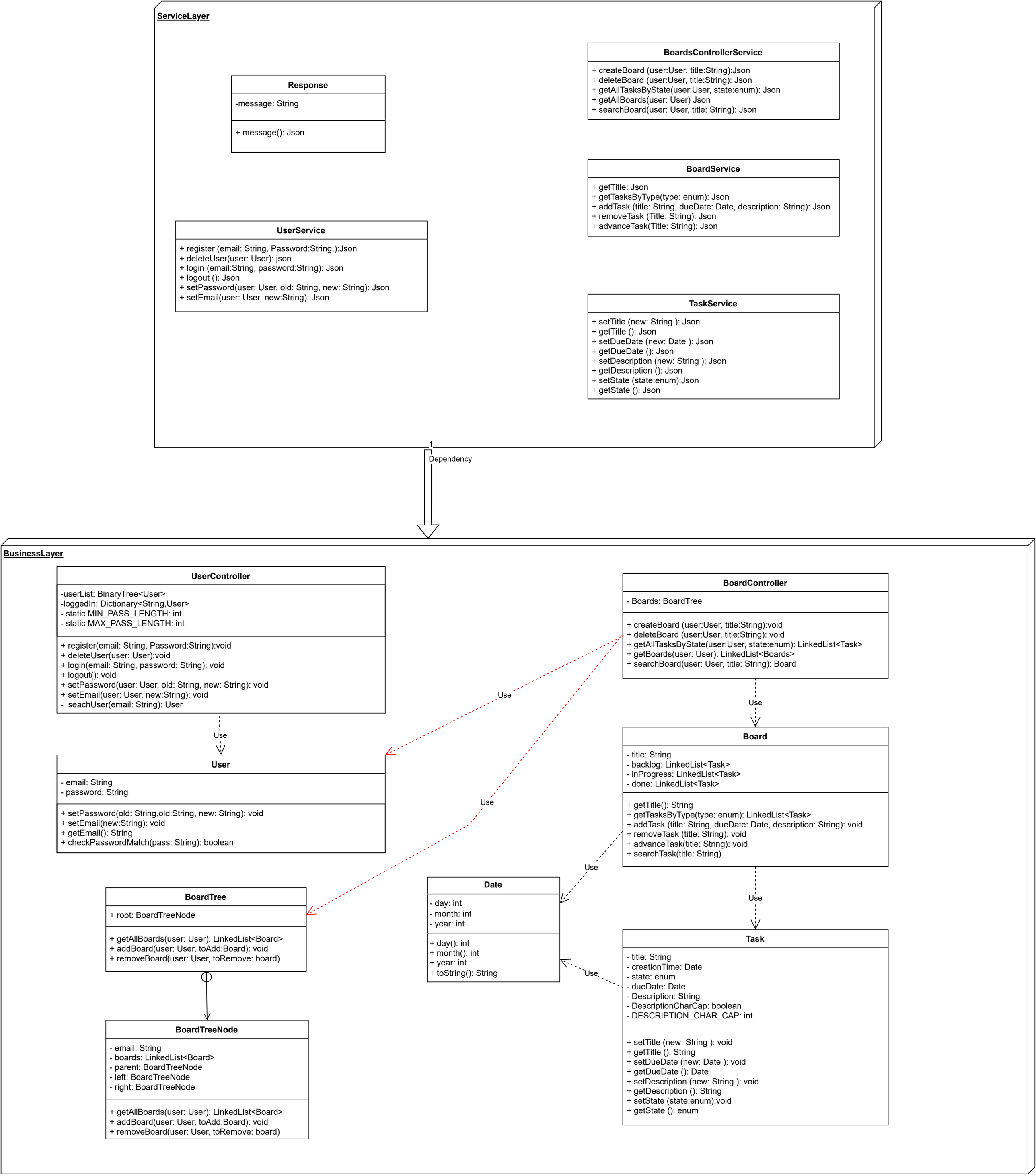
ServiceLayer
Response
-message: String
+ message(): Json
BoardsControllerService
+ createBoard (user:User, title:String):Json
+ deleteBoard (user:User, title:String): Json
+ getAllTasksByState(user:User, state:enum): Json
+ getAllBoards(user: User) Json
+ searchBoard(user: User, title: String): Json
BoardService
+ getTitle: Json
+ getTasksByType(type: enum): Json
+ addTask (title: String, dueDate: Date, description: String): Json
+ removeTask (Title: String): Json
+ advanceTask(Title: String): Json
UserService
+ register (email: String, Password:String,):Json
+ deleteUser(user: User): json
+ login (email:String, password:String): Json
+ logout (): Json
+ setPassword(user: User, old: String, new: String): Json
+ setEmail(user: User, new:String): Json
TaskService
+ setTitle (new: String): Json
+ getTitle (): Json
+ setDueDate (new: Date): Json
+ getDueDate (): Json
+ setDescription (new: String): Json
+ getDescription (): Json
+ setState (state:enum):Json
+ getState (): Json
1
Dependency
BusinessLayer
UserController
-userList: BinaryTree<User>
-loggedIn: Dictionary<String,User>
- static MIN_PASS_LENGTH: int
- static MAX_PASS_LENGTH: int
+ register(email: String, Password:String):void
+ deleteUser(user: User):void
+ login(email: String, password: String): void
+ logout(): void
+ setPassword(user: User, old: String, new: String): void
+ setEmail(user: User, new:String): void
- seachUser(email: String): User
BoardController
- Boards: BoardTree
+ createBoard (user:User, title:String):void
+ deleteBoard (user:User, title:String): void
+ getAllTasksByState(user:User, state:enum): LinkedList<Task>
+ getBoards(user: User): LinkedList<Boards>
+ searchBoard(user: User, title: String): Board
Use
Use
Use
Use
Use
Use
Use
User
- email: String
- password: String
+ setPassword(old: String,old:String, new: String): void
+ setEmail(new:String): void
+ getEmail(): String
+ checkPasswordMatch(pass: String): boolean
Board
- title: String
- backlog: LinkedList<Task>
- inProgress: LinkedList<Task>
- done: LinkedList<Task>
+ getTitle(): String
+ getTasksByType(type: enum): LinkedList<Task>
+ addTask (title: String, dueDate: Date, description: String): void
+ removeTask (title: String): void
+ advanceTask(title: String): void
+ searchTask(title: String)
Date
- day: int
- month: int
- year: int
+ day(): int
+ month(): int
+ year: int
+ toString(): String
BoardTree
+ root: BoardTreeNode
+ getAllBoards(user: User): LinkedList<Board>
+ addBoard(user: User, toAdd:Board): void
+ removeBoard(user: User, toRemove: board)
Task
- title: String
- creationTime: Date
- state: enum
- dueDate: Date
- Description: String
- DescriptionCharCap: boolean
- DESCRIPTION_CHAR_CAP: int
+ setTitle (new: String): void
+ getTitle (): String
+ setDueDate (new: Date): void
+ getDueDate (): Date
+ setDescription (new: String): void
+ getDescription (): String
+ setState (state:enum):void
+ getState (): enum
BoardTreeNode
- email: String
- boards: LinkedList<Board>
- parent: BoardTreeNode
- left: BoardTreeNode
- right: BoardTreeNode
+ getAllBoards(user: User): LinkedList<Board>
+ addBoard(user: User, toAdd:Board): void
+ removeBoard(user: User, toRemove: board)