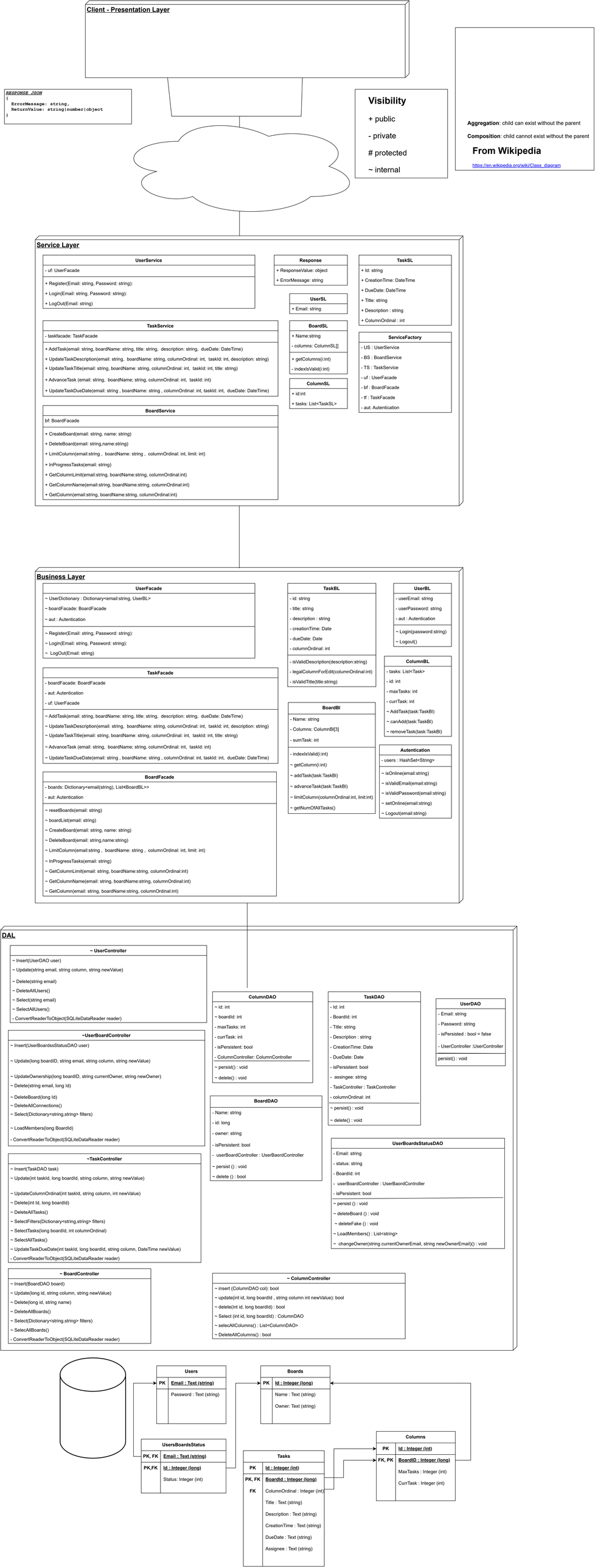
Client - Presentation Layer
RESPONSE JSON
{
ErrorMessage: string,
ReturnValue: string|number|object
}
Visibility
+ public
- private
protected
~ internal
Aggregation: child can exist without the parent
Composition: child cannot exist without the parent
From Wikipedia
https://en.wikipedia.org/wiki/Class_diagram
Service Layer
UserService
- uf: UserFacade
+ Register(Email: string, Password: string):
+ Login(Email: string, Password: string):
+ LogOut(Email: string)
Responne
+ ResponseValue: object
+ ErrorMessage: string
TaskSL
+ Id: string
+ CreationTime: DateTime
+ DueDate: DateTime
+ Title: string
+ Description : string
+ ColumnOrdinal : int
UserSL
+ Email: string
TaskService
- taskfacade: TaskFacade
+ AddTask(email: string, boardName: string, title: string, description: string, dueDate: DateTime)
+ UpdateTaskDescription(email: string, boardName: string, columnOrdinal: int, taskId: int, description: string)
+ UpdateTaskTitle(email: string, boardName: string, columnOrdinal: int, taskId: int, title: string)
+ AdvanceTask (email: string, boardName; string, columnOrdinal: int, taskId: int)
+ UpdateTaskDueDate(email: string , boardName: string , columnOrdinal: int, taskId: int, dueDate: DateTime)
BoardSL
+ Name:string
- columns: ColumnSL[]
+ getColumns(i:int)
- indexIsValid(i:int)
ServiceFactory
- US : UserService
- BS : BoardService
- TS : TaskService
- uf : UserFacade
- bf : BoardFacade
- tf : TaskFacade
- aut: Autentication
ColumnSL
+ id:int
+ tasks: List<TaskSL>
BoardService
bf: BoardFacade
+ CreateBoard(email: string, name: string)
+ DeleteBoard(email: string,name:string)
+ LimitColumn(email:string , boardName: string , columnOrdinal: int, limit: int)
+ InProgressTasks(email: string)
+ GetColumnLimit(email:string, boardName:string, columnOrdinal:int)
+ GetColumnName(email:string, boardName:string, columnOrdinal:int)
+ GetColumn(email:string, boardName:string, columnOrdinal:int)
Business Layer
UserFacade
~ UserDictionary : Dictionary<email:string, UserBL>
~ boardFacade: BoardFacade
~ aut : Autentication
~ Register(Email: string, Password: string):
~ Login(Email: string, Password: string):
~ LogOut(Email: string)
TaskBL
- id: string
- title: string
- description : string
- creationTime: Date
- dueDate: Date
- columnOrdinal: int
- isValidDescription(description:string)
- legalColumnForEdit(columnOrdinal:int)
- isValidTitle(title:string)
UserBL
- userEmail: string
- userPassword: string
- aut : Autentication
~ Login(password:string)
~ Logout()
TaskFacade
- boardFacade: BoardFacade
- aut: Autentication
- uf: UserFacade
~ AddTask(email: string, boardName: string, title: string, description: string, dueDate: DateTime)
~ UpdateTaskDescription(email: string, boardName: string, columnOrdinal: int, taskId: int, description: string)
~ UpdateTaskTitle(email: string, boardName: string, columnOrdinal: int, taskId: int, title: string)
~ AdvanceTask (email: string, boardName; string, columnOrdinal: int, taskId: int)
~ UpdateTaskDueDate(email: string , boardName: string , columnOrdinal: int, taskId: int, dueDate: DateTime)
BoardBl
- Name: string
- Columns: ColumnBl[3]
- sumTask: int
- indexIsValid(i:int)
~ getColumn(i:int)
~ addTask(task:TaskBl)
~ advanceTask(task:TaskBl)
~ limitColumn(columnOrdinal:int, linit:int)
~ getNumOfAllTasks()
ColumnBL
- tasks: List<Task>
- id: int
- maxTasks: int
- currTask: int
~ AddTask(task:TaskBl)
~ canAdd(task:TaskBl)
~ removeTask(task:TaskBl)
Autentication
- users : HashSet<String>
~ isOnline(email:string)
~ isValidEmail(email:string)
~ isValidPassword(email:string)
~ setOnline(email:string)
~ Logout(email:string)
BoardFacade
- boards: Dictionary<email(string), List<BoardBL>>
- aut: Autentication
~ resetBoards(email: string)
~ boardList(email: string)
~ CreateBoard(email: string, name: string)
~ DeleteBoard(email: string,name:string)
~ LimitColumn(email:string , boardName: string , columnOrdinal: int, limit: int)
~ InProgressTasks(email: string)
~ GetColumnLimit(email: string, boardName:string, columnOrdinal:int)
~ GetColumnName(email: string, boardName:string, columnOrdinal:int)
~ GetColumn(email: string, boardName:string, columnOrdinal:int)
DAL
~ UserController
~ Insert(UserDAO user)
~ Update(string email, string column, string newValue)
~ Delete(string email)
~ DeleteAllUsers()
~ Select(string email)
~ SelectAllUsers()
- ConvertReaderToObject(SQLiteDataReader reader)
ColumnDAO
~ id: int
~ boardId: int
- maxTasks: int
- currTask: int
- isPersistent: bool
- ColumnController: ColumnController
~ persist() : void
~ delete() : void
TaskDAO
- Id: int
- BoardId: int
- Title: string
- Description : string
- CreationTime: Date
- DueDate: Date
- isPersistent: bool
- assingee: string
- TaskController : TaskController
- columnOrdinal: int
~ persist() : void
~ delete() : void
UserDAO
- Email: string
- Password: string
- isPersisted : bool = false
- UserController :UserController
persist() : void
~UserBoardController
~ Insert(UserBoardssStatusDAO user)
~ Update(long boardID, string email, string column, string newValue)
~ UpdateOwnership(long boardID, string currentOwner, string newOwner)
~ Delete(string email, long Id)
~ DeleteBoard(long Id)
~ DeleteAllConnections()
~ Select(Dictionary<string,string> filters)
~ LoadMembers(long BoardId)
- ConvertReaderToObject(SQLiteDataReader reader)
BoardDAO
- Name: string
- id: long
- owner: string
- isPersistent: bool
- userBoardController : UserBaordController
~ persist () : void
~ delete () : bool
UserBoardsStatusDAO
- Email: string
- status: string
- BoardId: int
- userBoardController : UserBaordController
- isPersistent: bool
~ persist () : void
~ deleteBoard () : void
~ deleteFake () : void
~ LoadMembers() : List<string>
~ changeOwner(string currentOwnerEmail, string newOwnerEmail)() : void
~TaskController
~ Insert(TaskDAO task)
~ Update(int taskId, long boardId, string column, string newValue)
~ UpdateColumnOrdinal(int taskId, string column, int newValue)
~ Delete(int Id, long boardId)
~ DeleteAllTasks()
~ SelectFilters(Dictionary<string,string> filters)
~ SelectTasks(long boardId, int columnOrdinal)
~ SelectAllTasks()
~ UpdateTaskDueDate(int taskId, long boardId, string column, DateTime newValue)
- ConvertReaderToObject(SQLiteDataReader reader)
~ BoardController
~ Insert(BoardDAO board)
~ Update(long id, string column, string newValue)
~ Delete(long id, string name)
~ DeleteAllBoards()
~ Select(Dictionary<string,string> filters)
~ SelecAllBoards()
- ConvertReaderToObject(SQLiteDataReader reader)
~ ColumnController
~ insert (ColumnDAO col): bool
~ update(int id, long boardId , string column int newValue): bool
~ delete(int id, long boardId) : bool
~ Select (int id, long boardId) : ColumnDAO
~ selecAllColumns() : List<ColumnDAO>
~ DeleteAllColumns() : bool
Users
PK
Email : Text (string)
Password : Text (string)
Boards
PK
Id : Integer (long)
Name : Text (string)
Owner: Text (string)
UsersBoardsStatus
PK, FK
Email : Text (string)
PK,FK
Id : Integer (long)
Status: Integer (int)
Tasks
PK
Id : Integer (int)
PK, FK
BoardId : Integer (long)
FK
ColumnOrdinal : Integer (int)
Title : Text (string)
Description : Text (string)
CreationTime : Text (string)
DueDate : Text (string)
Assignee : Text (string)
Columns
PK
Id : Integer (int)
FK, PK
BoardID : Integer (long)
MaxTasks : Integer (int)
CurrTask : Integer (int)