasks: int
- List<string(usernames)>
- Owner(string)
-BoardID: int

+ addTask(title: String, Description: String, DueDate: String): void
+ changeState(TaskD: int, Assignee: string): void
+ getters
+ setters

User
+ USERNAME: string
- password: string

+ ChangePassword(oldP: string, newP: string): void
+ ValidatePassword(password: string): bool

Task
- Title(String)
- DueDate(String)
+ CREATION DATE(DateTime)
- Description(newBody: string): void
- State(int)
+ TaskID(int)
- Assignee(string)

+ EditTitle(newTitle: string): void
+ EditDueDate(newDate: String): void
+ EditDescription(newBody: string): void
+ EditAssignee(newAssignee: string):void
+ Getters
+ Setters

BoardController
- Dictionary<string(username), <string(boardname), Board> >
- Dictionary<board_id(int), Board>
- TotalNumOfBoards: int

+ Exists(Name: String): Boolean
+ createBoard(Name: String): void
+ deleteBoard(Name: String): void
+ getBoard(Name: String): void
+ changeMax(newMax: Int): void
+ DeleteBoard(owner: string, BoardName: string): void
+ JoinBoard(Joiner: username, BoardOwner: string, BoardName: string): void
+ LeaveBoard(Leaver: string, BoardName: string): void
+ IsOwner(BoardID: int, (OR) BoardName: string): bool
+ ChangeOwner(NewOwner: string, OldOwner: string,)

Data Access Layer

TaskDTO
- Title: String
- DueDate: String
- CREATION DATE: DateTime
- Description: string
- State: int
- TaskID: int (PK)
- Assignee: string (FK)

+ EditTitle(newTitle: string): void
+ EditDueDate(newDate: String): void
+ EditDescription(newBody: string): void
+ EditAssignee(newAssignee: string):void

BoardDTO
- Backlog: List<TaskDTO>
- InProgress: List<TaskDTO>
- Done: List<TaskDTO>
- Name: String
- maxTasks: Array<int>
- numTasks: Array<int>
- Users: List<string>
- Owner: string (FK)
- BoardID: int (PK)

+ addTask(title: String, Description: String, DueDate: String): void
+ changeState(TaskD: int, Assignee: string): void
+ DeleteTask
+ AddUser(): void
+ RemoveUser(): void
+ DeleteSelf():void

UserDTO
- Username: string (PK)
- Password: string

+ ChangePassword(NewPassword: string)

UserDTOMapper
- List<UserDTO>

+ CreateUser(password: string, email: string): bool
+ DeleteUser(email: string): bool
+ ChangePassword(email: string, NewPassword: string)
+ LoadUsers(): void

BoardDTOMapper
- List<BoardDTO>

+ CreateBoard(password: string, email: String): void
+ DeleteBoard(email: string): void
+ ChangeOwner(): void
+ ChangeMax(): void
+ LoadBoards(): void

Should we add an enum of owner/user for each DTO?

BoardUserDTO
- Username: string (FK of UserDTO, PK)
- BoardID: int (FK of BoardDTO, PK)

+ AddUser(): void
+ RemoveUser(): void
+ DeleteBoard(): void

BoardUsersMapper
- List<BoardUserDTO>

+ CreateBoard(password: string, email: String): void
+ DeleteBoard(email: string): void
+ AddUser(): void
+ RemoveUser(): void

Association
Inheritance
Realization / Implementation
Dependency
Aggregation
Composition

Aggregation: child can exist without the parent

Composition: child cannot exist without the parent

From Wikipedia
https://en.wikipedia.org/wiki/Class_diagram

Visibility
+ public
- private
protected