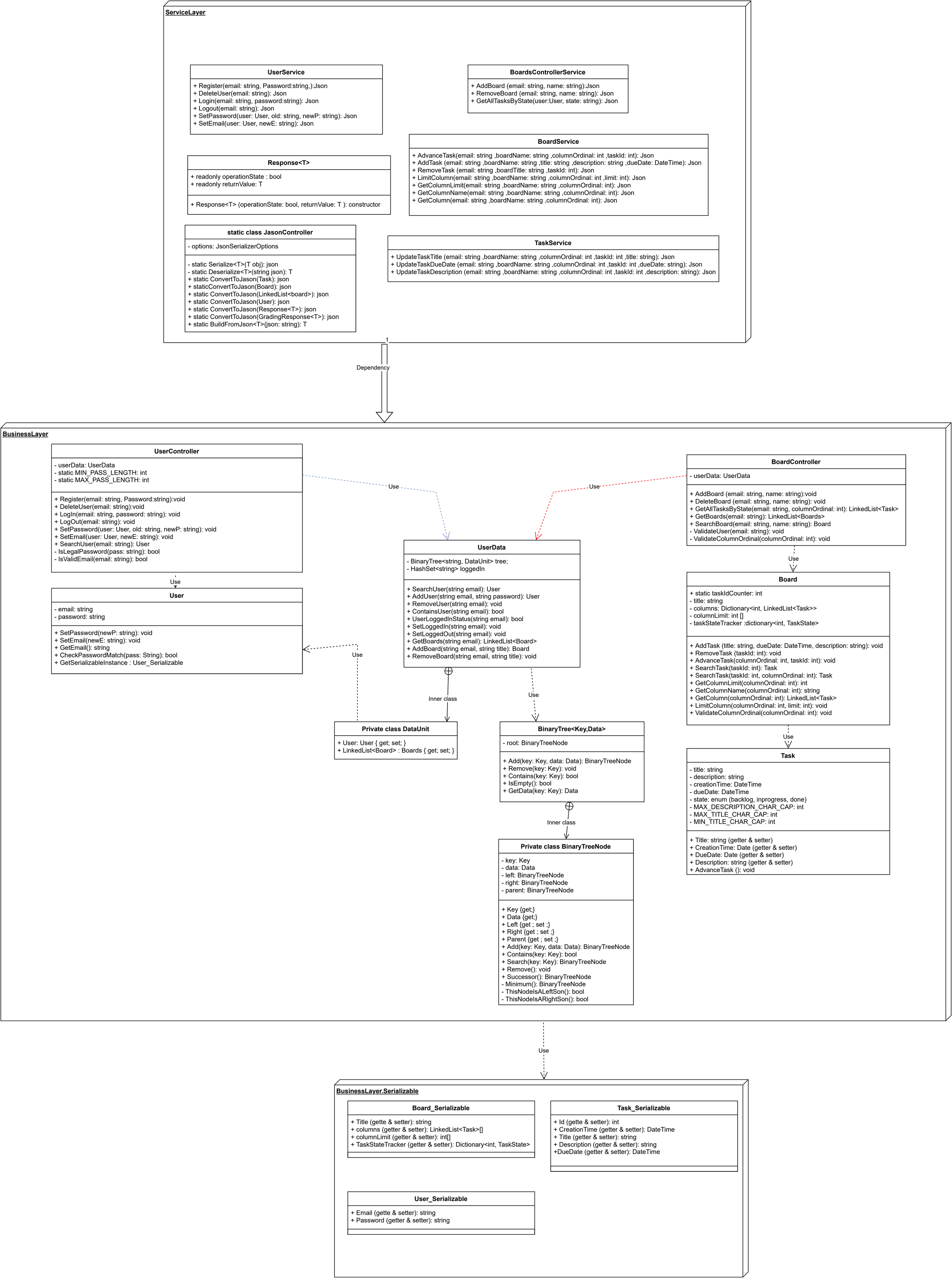
ServiceLayer
UserService
+ Register(email: string, Password:string,):Json
+ DeleteUser(email: string): Json
+ Login(email: string, password:string): Json
+ Logout(email: string): Json
+ SetPassword(user: User, old: string, newP: string): Json
+ SetEmail(user: User, newE: string): Json
BoardsControllerService
+ AddBoard (email: string, name: string):Json
+ RemoveBoard (email: string, name: string): Json
+ GetAllTasksByState(user:User, state: string): Json
BoardService
+ AdvanceTask(email: string ,boardName: string ,columnOrdinal: int ,taskId: int): Json
+ AddTask (email: string ,boardName: string ,title: string ,description: string ,dueDate: DateTime): Json
+ RemoveTask (email: string ,boardTitle: string ,taskId: int): Json
+ LimitColumn(email: string ,boardName: string ,columnOrdinal: int ,limit: int): Json
+ GetColumnLimit(email: string ,boardName: string ,columnOrdinal: int): Json
+ GetColumnName(email: string ,boardName: string ,columnOrdinal: int): Json
+ GetColumn(email: string ,boardName: string ,columnOrdinal: int): Json
Response<T>
+ readonly operationState : bool
+ readonly returnValue: T
+ Response<T> (operationState: bool, returnValue: T): constructor
static class JasonController
- options: JsonSerializerOptions
- static Serialize<T>(T obj): json
- static Deserialize<T>(string json): T
+ static ConvertToJason(Task): json
+ staticConvertToJason(Board): json
+ static ConvertToJason(LinkedList<board>): json
+ static ConvertToJason(User): json
+ static ConvertToJason(Response<T>): json
+ static ConvertToJason(GradingResponse<T>): json
+ static BuildFromJson<T>(json: string): T
TaskService
+ UpdateTaskTitle (email: string ,boardName: string ,columnOrdinal: int ,taskId: int ,title: string): Json
+ UpdateTaskDueDate (email: string ,boardName: string ,columnOrdinal: int ,taskId: int ,dueDate: string): Json
+ UpdateTaskDescription (email: string ,boardName: string ,columnOrdinal: int ,taskId: int ,description: string): Json
1
Dependency
BusinessLayer
UserController
- userData: UserData
- static MIN_PASS_LENGTH: int
- static MAX_PASS_LENGTH: int
+ Register(email: string, Password:string):void
+ DeleteUser(email: string):void
+ LogIn(email: string, password: string): void
+ LogOut(email: string): void
+ SetPassword(user: User, old: string, newP: string): void
+ SetEmail(user: User, newE: string): void
+ SearchUser(email: string): User
- IsLegalPassword(pass: string): bool
- IsValidEmail(email: string): bool
Use
Use
Use
BoardController
- userData: UserData
+ AddBoard (email: string, name: string):void
+ DeleteBoard (email: string, name: string): void
+ GetAllTasksByState(email: string, columnOrdinal: int): LinkedList<Task>
+ GetBoards(email: string): LinkedList<Boards>
+ SearchBoard(email: string, name: string): Board
- ValidateUser(email: string): void
- ValidateColumnOrdinal(columnOrdinal: int): void
Use
User
- email: string
- password: string
+ SetPassword(newP: string): void
+ SetEmail(newE: string): void
+ GetEmail(): string
+ CheckPasswordMatch(pass: String): bool
+ GetSerializableInstance : User_Serializable
Use
UserData
- BinaryTree<string, DataUnit> tree;
- HashSet<string> loggedIn
+ SearchUser(string email): User
+ AddUser(string email, string password): User
+ RemoveUser(string email): void
+ ContainsUser(string email): bool
+ UserLoggedInStatus(string email): bool
+ SetLoggedIn(string email): void
+ SetLoggedOut(string email): void
+ GetBoards(string email): LinkedList<Board>
+ AddBoard(string email, string title): Board
+ RemoveBoard(string email, string title): void
Board
+ static taskIdCounter: int
- title: string
- columns: Dictionary<int, LinkedList<Task>>
- columnLimit: int []
- taskStateTracker :dictionary<int, TaskState>
+ AddTask (title: string, dueDate: DateTime, description: string): void
+ RemoveTask (taskId: int): void
+ AdvanceTask(columnOrdinal: int, taskId: int): void
+ SearchTask(taskId: int): Task
+ SearchTask(taskId: int, columnOrdinal: int): Task
+ GetColumnLimit(columnOrdinal: int): int
+ GetColumnName(columnOrdinal: int): string
+ GetColumn(columnOrdinal: int): LinkedList<Task>
+ LimitColumn(columnOrdinal: int, limit: int): void
+ ValidateColumnOrdinal(columnOrdinal: int): void
Inner class
Use
Use
Private class DataUnit
+ User: User { get; set; }
+ LinkedList<Board> : Boards { get; set; }
BinaryTree<Key,Data>
- root: BinaryTreeNode
+ Add(key: Key, data: Data): BinaryTreeNode
+ Remove(key: Key): void
+ Contains(key: Key): bool
+ IsEmpty(): bool
+ GetData(key: Key): Data
Task
- title: string
- description: string
- creationTime: DateTime
- dueDate: DateTime
- state: enum (backlog, inprogress, done)
- MAX_DESCRIPTION_CHAR_CAP: int
- MAX_TITLE_CHAR_CAP: int
- MIN_TITLE_CHAR_CAP: int
+ Title: string (getter & setter)
+ CreationTime: Date (getter & setter)
+ DueDate: Date (getter & setter)
+ Description: string (getter & setter)
+ AdvanceTask (): void
Inner class
Private class BinaryTreeNode
- key: Key
- data: Data
- left: BinaryTreeNode
- right: BinaryTreeNode
- parent: BinaryTreeNode
+ Key {get;}
+ Data {get;}
+ Left {get ; set ;}
+ Right {get ; set ;}
+ Parent {get ; set ;}
+ Add(key: Key, data: Data): BinaryTreeNode
+ Contains(key: Key): bool
+ Search(key: Key): BinaryTreeNode
+ Remove(): void
+ Successor(): BinaryTreeNode
- Minimum(): BinaryTreeNode
- ThisNodeIsALeftSon(): bool
- ThisNodeIsARightSon(): bool
Use
BusinessLayer.Serializable
Board_Serializable
+ Title (gette & setter): string
+ columns (getter & setter): LinkedList<Task>[]
+ columnLimit (getter & setter): int[]
+ TaskStateTracker (getter & setter): Dictionary<int, TaskState>
Task_Serializable
+ Id (gette & setter): int
+ CreationTime (getter & setter): DateTime
+ Title (getter & setter): string
+ Description (getter & setter): string
+DueDate (getter & setter): DateTime
User_Serializable
+ Email (gette & setter): string
+ Password (getter & setter): string