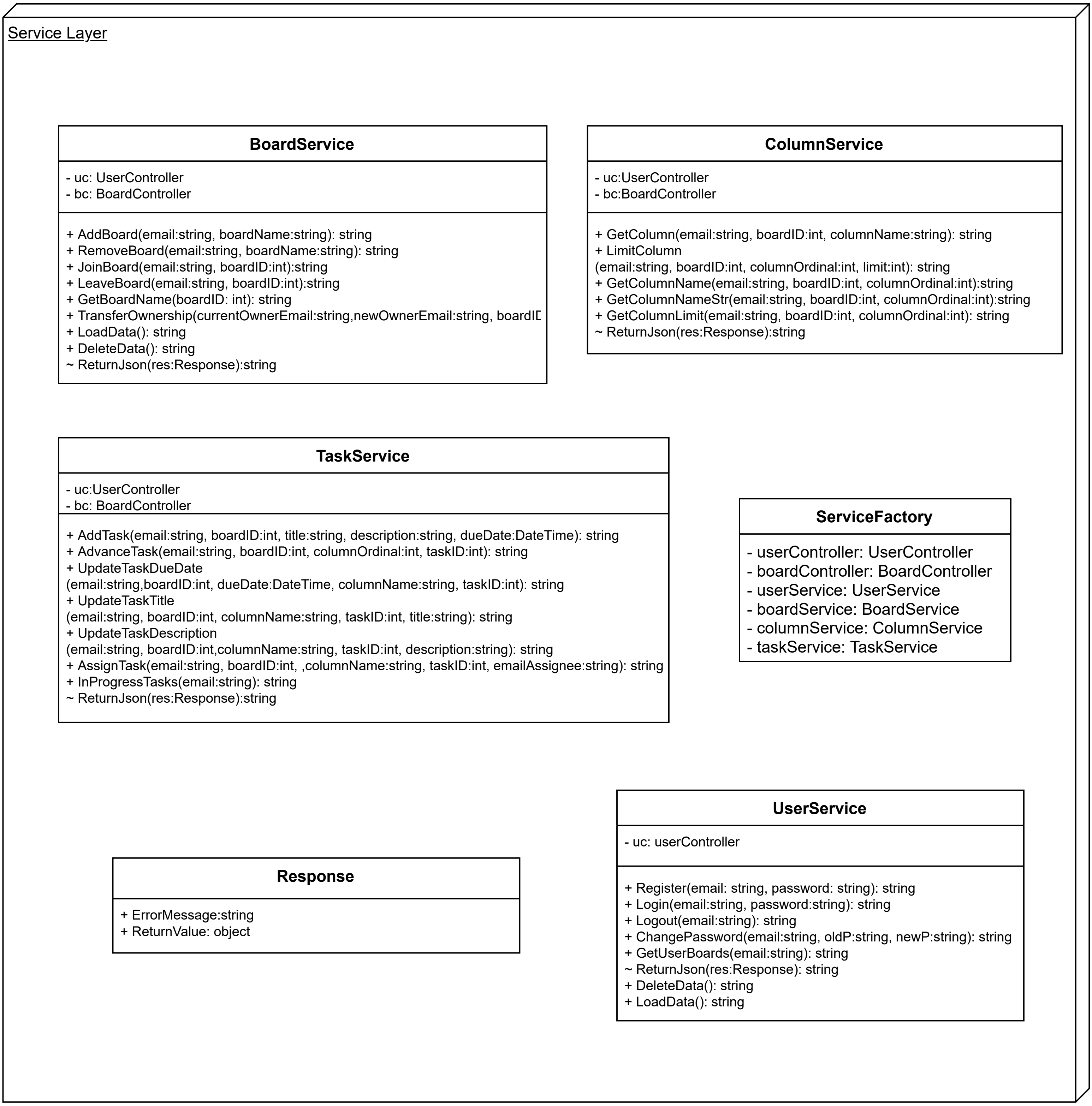


BoardService

- uc: UserController
- bc: BoardController

+ AddBoard(email:string, boardName:string): string
+ RemoveBoard(email:string, boardName:string): string
+ JoinBoard(email:string, boardID:int):string
+ LeaveBoard(email:string, boardID:int):string
+ GetBoardName(boardID: int): string
+ TransferOwnership(currentOwnerEmail:string,newOwnerEmail:string, boardID
+ LoadData(): string
+ DeleteData(): string
~ ReturnJson(res:Response):string

ColumnService

- uc:UserController
- bc:BoardController

+ GetColumn(email:string, boardID:int, columnName:string): string
+ LimitColumn
(email:string, boardID:int, columnOrdinal:int, limit:int): string
+ GetColumnName(email:string, boardID:int, columnOrdinal:int):string
+ GetColumnNameStr(email:string, boardID:int, columnOrdinal:int):string
+ GetColumnLimit(email:string, boardID:int, columnOrdinal:int): string
~ ReturnJson(res:Response):string

TaskService

- uc:UserController
- bc: BoardController

+ AddTask(email:string, boardID:int, title:string, description:string, dueDate:DateTime): string
+ AdvanceTask(email:string, boardID:int, columnOrdinal:int, taskID:int): string
+ UpdateTaskDueDate
(email:string,boardID:int, dueDate:DateTime, columnName:string, taskID:int): string
+ UpdateTaskTitle
(email:string, boardID:int, columnName:string, taskID:int, title:string): string
+ UpdateTaskDescription
(email:string, boardID:int,columnName:string, taskID:int, description:string): string
+ AssignTask(email:string, boardID:int, ,columnName:string, taskID:int, emailAssignee:string): string
+ InProgressTasks(email:string): string
~ ReturnJson(res:Response):string

ServiceFactory

- userController: UserController
- boardController: BoardController
- userService: UserService
- boardService: BoardService
- columnService: ColumnService
- taskService: TaskService

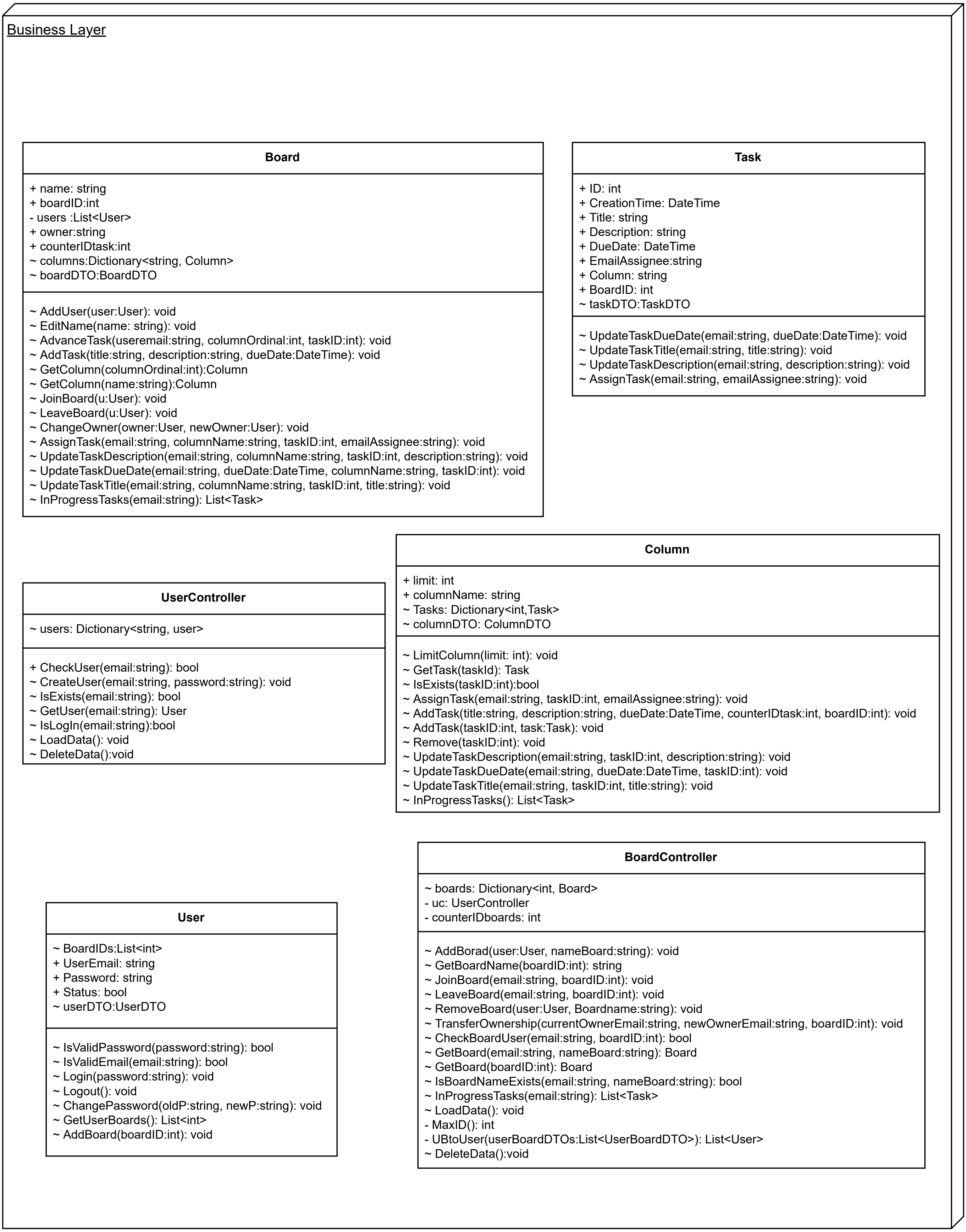
UserService

- uc: userController

+ Register(email: string, password: string): string
+ Login(email:string, password:string): string
+ Logout(email:string): string
+ ChangePassword(email:string, oldP:string, newP:string): string
+ GetUserBoards(email:string): string
~ ReturnJson(res:Response): string
+ DeleteData(): string
+ LoadData(): string

Response

+ ErrorMessage:string
+ ReturnValue: object



Business Layer

Board

+ name: string
+ boardID:int
- users :List<User>
+ owner:string
+ counterIDtask:int
~ columns:Dictionary<string, Column>
~ boardDTO:BoardDTO

~ AddUser(user:User): void
~ EditName(name: string): void
~ AdvanceTask(useremail:string, columnOrdinal:int, taskID:int): void
~ AddTask(title:string, description:string, dueDate:DateTime): void
~ GetColumn(columnOrdinal:int):Column
~ GetColumn(name:string):Column
~ JoinBoard(u:User): void
~ LeaveBoard(u:User): void
~ ChangeOwner(owner:User, newOwner:User): void
~ AssignTask(email:string, columnName:string, taskID:int, emailAssignee:string): void
~ UpdateTaskDescription(email:string, columnName:string, taskID:int, description:string): void
~ UpdateTaskDueDate(email:string, dueDate:DateTime, columnName:string, taskID:int): void
~ UpdateTaskTitle(email:string, columnName:string, taskID:int, title:string): void
~ InProgressTasks(email:string): List<Task>

Task

+ ID: int
+ CreationTime: DateTime
+ Title: string
+ Description: string
+ DueDate: DateTime
+ EmailAssignee:string
+ Column: string
+ BoardID: int
~ taskDTO:TaskDTO

~ UpdateTaskDueDate(email:string, dueDate:DateTime): void
~ UpdateTaskTitle(email:string, title:string): void
~ UpdateTaskDescription(email:string, description:string): void
~ AssignTask(email:string, emailAssignee:string): void

Column

+ limit: int
+ columnName: string
~ Tasks: Dictionary<int,Task>
~ columnDTO: ColumnDTO

~ LimitColumn(limit: int): void
~ GetTask(taskId): Task
~ IsExists(taskID:int):bool
~ AssignTask(email:string, taskID:int, emailAssignee:string): void
~ AddTask(title:string, description:string, dueDate:DateTime, counterIDtask:int, boardID:int): void
~ AddTask(taskID:int, task:Task): void
~ Remove(taskID:int): void
~ UpdateTaskDescription(email:string, taskID:int, description:string): void
~ UpdateTaskDueDate(email:string, dueDate:DateTime, taskID:int): void
~ UpdateTaskTitle(email:string, taskID:int, title:string): void
~ InProgressTasks(): List<Task>

UserController

~ users: Dictionary<string, user>

+ CheckUser(email:string): bool
~ CreateUser(email:string, password:string): void
~ IsExists(email:string): bool
~ GetUser(email:string): User
~ IsLogIn(email:string):bool
~ LoadData(): void
~ DeleteData():void

BoardController

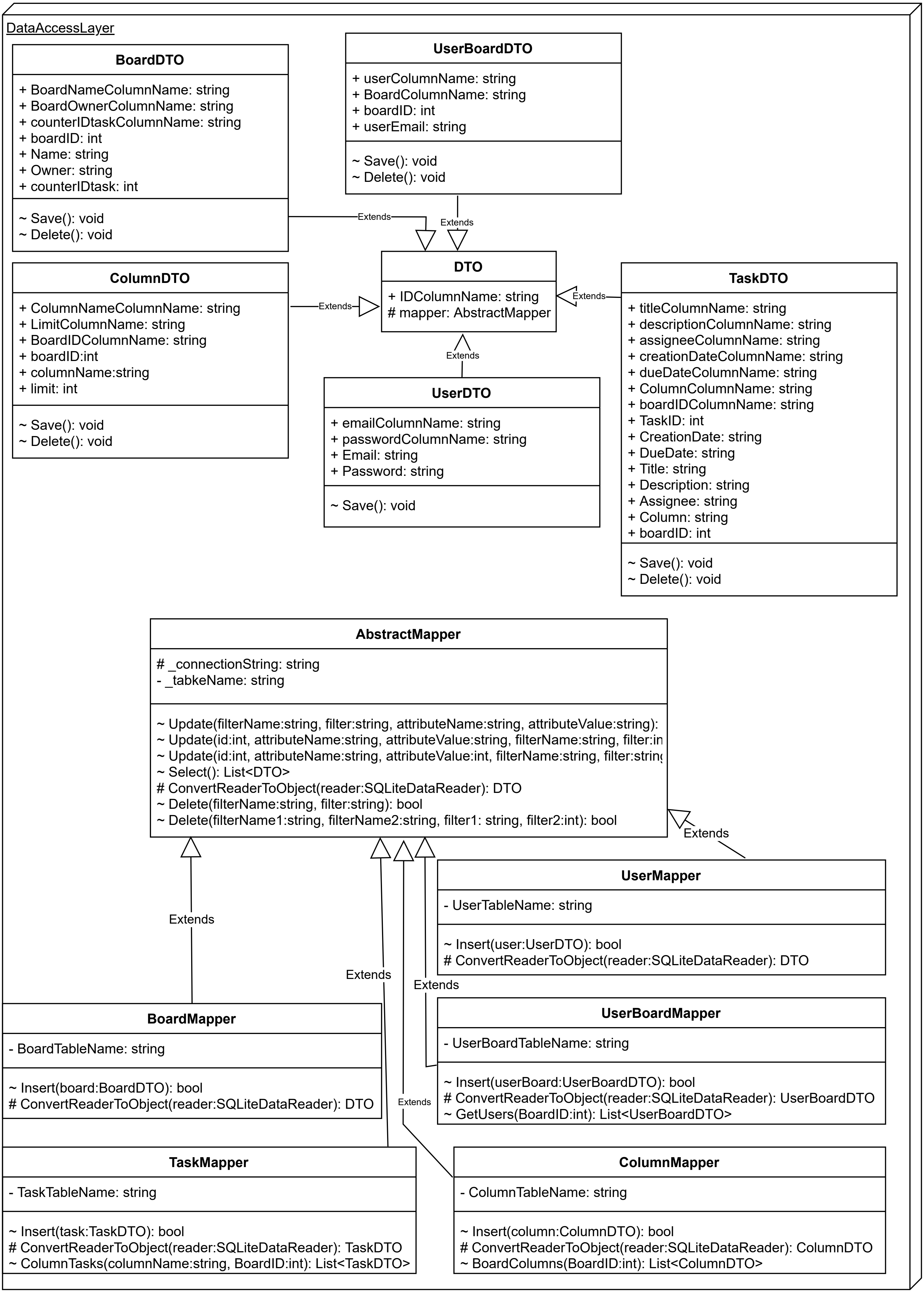
~ boards: Dictionary<int, Board>
- uc: UserController
- counterIDboards: int

~ AddBorad(user:User, nameBoard:string): void
~ GetBoardName(boardID:int): string
~ JoinBoard(email:string, boardID:int): void
~ LeaveBoard(email:string, boardID:int): void
~ RemoveBoard(user:User, Boardname:string): void
~ TransferOwnership(currentOwnerEmail:string, newOwnerEmail:string, boardID:int): void
~ CheckBoardUser(email:string, boardID:int): bool
~ GetBoard(email:string, nameBoard:string): Board
~ GetBoard(boardID:int): Board
~ IsBoardNameExists(email:string, nameBoard:string): bool
~ InProgressTasks(email:string): List<Task>
~ LoadData(): void
- MaxID(): int
- UBtoUser(userBoardDTOs:List<UserBoardDTO>): List<User>
~ DeleteData():void

User

~ BoardIDs:List<int>
+ UserEmail: string
+ Password: string
+ Status: bool
~ userDTO:UserDTO

~ IsValidPassword(password:string): bool
~ IsValidEmail(email:string): bool
~ Login(password:string): void
~ Logout(): void
~ ChangePassword(oldP:string, newP:string): void
~ GetUserBoards(): List<int>
~ AddBoard(boardID:int): void



BoardDTO

+ BoardNameColumnName: string
+ BoardOwnerColumnName: string
+ counterIDtaskColumnName: string
+ boardID: int
+ Name: string
+ Owner: string
+ counterIDtask: int

~ Save(): void
~ Delete(): void

UserBoardDTO

+ userColumnName: string
+ BoardColumnName: string
+ boardID: int
+ userEmail: string

~ Save(): void
~ Delete(): void

DTO

+ IDColumnName: string
# mapper: AbstractMapper

ColumnDTO

+ ColumnNameColumnName: string
+ LimitColumnName: string
+ BoardIDColumnName: string
+ boardID:int
+ columnName:string
+ limit: int

~ Save(): void
~ Delete(): void

TaskDTO

+ titleColumnName: string
+ descriptionColumnName: string
+ assigneeColumnName: string
+ creationDateColumnName: string
+ dueDateColumnName: string
+ ColumnColumnName: string
+ boardIDColumnName: string
+ TaskID: int
+ CreationDate: string
+ DueDate: string
+ Title: string
+ Description: string
+ Assignee: string
+ Column: string
+ boardID: int

~ Save(): void
~ Delete(): void

UserDTO

+ emailColumnName: string
+ passwordColumnName: string
+ Email: string
+ Password: string

~ Save(): void

AbstractMapper

_connectionString: string
- _tabkeName: string

~ Update(filterName:string, filter:string, attributeName:string, attributeValue:string):
~ Update(id:int, attributeName:string, attributeValue:string, filterName:string, filter:in
~ Update(id:int, attributeName:string, attributeValue:int, filterName:string, filter:strin
~ Select(): List<DTO>
# ConvertReaderToObject(reader:SQLiteDataReader): DTO
~ Delete(filterName:string, filter:string): bool
~ Delete(filterName1:string, filterName2:string, filter1: string, filter2:int): bool

UserMapper

- UserTableName: string

~ Insert(user:UserDTO): bool
# ConvertReaderToObject(reader:SQLiteDataReader): DTO

BoardMapper

- BoardTableName: string

~ Insert(board:BoardDTO): bool
# ConvertReaderToObject(reader:SQLiteDataReader): DTO

UserBoardMapper

- UserBoardTableName: string

~ Insert(userBoard:UserBoardDTO): bool
# ConvertReaderToObject(reader:SQLiteDataReader): UserBoardDTO
~ GetUsers(BoardID:int): List<UserBoardDTO>

TaskMapper

- TaskTableName: string

~ Insert(task:TaskDTO): bool
# ConvertReaderToObject(reader:SQLiteDataReader): TaskDTO
~ ColumnTasks(columnName:string, BoardID:int): List<TaskDTO>

ColumnMapper

- ColumnTableName: string

~ Insert(column:ColumnDTO): bool
# ConvertReaderToObject(reader:SQLiteDataReader): ColumnDTO
~ BoardColumns(BoardID:int): List<ColumnDTO>