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

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

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):json
+ GetColumnLimit(ColName: string, BoardName: string, ColState: int) :json
+ SetColumnLimit(ColName: string, BoardName: string, ColState: int, NewLimit: int):json
+ JoinBoard(boardId: int ,userEmail: string): json
+ LeaveBoard(boardId: int, userEmailLeaving: string): json
+ AssignTask(userEmailToAssign: string,boardName: string, columnOrdinal: int, userEmailAssigning: string, taskId: int): json
+ TransferOwnership(currentOwnerEmail :string,newOwnerEmail: string,boardName:string)

Changes Made after submitting Milestone 1:
Created:
IsLegalPassword- in order to validate the password
IsValidEmail- To validate the email address at registration
IsHebrew - to validate email
HasAnyBoard- Check if a BoardController is empty - for exception handling purposes
HasThisBoard- Checks if a BoardController has a board - for exception handling purposes
Added getters and setters in Board
GetColList - in order to get all tasks in a given column in a board
GetNameOrdinal - Returns the name of a column given its id in order to fulfill GradingService requirements
GetAllInProgress - See design changes for explanation

Design changes:
A User object no longer holds a collection of their boards and Tasks.
The class user is therefore connected only to the UserConroller, which limits our duality.
The checks in order to get a user's board and tasks are preformed through calling an instance of UserController.IsLoggedIn(). This allows a better (lower) duality between the classes.
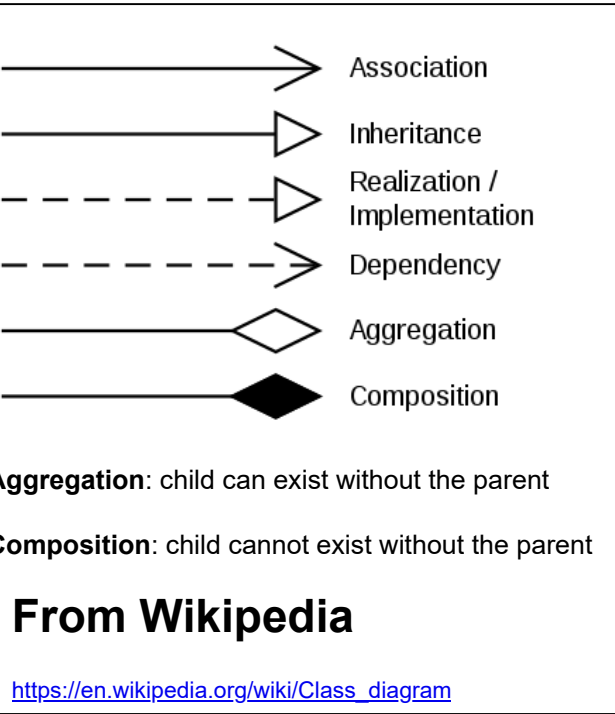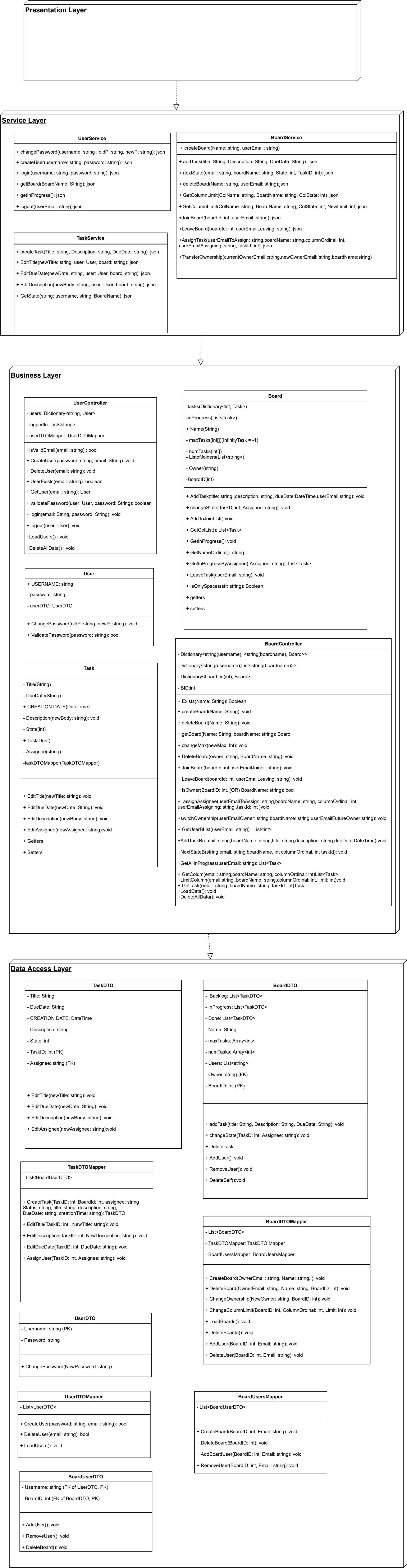
Setting the max number of tasks in now done from the Board (rather than Task).This is done because
it is an attribute of a board rather than a task. It can be said that initially we made a design mistake.
AddTask - same as max tasks.

Changes Made before starting Milestone 2 (2/6):
Created:
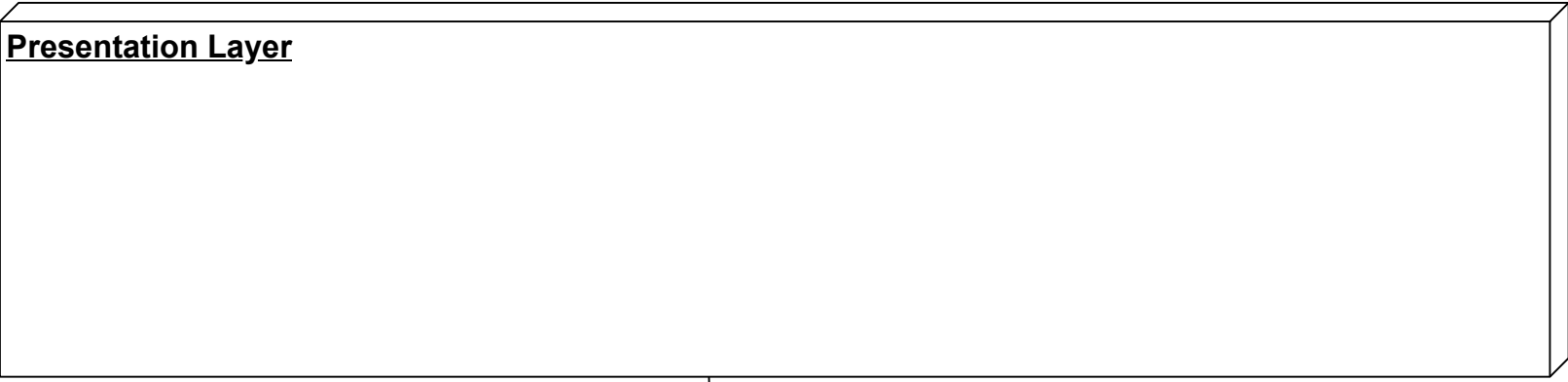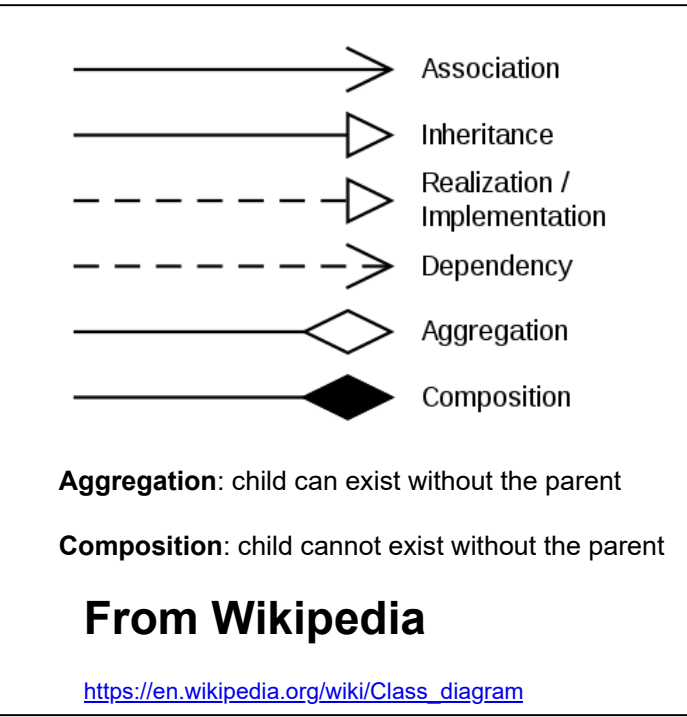lNew methods for all new functional requirements. Data Access Layer DTO's and Mappers.

Design changes:
None

Business Layer

UserController
- users: Dictionary<string, User>
- loggedIn: List<string>
- userDTOMapper: UserDTOMapper

+IsValidEmail(email: string) : bool
+ 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
+ LoadUsers() : void
+DeleteAllData() : void

User
+ USERNAME: string
- password: string
- userDTO: UserDTO

+ 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)
-taskDTOMapper(TaskDTOMapper)

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

Board
-tasks(Dictionary<int, Task>)
-inProgress(List<Task>)
+ Name(String)
- maxTasks(int[])\InfinityTask = -1)
- numTasks(int[])
- ListofJoiners(List<string>)
- Owner(string)
-BoardID(int)

+ AddTask(title: string ,description: string, dueDate:DateTime,userEmail:string): void
+ changeState(TaskD: int, Assignee: string): void
+ AddToJoinList():void
+ GetColList(): List<Task>
+ GetInProgress(): void
+ GetNameOrdinal(): string
+ GetInProgressByAssignee(Assignee: string): List<Task>
+ LeaveTask(userEmail: string): void
+ IsOnlySpaces(str: string): Boolean
+ getters
+ setters

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

+ Exists(Name: String): Boolean
+ createBoard(Name: String): void
+ deleteBoard(Name: String): void
+ getBoard(Name: String ,boardName: string): Board
+ changeMax(newMax: Int): void
+ DeleteBoard(owner: string, BoardName: string): void
+ JoinBoard(boardId: int,userEmailJoiner: string): void
+ LeaveBoard(boardId: int, userEmailLeaving: string): void
+ IsOwner(BoardID: int, (OR) BoardName: string): bool
+ assignAssignee(userEmailToAssign: string,boardName: string, columnOrdinal: int, userEmailAssigning: string ,taskId: int)void
+switchOwnership(userEmailOwner: string,boardName: string,userEmailFutureOwner:string): void
+ GetUserBList(userEmail: string): List<int>
+AddTaskB(email: string,boardName: string,title: string,description: string,dueDate:DateTime):void
+NextStateB(string email, string boardName, int columnOrdinal, int taskId): void
+GetAllInPrograss(userEmail: string): List<Task>
+ GetColum(email: string,boardName: string, columnOrdinal: int):List<Task>
+LimitColumn(email:string, boardName: string,columnOrdinal: int, limit: int)void
+ GetTask(email: string, boardName: string, taskId: int)Task
+LoadData(): void
+DeleteAllData(): void

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

TaskDTOMapper
- List<BoardUserDTO>

+ CreateTask(TaskID: int, BoardId: int, assignee: string
Status: string, title: string, description: string,
DueDate: string, creationTime: string): TaskDTO
+ EditTitle(TaskID: int , NewTitle: string): void
+ EditDescription(TaskID: int, NewDescription: string): void
+ EditDueDate (TaskID: int, DueDate: string): void
+ AssignUser(TaskID: int, Assignee: string): void

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

+ ChangePassword(NewPassword: string)

UserDTOMapper
- List<UserDTO>

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

BoardDTOMapper
- List<BoardDTO>
- TaskDTOMapper: TaskDTO Mapper
- BoardUsersMapper: BoardUsersMapper

+ CreateBoard(OwnerEmail: string, Name: string,): void
+ DeleteBoard(OwnerEmail: string, Name: string, BoardID: int): void
+ ChangeOwnership(NewOwner: string, BoardID: int): void
+ ChangeColumnLimit(BoardID: int, ColumnOrdinal: int, Limit: int): void
+ LoadBoards(): void
+ DeleteBoards(): void
+ AddUser(BoardID: int, Email: string): void
+ DeleteUser(BoardID: int, Email: string): void

BoardUsersMapper
- List<BoardUserDTO>

+ CreateBoard(BoardID: int, Email: string): void
+ DeleteBoard(BoardID: int): void
+ AddUser(BoardID: int, Email: string): void
+ RemoveUser(BoardID: int, Email: string): void

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

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

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

Legend
→(solid)	Association
→(hollow triangle)	Inheritance
- - →(hollow triangle)	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

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

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
+ IsAssignee(): bool
+ IsAssigned(): bool

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

+ addTask(title: String, Description: String, DueDate: String): void
+ changeState(TaskD: int, Assignee: string): void
+ AddToJoinList():void
+ GetColList(): void
+ GetInProgress(): void
+ GetNameOrdinal(): void
+ GetInProgressByAssignee(string: Assignee)
+ LeaveTask(string: email): void
+ getters
+ setters

Board
- Backlog(Column)
- InProgress(Column)
- Done(Column)
+ Name(String)
- ListofJoiners(List<string>)
- Owner(string)
-BoardID: int
- Assignees(Dictionary<string, Tasks>) // username, tasks they are assigned to

+ addTask(title: String, Description: String, DueDate: String): void
+ changeState(TaskD: int, Assignee: string): void
+ Join():void
+ GetColumnTasks(username: string, column: enum): Column
+ GetColumnTaks(column: enum): Column
+ LeaveTask(string: email): void
+ ChangeMaxTasks(column: enum): void
+ ChangeAssignee(currentAssignee: string, newAssignee: string): void
+ getters
+ setters

BoardController
- Participants(Dictionary<string, List<Board>>) // username, list<Boards> user is in
- Boards(Dictionary<board_id(int), Board>)
- TotalNumOfBoards: int

+ Exists(Name: String): Boolean
+ createBoard(Name: String): void
+ deleteBoard(Name: String): 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,)
+

Column
- Tasks(List)
- maxTasks: int = -1

+ addTask(title: String, Description: String, DueDate: String): void
+ Remove Task(int: TaskId): void
+ getters
+ setters

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)

BoardDTOMapper
- List<BoardDTO>

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

UserDTOMapper
- List<UserDTO>

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

BoardUsersMapper
- List<BoardUserDTO>

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