#### UserService

+ user: UserController

- + LogIn(email: String, password : String): ison
- + Register(email: String, password: String): json + ChangePassword(email: String, newPassword:

String): json

- + LogOut(email: String): json
- + LoadData():ison
- + AddBoard(email: String, boardTitle: String):json
- + RemoveBoard(email: String, title: String): json
- + GetInProgressTasks(email: String):ison IsValidPassword(password: String): bool

- TaskService
- + CreateTask(email: string, boardName: string, title: string, description: string, dueDate: DateTime):json
- + UpdateDueDate(email: String, boardName: String, taskID: int, newDueDate:DateTime): json + UpdadeTitle(email: String, boardName: String, taskId: int, newTitle: String): json
- + UpdateDescription(email: String, boardName: String, taskId: int newDescription: String) json

#### **BoardService**

+userController: UserController

- + CreateBoard(email: String, boardTitle: String):json
- + AdvanceTask(email: String, taskID: int): ison
- + SetColumnLimit(email: String, boardTitle: String, columnOrd: int, limit: int): json
- + GetColumn(email: string, boardName: string, columnOrd: int):ison
- + changeOwner(email: string): ison

### **FactoryService**

userController: UserController - boardService: BoardService - taskService: TaskService userService: UserService

#### Buisness Layer

#### User

- + email: Strina + password: String + connected: bool + dto: UserDTO
- + isConnected(): bool
- + logIn(): string
- + logOut(): void
- + GetInProgressTasks(email: String): List<Task>
- Persist(): Void
- + ToDalObject(): User + AddBoard(string): Void
- + DeleteBoard(string): Void

# **BoardController**

- + boards: Dictionary<int, Board>
- users: UserController
- + boardDTOController: BoardControllerDTO
- + AddBoard(email: String, boardName: String): void
- + GetBoard(BoardId: int): Board
- + RemoveBoard(email: string, boardName: string)
- + GetInProgressTasks(email: String): List<Task>
- LoadData(): void

#### Board

- + name: String + boardID: int
- owner: User
- + users: List<string> meaning email
- + columns: Column []
- + currentTaskID: int
- + dto: BoardDTO
- + AddTask(email: string, title: string, description: string, dueDate: string):void
- + GetTask(taskID: int): Task InPrograssTasks(): List<Task>
- SetMaxLength(maxLength: int, columnOrd: int): void
- + AdvanceTask(taskTitle: String, taskID: int, columnOrd: int): void
- + Getcolomn(columnOrd: int) : List<Task>
- + changeOwner(email: string): void
- + removeTask (): void
- + JoinBoard(email: string, BoardId: string): void
- + RemoveUser(email: string, BoardId: string): void

#### Task

- taskId: int
- creationTime: String title: String
- dueDate: DateTime
- description: string
- + assignee: string
- + dto: TaskDTO
- IsValidTitle(title: string): bool
- IsValidDescription(description: string): bool
- EditDescription(TaskTitle: String, description: String):void + EditDueDate(TaskTitle: String, date: DateTime):void
- + EditTitle(TaskTitle: String, newTitle: String): void
- + SetAssignee(email: string):void
- + isAssigned(): bool
- + isAssignee(): bool

#### liserController

+users: Dictionary < String , String >

- + CreateUser(email: String, password: String): void LoadAllUsers(): void
- + RemoveBoard(email: String, boardTitle: String): void
- + GetInProgressTasks(email: String): List<Task>

#### Response

- + ErrorMessage: string + ReturnValue : object
- Response(string msq):
- RespnseObj(Object obj): void

## Column

- + tasks: Dictionary<int, Task>
- + length: int
- + maxLength: int + dto:
- + SetMaxLength(maxLength: int): void
- + AddTask(taskTitle: String, taskID: int, columnOrd: int): v

#### Data-Access layer

# UserDTO

- + email: String
- + password: String
- + connected: bool + dto: UserDTO
- + isConnected(): bool
- + logIn(): string
- + logOut(): void + GetInProgressTasks(email: String): List<Task>
- Persist(): Void
- + ToDalObject(): User + AddBoard(string): Void
- + DeleteBoard(string): Void

# BoardDTOMapper

+ LoadBoards(): List<BoardDTO>

# TaskDTO

- taskId: int
- creationTime: String
- title: String
- dueDate: DateTime
- description: string
- + assignee: string + dto: TaskDTO
- IsValidTitle(title: string): bool

# **BoardDTO**

- + name: String
- + boardID: int
- owner: User + users: List<string> meaning email
- + columns: Column []
- + currentTaskID: int + dto: BoardDTO
- + AddTask(email: string ,title: string, description: string, dueDate: string):void + GetTask(taskID: int): Task
- · InPrograssTasks(): List<Task> SetMaxLength(maxLength: int, columnOrd: int): void
- + AdvanceTask(taskTitle: String, taskID: int, columnOrd: int): void + Getcolomn(columnOrd: int) : List<Task>
- + changeOwner(email: string): void
- + removeTask (): void
- + JoinBoard(email: string, BoardId: string): void
- + RemoveUser(email: string, BoardId: string): void

### **BoardUserDTO**

- + email: string (מה זה FK, PK)
- + BoardId: int

- + AddTask(email: string, title: string, des
- + GetTast(taskID: int): Task InPrograssTasks(): List<Task>
- SetBackLogMaxLength(maxLength: int)
- SetInProgressMaxLength(maxLength: in
- SetDoneMaxLength(maxLength: int): vc
- + AdvanceTask(taskTitle: String, taskID:

- IsValidDescription(description: string): bool + EditDescription(TaskTitle: String, description: String):void + EditDueDate(TaskTitle: String, date: DateTime):void + EditTitle(TaskTitle: String, newTitle: String): void + SetAssignee(email: string):void

+ isAssigned(): bool + isAssignee(): bool

ואם צריך לעשות משהו על היוזרים בבורד , הוא שם בה נגיד את הפונקציה של לחסום יוזר

# BoardUserMapper

+ Getcolomn(columnOra: Int) : List< lask GetBackLog(): List<Task> - GetInProgress(): List<Task> - GetDone(): List<Task>

# : הערות שלי

- 3. איזה טיפוס כדאי לשמור את הבורדם שכל יוזר נמצא בהם לדעתי רשימה רק של ID 4. לעבור על השדות ולהבין בדיוק מה צריך להיות פמבי ומה פרטי
- 5. איזה שדה לשים לבעל הלוח , האם הוא צריך להחזיק את כל היוזר או שעדיף שלא
  - 6. לעבור על כל המילונים ולחשוב אם צריך רשימה שמחזיקה רק
    - 7. לעשות getters אם נצטרך נגיד של משימה וכאל ה
- 8. האם הלוח או את שם הלוח או את ברימוב לקבל את שם הלוח או את ה

- explanations to the changes:
  1. delete the class "TaskController" to improve the cohesion and the coupling.
- 2. change the Task's fields to private to improve privacy between classes.
- delete the class "BoardController" to improve the cohesion and the coupling.
   changing class "UserController" from internal to public to get access to the object in the service layer.
   adding "taskID" fiels to Board class.

- adding Response class for an ordered Json response.
   creating FactoryService class improve cohesion in ServiceLayer.