Presentation Layer

Service Layer

ServiceFactory

+ UserService: UserService
+ BoardService: BoardService
+ TaskService: TaskService
- boardController: BoardController
- userController: UserController

+ LoadData(): json
+ DeleteData(): json

UserService

- userController: UserController

+ CreateUser(email: string, password: string): json
+ Login(email: string, password: string): json
+ Logout(email: string): json
+ ChangePassword(email: string , oldPassword: string, newPassword: string): json
+ UserExists(email: string): json
+ GetUser(email: string): json

BoardService

- boardController: BoardController
- userController: userController

+ CreateBoard(userEmail: string, boardName: string) :json
+ RemoveBoard(userEmail: string, boardName: string) :json
+ GetBoard(userEmail: string, boardName: string) :json
+ GetBoard(userEmail: string, boardID: int) :json
+ GetBoard(boardID: int) :json
+ LimitColumn(userEmail: string, boardName: string, columnNumber: int, tasksLimit: int) :json
+ JoinBoard(userEmail: string, boardID: int): json
+ LeaveBoard(userEmail: string, boardName: string): json
+ TransferOwner(userEmail: string, boardName: string, newOwner: string): json
+ GetUserBoards(userEmail: string): json

TaskService

- boardController: BoardController
- userController: userController

+ ListInProgressTasks(userEmail: string) :json
+ CreateTask(userEmail: string, boardName: string, dueDate: DateTime, title: string, description: string): json
+ MoveTask(userEmail: string, boardName: string, columnNumber: int, taskID: int): json
+ EditTask(userEmail: string, boardName: string, taskID: int, dueDate: DateTime, title: string, description: string): json
+ GetTask(userEmail: string, boardName: string, taskID: int): json
+ AssignTask(userEmail: string, boardName: string, taskID: int, newAssignee: string): void

User

+ EmailAddress: string
+ IsLoggedIn: boolean

Board

+ BoardID: int
+ Name: string
+ BoardOwner: string
+ Columns: List<Column>

Response<T>

+ ErrorMessage: string
+ ErrorOccured: boolean
+ ReturnValue: json

+ ToJson(): json
+ FromJson(jsonResponse: string): Response
+ DeserializeReturnValue(): T

Task

+ TaskID: int
+ Title: string
+ CreationTime: DateTime
+ DueDate: DateTime
+ Description: string
+ AssigneeUser: string

Column

+ ColumnNumber: int
+ TasksLimit: int
+ Tasks: Dictionary<string, Task>

Business Layer

UserController

- _users: Dictionary<string, User>
- _mapper: UserMapper

+ LoadData(): void
+ DeleteData(): void
+ CreateUser(email: string, password: string): void
+ UserExists(email: string): boolean
+ GetUser(email: string): User
+ IsLoggedIn(email: string): boolean
+ ChangePassword(email: string , oldPassword: string, newPassword: string): void
+ Login(email: string, password: string): User
+ Logout(userEmail: string): void

BoardController

- _boards: Dictionary<int, Board>
- _mapper: BoardMapper

+ CreateBoard(userEmail: string, boardName: string): Board
+ RemoveBoard(userEmail: string, boardName: string): void
+ GetBoard(userEmail: string, boardName: string): Board
+ GetBoard(userEmail: string, boardID: int): Board
+ GetBoard(boardID: int): Board
+ ListInProgressTasks(userEmail: string) : List<Task>
+ GetUserBoards(userEmail: string) : List<Board>
+ JoinBoard(userEmail: string, boardID: int): Board
+ LoadData(): void
+ DeleteData(): void

Task

+ TaskID(readonly): int
+ Title(readonly): string
+ CreationTime(readonly): DateTime
+ DueDate(readonly): DateTime
+ Description(readonly): string
+ AssigneeUser(readonly): string
+ TaskDTO(readonly): TaskDTO

+ EditTask(dueDate: DateTime, title: string, description: string): void
+ ChangeAssignee(userEmail: string, newAssignee: string): void
+ Unassign(): void
+ IsAssignee(userEmail: string): boolean
- ValidTaskParameters(dueDate: DateTime,title: string, description: string): boolean

User

+ EmailAddress(readonly): string
- _password: string
+ IsLoggedIn(readonly): boolean
+ UserDTO(readonly): UserDTO

+ Login(password: string): boolean
+ Logout(): void
+ ChangePassword(oldPassword: string, newPassword: string): void
- CheckValidPassword(password: string): boolean
- CheckEmailAddress(email: string): boolean

Board

+ BoardID(readonly): int
+ BoardOwner(readonly): string
+ BoardName(readonly): string
- _columns: List<Column>
- _boardMembers: Set<string>
+ BoardDTO(readonly): BoardDTO

+ CreateTask(userEmail:string, dueDate: DateTime, title: string, description: string): Task
+ MoveTask(userEmail: string, currentColumnNumber: int, taskID: int): void
+ EditTask(userEmail: string, taskID: int,dueDate: DateTime, title: string, description: string): void
+ GetTask(taskID: int): Task
+ GetColumn(columnNumber: int): Column
+ AddMember(userEmail: string): void
+ IsBoardMember(userEmail: string): bool
+ RemoveMember(userEmail: string): void
+ TransferOwner(oldOwner: string, newOwner: string): void
+ AssignTask(taskID: int, userEmail: string, newAssignee: string): void

Column

+ ColumnNumber(readonly): int
- tasks: Dictionary<int, Task>
+ TasksLimit(readonly): int
+ ColumnDTO(readonly): ColumnDTO

+ LimitColumn(tasksLimit: int): void
+ GetTasks(): List<Task>
+ GetTasksOfAssignee(userEmail: string): List<Task>
+ AddTask(task: Task): void
+ RemoveTask(taskID: int): Task
+ GetTask(taskID: int): Task
+ TaskExists(taskID: int): boolean
+ UnassignTasks(userEmail: string): void

Data Access Layer

TaskDTO

+ TaskID: int (AI - PK)
+ Title: string
+ CreationTime: DateTime
+ DueDate: DateTime
+ Description: string
+ AssigneeUser: string
+ BoardID: int
+ ColumNumber: int

+ Insert(): void
+ Delete(): void

<abstract>
DTO

- _dalController: DalController

+ Insert(): void
+ Delete(): void

UserDTO

+ EmailAddress: string (PK)
+ Password: string

+ Insert(): void
+ Delete(): void

UserBoardDTO

+ EmailAddress: string (PK)
+ BoardID: int (PK)

+ Insert(): void
+ Delete(): void

BoardDTO

+ BoardID(readonly): int (AI - PK)
+ BoardOwner(readonly): string
+ BoardName(readonly): string
+ Columns: List<ColumnDTO>
+ BoardMembers: Dictionary<string,UserBoardDTO>

+ Insert(): void
+ Delete(): void
+ InsertMember(userEmail: string): void
+ DeleteMember(userEmail: string): void

<abstract>
DalController

_connectionString: string
_primaryKeyNames: string[]
_tableName: string

+ Select(): List<DTO>
+ Select(attributeNames: string[] ,attributeValues: string[]): List<DTO>
+ Insert(attributeNames: string[] ,attributeValues: string[]): void
+ Delete(keyValues: string[]): bool
+ DeleteAll(): bool
+ DeleteAll(tableName: string): bool
+ Update(keyValues: string[], attributeNames: string[] ,attributeValues[]: string): void
+ ConvertReaderToDTO(reader: SQLiteDataReader): DTO
+ GetMaxValue(columnName: string): int
+ DeleteSqliteSequence(): bool

ColumnDTO

+ ColumnNumber(readonly): int
+ TasksLimit(readonly): int
+ BoardID: int
+ Tasks: Dictionary<int, TaskDTO>

+ Insert(): void
+ Delete(): void

Extends

UserBoardDalController

Extends

BoardDalController

Extends

UserDalController

Extends

ColumnDalController

Extends

TaskDalController

BoardMapper

+ LoadAllBoards(): List<BoardDTO>
+ DeleteAllBoards(): void

UserMapper

+ LoadAllUsers(): List<UserDTO>
+ DeleteAllUsers(): void

Design Updates - Milestone 1

1)
Added "ServiceFactory" class – We used the factory pattern learned in class, to make sure that the services are initialized correctly, with the same instance of the UserController.

2)
Removed email from list of parameters in "Login" and "Logout" methods in User class – We call these methods using an instance of the User class, which already has the email address of the user that is logging in / out.

3)
Changed the following fields to properties and defined them as readonly - We wanted to be able to access these fields from other classes, but we didn't want them to be changed.
- "emailAddress" and "isLoggedIn" fields in User class.
- "BoardName" in Board class.
- "TaskId", "CreationTime", "DueDate", "Title", "Description" in Task class.
- "ColumnNumber" and "TaskLimit" in Column class.

4)
Added the following properties and defined them as readonly.
- "BoardID" in Board class - for having the option to identify boards using an ID.
- static "COUNT_COLUMNS" to Board class – to avoid rewriting the "magic number" 3, as the total number of columns in Board operations.

5)
Added the following fields.
- "boardCounter" to BoardController class – in order to generate the correct unique ID for each new board.
- Added "BoardID" field in Board class (Service layer) – for the same reason as the Board in business layer.

6)
Added the following methods.
- "EditTask" and "GetTask" methods to Board class – in order to check and disallow editing tasks that are placed in the last columns of the board.
- Added "CheckEmailAddress" method in User class – When creating a new user, we have to make sure that their email address is valid.

Design Updates - Milestone 2

1)
Added "ServiceFactory" class – We used the factory pattern learned in class, to make sure that the services are initialized correctly, with the same instance of the UserController.

2)
Board controller now holds a dictionary where the key is the board id instead of the user email and board name, because now board can have multiple board members.

3)
We added the Data Access Layer the connects between the Business Layer and the Database, and is responsible for storing and loading the data.

4)
We enabled a board to have multiple users instead of one. We added the "JoinBoard" and "LeaveBoard" functions that enable the users to join or leave the board.

5)
We added the "BoardOwner" field - each board can have many members but only one user who is the board owner.

6)
We added the "TransferOwner" function that enables the board owner to transfer the ownership to another board member.

7)
We added the "AssignTask" function that enables users to assign and reassign a task to any board member.