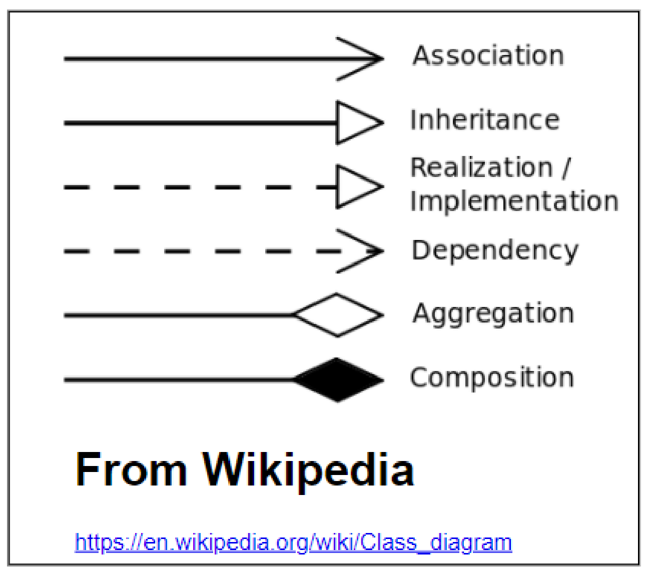
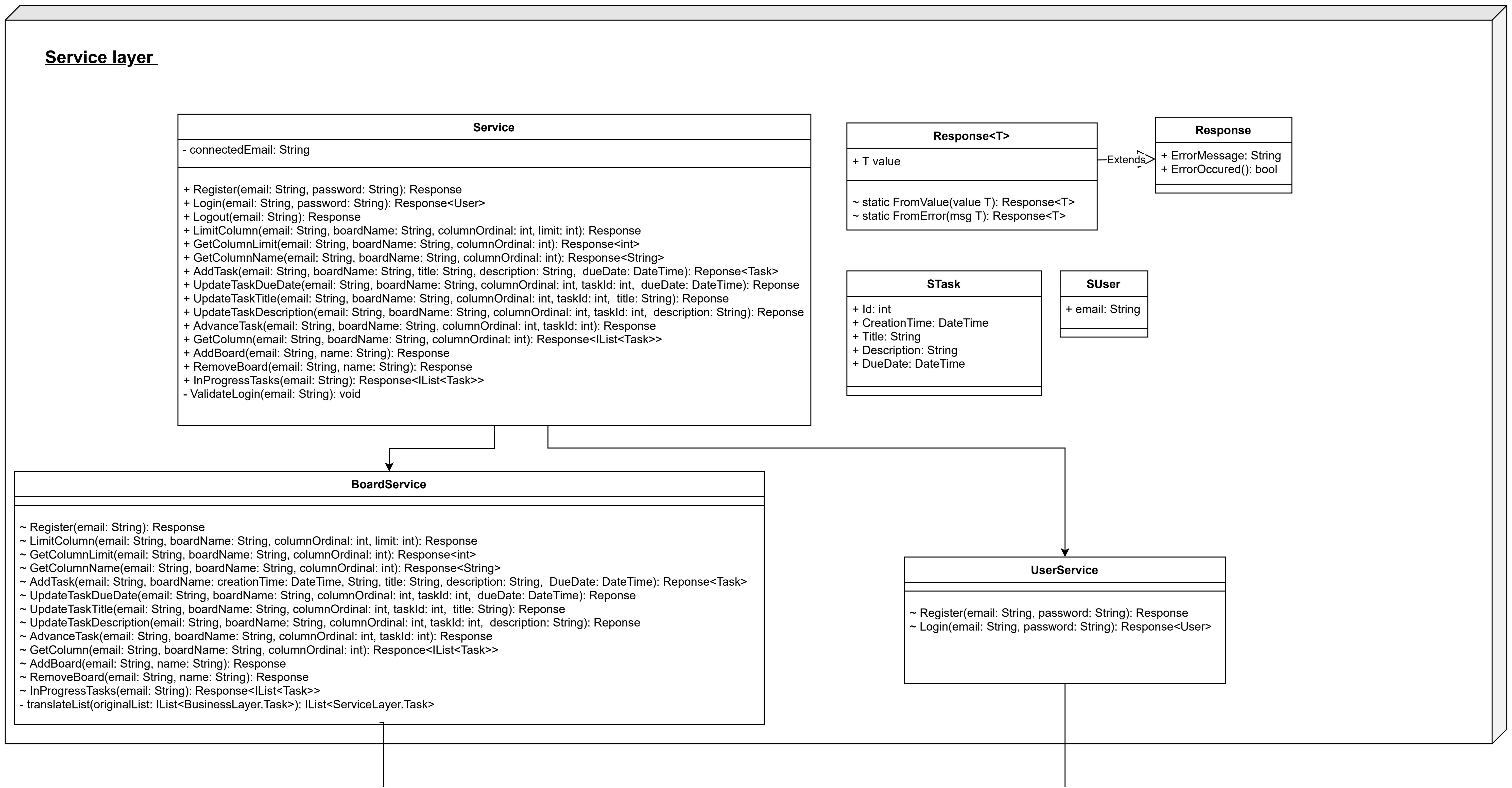
**visibility**
+ public
- private
~ internal


Association
Inheritance
Realization / Implementation
Dependency
Aggregation
Composition
From Wikipedia
https://en.wikipedia.org/wiki/Class_diagram



```
Service
- connectedEmail: String

+ Register(email: String, password: String): Response
+ Login(email: String, password: String): Response<User>
+ Logout(email: String): Response
+ LimitColumn(email: String, boardName: String, columnOrdinal: int, limit: int): Response
+ GetColumnLimit(email: String, boardName: String, columnOrdinal: int): Response<int>
+ GetColumnName(email: String, boardName: String, columnOrdinal: int): Response<String>
+ AddTask(email: String, boardName: String, title: String, description: String,  dueDate: DateTime): Reponse<Task>
+ UpdateTaskDueDate(email: String, boardName: String, columnOrdinal: int, taskId: int,  dueDate: DateTime): Reponse
+ UpdateTaskTitle(email: String, boardName: String, columnOrdinal: int, taskId: int,  title: String): Reponse
+ UpdateTaskDescription(email: String, boardName: String, columnOrdinal: int, taskId: int,  description: String): Reponse
+ AdvanceTask(email: String, boardName: String, columnOrdinal: int, taskId: int): Response
+ GetColumn(email: String, boardName: String, columnOrdinal: int): Response<IList<Task>>
+ AddBoard(email: String, name: String): Response
+ RemoveBoard(email: String, name: String): Response
+ InProgressTasks(email: String): Response<IList<Task>>
- ValidateLogin(email: String): void
```

```
Response<T>
+ T value

~ static FromValue(value T): Response<T>
~ static FromError(msg T): Response<T>
```

```
Response
+ ErrorMessage: String
+ ErrorOccured(): bool
```

```
STask
+ Id: int
+ CreationTime: DateTime
+ Title: String
+ Description: String
+ DueDate: DateTime
```

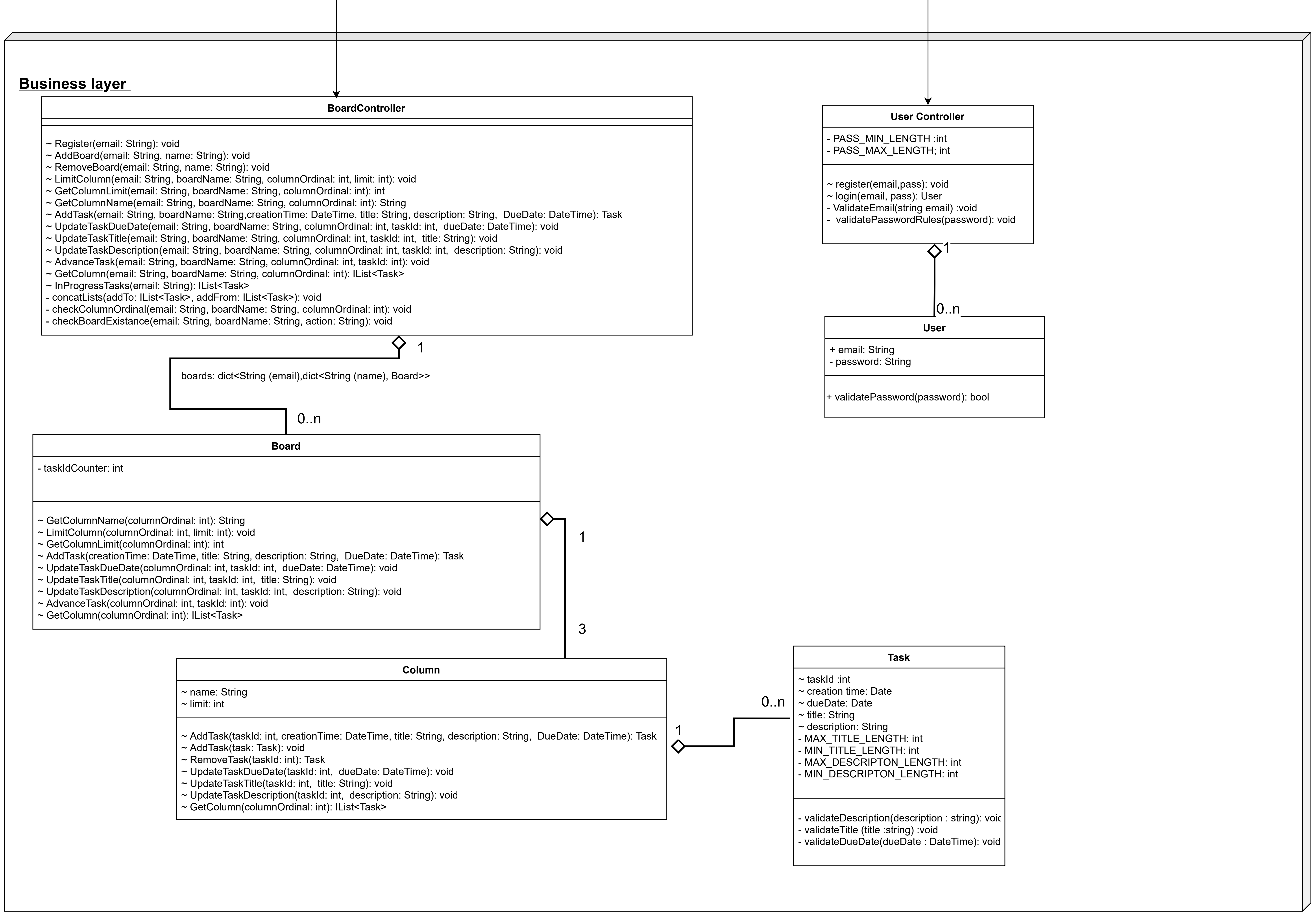
```
SUser
+ email: String
```

```
BoardService

~ Register(email: String): Response
~ LimitColumn(email: String, boardName: String, columnOrdinal: int, limit: int): Response
~ GetColumnLimit(email: String, boardName: String, columnOrdinal: int): Response<int>
~ GetColumnName(email: String, boardName: String, columnOrdinal: int): Response<String>
~ AddTask(email: String, boardName: creationTime: DateTime, String, title: String, description: String,  DueDate: DateTime): Reponse<Task>
~ UpdateTaskDueDate(email: String, boardName: String, columnOrdinal: int, taskId: int,  dueDate: DateTime): Reponse
~ UpdateTaskTitle(email: String, boardName: String, columnOrdinal: int, taskId: int,  title: String): Reponse
~ UpdateTaskDescription(email: String, boardName: String, columnOrdinal: int, taskId: int,  description: String): Reponse
~ AdvanceTask(email: String, boardName: String, columnOrdinal: int, taskId: int): Response
~ GetColumn(email: String, boardName: String, columnOrdinal: int): Responce<IList<Task>>
~ AddBoard(email: String, name: String): Response
~ RemoveBoard(email: String, name: String): Response
~ InProgressTasks(email: String): Response<IList<Task>>
- translateList(originalList: IList<BusinessLayer.Task>): IList<ServiceLayer.Task>
```

```
UserService

~ Register(email: String, password: String): Response
~ Login(email: String, password: String): Response<User>
```


Business layer
BoardController
~ Register(email: String): void
~ AddBoard(email: String, name: String): void
~ RemoveBoard(email: String, name: String): void
~ LimitColumn(email: String, boardName: String, columnOrdinal: int, limit: int): void
~ GetColumnLimit(email: String, boardName: String, columnOrdinal: int): int
~ GetColumnName(email: String, boardName: String, columnOrdinal: int): String
~ AddTask(email: String, boardName: String,creationTime: DateTime, title: String, description: String, DueDate: DateTime): Task
~ UpdateTaskDueDate(email: String, boardName: String, columnOrdinal: int, taskId: int, dueDate: DateTime): void
~ UpdateTaskTitle(email: String, boardName: String, columnOrdinal: int, taskId: int, title: String): void
~ UpdateTaskDescription(email: String, boardName: String, columnOrdinal: int, taskId: int, description: String): void
~ AdvanceTask(email: String, boardName: String, columnOrdinal: int, taskId: int): void
~ GetColumn(email: String, boardName: String, columnOrdinal: int): IList<Task>
~ InProgressTasks(email: String): IList<Task>
- concatLists(addTo: IList<Task>, addFrom: IList<Task>): void
- checkColumnOrdinal(email: String, boardName: String, columnOrdinal: int): void
- checkBoardExistance(email: String, boardName: String, action: String): void
1
boards: dict<String (email),dict<String (name), Board>>
0..n
Board
- taskIdCounter: int
~ GetColumnName(columnOrdinal: int): String
~ LimitColumn(columnOrdinal: int, limit: int): void
~ GetColumnLimit(columnOrdinal: int): int
~ AddTask(creationTime: DateTime, title: String, description: String, DueDate: DateTime): Task
~ UpdateTaskDueDate(columnOrdinal: int, taskId: int, dueDate: DateTime): void
~ UpdateTaskTitle(columnOrdinal: int, taskId: int, title: String): void
~ UpdateTaskDescription(columnOrdinal: int, taskId: int, description: String): void
~ AdvanceTask(columnOrdinal: int, taskId: int): void
~ GetColumn(columnOrdinal: int): IList<Task>
1
3
Column
~ name: String
~ limit: int
~ AddTask(taskId: int, creationTime: DateTime, title: String, description: String, DueDate: DateTime): Task
~ AddTask(task: Task): void
~ RemoveTask(taskId: int): Task
~ UpdateTaskDueDate(taskId: int, dueDate: DateTime): void
~ UpdateTaskTitle(taskId: int, title: String): void
~ UpdateTaskDescription(taskId: int, description: String): void
~ GetColumn(columnOrdinal: int): IList<Task>
1
0..n
Task
~ taskId :int
~ creation time: Date
~ dueDate: Date
~ title: String
~ description: String
- MAX_TITLE_LENGTH: int
- MIN_TITLE_LENGTH: int
- MAX_DESCRIPTON_LENGTH: int
- MIN_DESCRIPTON_LENGTH: int
- validateDescription(description : string): voic
- validateTitle (title :string) :void
- validateDueDate(dueDate : DateTime): void
User Controller
- PASS_MIN_LENGTH :int
- PASS_MAX_LENGTH; int
~ register(email,pass): void
~ login(email, pass): User
- ValidateEmail(string email) :void
- validatePasswordRules(password): void
1
0..n
User
+ email: String
- password: String
+ validatePassword(password): bool

## diagram change explanation

serviceLayer:

Adding UserService and Board service- we decided that we need to separate the functionality of service to improve the cohesion and to handle our 2 main entites of businessLayer (User,Board)

bussinesLayer:

BoardController
1. Adding BoardController- we realise that it is more logical and efficient to have an entitie that will handle all the existing boards and will be responsible for logical rules that depends on few boards. other then that by doing that we lower the coupling of Board and User, so now User want change the behavior of Board.

Board:
1.Add GetColumnName-since we added class Column, we realized utilizing the column field + all the tests running through all the chain of objects we decided to chain GetColumn as well
2.Add LimitColumn-need a function to execute the the demand of letting the user change the amount of Task that can be in a Column.
3.Add GetColumnLimit - since we added class Column, we realized utilizing the column field + all the tests running through all the chain of objects we decided to chain GetColumn as well
4.AddTask - this function replase createTask of the old Diagram, and the signature had change so we will be able to get to our wanted Task.
5.Change signature of Update Functions- we needed more information to get to specific Task. we need to know the taskID and the columnOrdinal.
6.AdvanceTask- this function replace moveTask of the old Diagram, the signature change because we needed more information to get to specific Task. we need to know the taskID and the columnOrdinal.

Column
1. Adding Column- Initially we thought of using 3 dictionaries for each column but found out we checked the columnOrdinal way too much, creating an array of columns and accessing it with columnOrdinal was easier and better

Task
1.Adding validators to Task- we want to restrict and manipulate if needed the option for title,description and dueDate.
2.Adding max and min static variable- if in the future we will want to change the rules of a specific filed it will be easy.
3.Adding taskId - important to have a key for each Task, so when needed we will be able to get to him easily and change only the wanted Task

UserController:
1.Add EmailExist - when register we want to check that no email as this already enrolled, and when login to verify that this email is enrolled, may use us in future changes and requirements
2.Add ValidateEmail - we need to make sure that the input of an email is indeed an email.
3.Adding max and min static variable- if in the future we will want to change the rules of a specific filed it will be easy.
4.Adding validatePasswordRules - when register or log in we want to check that the given password answer on the rules that we difine for a valid password.

User:

1.Remove logout function - we decided that it is much more efficient and lower the coupling to save the log in user in the Servece, so the log out will only be to change the field in Service and has no relation to User
2.Remove Board and Task functions - we decided to have a BoardController that will manipulate and no need to this function in User.
3.moving validatePasswordRules to UserController - we dicided that is more logical to have email rules and password rules in the same plase to cohesion so we moved it to UserController