Business Layer

UserController

- users: Dictionary<string, User>
- userDAO: UserDAO
- boardsMembersDAO: BoardsMembersDAO

+ userRegister(email:String, password:String):void
+ usersAvailability(email:String):bool
+ userRemove(email:String):string
+ usersPasswordValid(password:String):string
+ ValidateLegalEmail(email: string):bool
+ getUsersList(): List<User>
+ logOut(email: string): string
+ getUser(email: string): User
+ login(email: string, password: string): string
+ deleteData(): void
+ getUser(email: string, constructor: int): User
+ RemoveBoard(Id: int): void

User

+ email: String
+ password: String
+ loggedIn: bool
+ asiggnedTasks: List<Taskk>
- userDTO: UserDTO
- userDAO: UserDAO

+ userEditPassword(OldPassword:String,newPassword:String):string
+ getEmail(): String
+ userMatchPassword(password:string): String
+ logIn(password: string): bool
+ logOut(): bool
+ inProgressTasks(): List<Task>
+ addAssignee(task: Task): void
+ getLogInStatus(): bool
+ RemoveUser(): void
+ removeBoardsTasks(boardId: int): void
+ leaveBoardsTasks(boardId: int): void

BoardController

- boards: Dictionary<int, Board>
- boardIdCounter: int
- userController: UserController
- boardDAO: BoardDAO

+ loadData(): void
+getBoard(BoardName:String): Board
+ loginStatus(email: string): void
+userRemove(email: string): void
+ getAllBoardNames(): List<string>
+ getUserBoard(email: string) List<int>
+ getBoardByName(boardName: string, email: string): Board
+ joinMember(email: stirng, boardId: int): void
+ leaveBoard(email: string, boardId: int): void
+ deleteData(): void
+ transferOwnership(currentOwnerEmail: string, newOwnerEmail: string, boardName: string): void
+ addBoard(boardName:String, userEmail: string):Board
+ removeBoard(userEmail: string, boardName:String): void
+ removeBoard(id: int, email: stirng): void
+ getBoard(Id:int):Board

Task

+ title: String
+ status: String
+ description:String
+ dueDate:Datetime
+ creationDate:Datetime
+ taskId: int
+ assigne: string
- taskDTO: TaskDTO
- taskDAO: TaskDAO

+ taskEditTitle(newTitle:String):string
+ taskEditDescription(newDescription:String):string
+ taskEditDueDate(newDueDate:Datetime):string
+ setAssignee(newAssignee: string): void

Board

+ colunms: List<Column>
+ name: String
+ boardId: int
+ counterTaskId: int
+ Owner: string
+ joinedMembers: Dictionary<string , User>
- boardDAO: BoardDAO
- boardDTO: BoardDTO
- columnDAO: ColumnDAO
- boardsMembersDAO: BoardsMembersDAO
- boardsMembersDTO: BoardsMembersDTO

+ GetColumn(index: int): Column
+ boardEditName(newName:String): void
+taskMove(taskId: int, columnIndex: int):bool
+ getJoinedMembers(): List<string>
+ getUserOwner(): string
+ setOwner(email: string): void
+ addNewTask(title: string, description: string, dueDate:DateTime): void
+ joinAmember(email: string): void
+ inProgressTasks(): List<Task>
+ removeUser(email: string): bool
+ membersContain(email: string): bool
+ taskEditTitle(email: string, taskId: int, columnIndex: int, newTitle: string): void
+ taskEditDueDate(email: string, taskId: int, columnIndex: int, newDueDate: DateTime): void
+ taskEditDescription(email: string, taskId: int, columnIndex: int, newDescriptiom: string): void
+ taskMove(email: string, taskId: int, columnIndex: int): void
+ leaveBoard(email: string): void
+ getUser(email: string): User
+ setUnassignedTask(email: string, columnIndex: int, taskId: int, newAsignee: string): void
+ assignedTask(email: string, columnIndex: int, taskId: int, newAsignee: string): void

Colunm

+ tasks:List<Task>
+ limit:int
+ name:string
- columnDTO: ColumnDTO
- columnDAO: ColumnDAO
- taskDAO: TaskDAO

+ getTask(taskId:int):Task
+ taskAdd(task:Task): string
+ taskRemove(taskId: int):string
+ editLimit(newLimit: int): void

Service Layer

UserService

- boardController: BoardController
- userController: UserController

+ userRegister(email: string, password: string): json

+ userLogout(email: string): json

+ userRemove(email: string): json

+ userLogin(email:string, password:string): json

+ userEditPassword(email: string, password: string, newPassword: string): json

+ joinBoard(email: string, boardId: int): json

+ leaveBoard(email: string, boardId: int): json

+ getAllUsers(): json

+ transferOwnership(email:string, TransferTo: string, boardName: string): json

BoardService

- boardController: BoardController

+ RemoveBoard(boardId: int, removerEmail: string):json

+ addBoard(email:string,boardName:string):json

+ getBoard(email: string, boardId: int): json

+ boardEditName(email: string, boardId: int, newName: string): json

+ getAllMembers(email: string, BoardId: int): json

+ DeleteData(): json

+ LoadData(): json

FactoryService

+boardService: BoardService
+userServise: UserService
+columnService : ColumnService
+taskService : TaskService
+boardcontroller : BoardController

ColumnService

- boardController: BoardController

+ getColumn(email:string,boardId: int,columnIndex: int): json

+ getLimit(email:string, boardId: int, columnIndex:int):json

+ getColumnTasks(email: string, boardId: int, columnindex: int): json
+ editLimit(email: string, boardId: int, columnIndex: int, newLimit: int): json

+ getTasks(email: string, boardId: int, columnIndex: int): json

TaskService

- boardController: BoardController

-userController: UserController

+ taskEditDescription(email:string, boardId: int,ColumnIndex: int ,taskId: int, newDescription:String):json

+ addNewTask(email: string, boardId: int, title: string, description: string, dueDate: DateTime): json

+ taskEditTitle(email:string, boardId: int, ColumnIndex: int ,taskId: int, newTitle: string):json

+ taskEditDueDate(email:string, boardId: int, ColumnIndex: int ,taskId: int , newDueDate:DateTime): json

+ taskMove(email: string, boardId: int, columnIndex: int, taskId: int): json

+ inProgressTasks(email: string): json

+ getTask(email: string, boardId: int, columnIndex: int, taskId: int): json

+ setUnassignedTask(email: string, BoardId: int, columnIndex: int, taskId: int, newassigneeEmail: string): json
+ assignedTask(email: string, boardName: string, ColumnIndex: int, taskId: int, newAssigneeEmail: string): json

Data-Access Layer

BoardDTO

+ boardIDColumnName: string

+ boardNameColumnName: string

+ boardOwnerColumnName: string

- Id: int
+ Id: int

- name: string
+ name: stirng

- owner: string
+ owner: stirng

ColumnDTO

+ columnKeyColumnName: string

+ columnNameColumnName: string

+ columnLimitColumnName: string

+ columnBoardColumnName: string

- index: int
+ index: int

- name: string
+ name: string

- boardId: int
+ boardId: int

- limit: int
+ limit: int

BoardsMembersDTO

+ boardMembersIDColumnName: string

+ boardMembersEmailColumnName: string

- email: stirng
+ email: string

- boardId: int
+ boardId: int

DTOAbstract

controller: DAOAbstract

TaskDTO

+ taskIdColumnName: string

+ taskTitleColumnName: string

+ taskdescriptionColumnName: string

+ taskdueDateColumnName: string

+ taskCreationDateColumnName: string

+ taskColumnColumnName: string

+ taskAssigneeColumnName: string

- id: int
+ id: int

- title: string
+ title: string

- description
+ description

- dueDate: stirng
+ dueDate: string

- creationDate: string
+ creationDate: string

- column: int
+ column: int

- assigne: string
+ assigne: string

- boardId: int
+ boardId: int

userDTO

+ userEmailColumnName: string

+ userPasswordColumnName: string

- email: stirng
+ email: string

- password: string
+ password: strin

BoardDAO

- FirstColumn: sting

- SecondColumn: string

- ThirdColumn: string

---

convertReadertoObject(reader: SQLiteDataReader): BoardDTO

+ Insert(boardDTO: BoardDTO): bool

+ Delete(BoardId: int): bool

BoardsMembersDAO

- FirstColumn: sting

- SecondColumn: string

---

convertReadertoObject(reader: SQLiteDataReader): BoardsMembersDTO

+ Insert(boardsMembersDTO: BoardsMembersDTO): bool

+ Delete(email: string, BoardId: int): bool

+ Delete(email: stirng): bool

UserDAO

- FirstColumn: sting

- SecondColumn: string

---

convertReadertoObject(reader: SQLiteDataReader): UserDTO

+ Insert(userDTO: UserDTO): bool

+ Insert(email: string, password: string): bool

+ Delete(userDTO: UserDTO): bool

+ Delete(email: stirng): bool

+ Update(userDTO: UserDTO, rellevantColumnName: stirng, newValue: Object): bool

DAOAbstract

ConnectionString: string

tableName: string

---

convertReadertoObject(reader: SQLiteDataReader): DTOAbstract

+ Update(id: int, attributeName: string, attributeValue: string): bool

+ update(id: string, attributeName: string, attributeValue: string): bool

+ update(id: string, attributeName : string, attributeValue: int): bool

+ update(id: int, attributeName: string, attributeValue: int): bool

+ select(): List<DTOAbstract>

- clear(): bool

TaskDAO

- FirstColumn: sting
- SecondColumn: string
- thirdColumn: string
- fourthColumn: string
- fifthColumn: string
- sixthColumn: string
- seventhColumn: string
- eigthColumn: string

---

convertReadertoObject(reader: SQLiteDataReader): TaskDTO

convertReadertoObjectMaxId(reader: SQLiteDataReader): int

+ Insert(taskDTO: TaskDTO): bool

+ Delete(taskDTO: TaskDTO): bool

+ Update(taskDTO: TaskDTO, rellevantColumnName: string, newValue: Object): bool

+ Update(taskId: int , boardID: int, rellevantColumnName: string, newValue: int): bool

+ SelectColumnTasks(boardId: int, ColumnIndex: int): List<DTOAbstract>

+ SelectTasksNextId(boardId: int): int

UserDAO

- FirstColumn: sting

- SecondColumn: string

---

convertReadertoObject(reader: SQLiteDataReader): UserDTO

+ Insert(userDTO: UserDTO): bool

+ Insert(email: string, password: string): bool

+ Delete(userDTO: UserDTO): bool

+ Delete(email: stirng): bool

+ Update(userDTO: UserDTO, rellevantColumnName: string, newValue: Object): bool

1.	for each class in the bussiness layer, we added to it's field it's DAO and DTO class from the dal in orfer to makr changes in the database .
2.	In, the bussiness layer we changed that that the boardController has a field of userContrtoller instead of userController has a field of boardControler, we changed the cycle here that the board knows all the users. by this change we create better cohesion between the buisness classes , relativley to the new milstone Requirements.
3.	we added the loadData and deleteData function to the boardController and userController.
4.	In the bussiness layer, we removed getters and setters funciton from the uml - made get/set at the propertie
5.	In the bussined layer we moved the function : leaveBoard and joinBoard from the user class to the board class since by the new design the board know the users that belong to him.
6.	In the bussined layer we moved all the task edits function(title,description, duedate) from the user class to the board class since the cycle statrs from the board now
7.	for al the classed in the service layer, we add a field of boardControler class since everything now statrs from there.
8.	we added the loadData and deleteData to the board service class
9.	we added the setUnassignedTask to the task service class , it more compatible with his functunallity
10.	In the data access layer, we added DAO class for each class in the bunssined layer with the functions selecet, delete and update
11.	we removed unnecceserry function from the class in the DTO.
12.	in order to prevent high cuplling and circular design we removed the property "ownerOf" from the users. and instead we added the property "owner" and "joinedMembers" to the board . the new designis more suitable to the new requirments.
13	we add the UserMember DAO/DTO so we would have quick access to every user who member in each board.
14	we add A factoryService class to create all service classes and own them.