**תיעוד פרוייקט - צד שרת :**

**מחלקת XXX:**

|  |  |
| --- | --- |
| תיאור |  |
| ירושת ממחלקה |  |
| מודולים: |  |
| מיקום |  |

**שדות:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | -/ |  | **:** |  |  |  |  |  |
| 2 | - |  | **:** |  |  |  |  |  |

**מתודות**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם מתודה** | **:** | **ערך החזרה** | **פונקציות נדרשות** | **שדות קשורות** | **תנאי פתיחה** | **תיאור** |
|  |  |  | **:** |  |  |  |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
|  |  |  | **:** |  |  |  |  |

**אופן שימוש :**

**Types**

הטיפוסים המותאמים אישית אלו טיפוסים המיועדים לשדה אחד. בשונה ממשק שמותאם לאובייקט של נתונים

|  |  |  |
| --- | --- | --- |
|  |  | listAllTableType:  מכיל רשימה לבחירה של כל השמות של הטבלאות בDB |
|  |  |  |

**Interfaces**

הממשקים מנתבים את המשתמש איזה ערכים יש להזין. הממשק הוא נתון שיש להזין ולא יבוא של ערכים מבסיס הנתונים

|  |  |  |
| --- | --- | --- |
|  |  | ILogin:  username :string;  password :string; |
|  |  | IRegister:  username :string;  password :string;  email :string;  firstName? :string  lastName? :string; |

**models:**

המודלים הם יבוא ישיר של בסיס הנתונים. מטרתם לקבע את המשתמש לערכים המתאימים ולמנוע מטעויות. המודל אינו מחייב את המשתמש אלא רק מהווה תבנית למידע.

כל מודל מכיל משתנה סטטי בשם "type" המהווה טיפוס של כל המפתחות של המודל. מטרתו למנוע טעויות בשם המפתחות בעת פיתוח.

|  |  |  |
| --- | --- | --- |
| itemModel:  id? :number;  name? :string;  description? :string;  freeze? :boolean;  active? :boolean;  price? :string;  sale? :string;  color? :string;  upgrade? :string;  rank? :number;  categoryItem? :string;  magicID? :number; | avatarModel:  id? :number;  name? :string;  createdDate? :string  freeze? :boolean;  exp? :number;  silver? :number;  gold? :number;  redPowder? :number;  diamond? :number;  userID? :number;  missionID? :number;  magicID? :number; | userModel:  id? :number;  username? :string;  email? :string;  firstName? :string;  lastName? :string;  birthday? :string; registerDate? :string;  freeze? : boolean;  token? :string; |
|  |  | avatarItemModel:  id? :number; active :boolean;  level : number;  avatarID : number;  itemID : number |
|  |  |  |
|  |  |  |
|  |  |  |

**JSON MODELS**

JSON מודל הוא אובייקט נתונים הממיר מידע בפורמט JSON המתקבל ממסד הנתונים ומחזיר אובייקט מסודר.

לכל מודל כזה יש בנאי המקבל JSON וממיר אותו לאובייקט תקין.

משום שמדובר בתבנית קבוע. כל השדות הם חובה.

|  |  |  |
| --- | --- | --- |
| Sale:  leven1 :Moneys;  leven2 :Moneys;  leven3 :Moneys;  leven4 :Moneys;  leven5 :Moneys;  leven6 :Moneys;  Moneys:  silver :number;  gold :number;  diamond :number;  redPowder :number; | Price:  exp :number;  silver :number;  gold :number;  diamond :number;  redPowder :number; | UpgradeItems:  leven1 :\_upgradeItems;  leven2 :\_upgradeItems;  leven3 :\_upgradeItems;  leven4 :\_upgradeItems;  leven5 :\_upgradeItems;  leven6 :\_upgradeItems;  \_upgradeItems:  hp :number;  energy :number;  refillEnergy :number;  exp :number;  silver :number;  gold :number;  diamond :number;  redPowder :number; |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**מחלקות בסיס:**

**מחלקת הבסיס Databae:**

|  |  |
| --- | --- |
| תיאור | מחלקה סטטית המבצעת שאילתות על בסיס הנתונים. |
| ירושת ממחלקה |  |
| מודולים: | msnodesqlv8 , synchronized-promise |
| מיקום | ./Database/connection |

**שדות (סטטיות)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | + | connection | **:** | string |  |  |  | מכיל את המחרוזת לחיבור לבסיס הנתונים מסוג MSSQL |
| 2 | + |  | **:** |  |  |  |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
| 1 | + | Query(query:string) | **:** | promise |  |  | מבצע שאילתא אסיכרונית |
| 2 | + | QuerySync(query:string) |  | ResultSQL |  |  | מבצע שאילתא סיכרונית. מחזיר אובייקט מיוחד בעל 2 מתודות : מתודה VALID ומתודה NOVALID |
|  |  |  |  |  |  |  |  |
| 3 | + | \_Select( obj: ISelect<any> ) |  | string |  | Database.protection | מחזיר שאילתא SELECT תקינה. מטרתה לקבל אובייקט עם נתונים אודות השאילתא ולהחזיר את השאילתא התקינה |
| 4 |  | Select<T> (obj:ISelect<T>) |  | promise |  | Database.Query  Database.\_select | ביצוע שאילתת SELECT אסיכרונית. המתודה מקבלת את המודל של בטבלה הרצויה. ובעזרת מחלקת העזר ISelect<> TS יודעת איזה שדות אפשר להזין |
| 5 |  | SelectSync<T>(obj:ISelect<T>) |  | ResultSQL |  | Database.QuerySync  Database.\_select | מבצע שאילתא SELECT סיכרונית . מחזיר אובייקט מיוחד בעל 2 מתודות : מתודה VALID ומתודה NOVALID |
| 6 |  | protection(value : any) |  | any |  |  | פונקצית אבטחה. מקבלת מחרוזת ומחזירה את המחרוזת מבלי תווים מיוחדים כלל. |

**אופן שימוש :**

מטרת מחלקה זו היא לשימוש בפונקציות סטטיות כלשהם.

השימוש העיקרי בה היא במחלקת הבסיס DB.

**מחלקת הבסיס DB:**

|  |  |
| --- | --- |
| תיאור | מחלקה מיועדת לירושה ותפקידה לבצע שאילתות עבור כל ישות באופן ספציפי |
| ירושת ממחלקה |  |
| מודולים: | Database |
| מיקום | ./Database/DB |

**שדות (סטטיות)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | /- | tableName |  | IListTableDB |  |  |  | שדה המכיל את שם הטבלה של הישות בDB. בעזרת נתון זה ניתן לבצע את השאילתא באופן דינמי ונוח יותר |
| 2 | /- | id |  | number | **null** |  |  | שדה של הID של הישות. |

**מתודות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
| 0 | + | constractor(obj : {**tableName**:IListTableDB} ) | **:** | void |  |  | מקבל בתוכו את שם הטבלה של הישות. את שם הטבלה עצמה יש להזין במחלקה יורשת תחת SUPER |
| 1 | /- | SelectSync<T>(obj:ISelect<T>) |  | ResultSQL |  | **Database**.SelectSync | מבצע שאילתא SELECT סיכרונית . מחזיר אובייקט מיוחד בעל 2 מתודות : מתודה VALID ומתודה NOVALID |
| 2 | /- | Query(query:string) |  | promiss |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**אופן שימוש :**

מחלקה זו היא מחלקה גנרית. המקבלת את המודל של הישות עליו היורשת אותו.  
בעזרת התכונה הגנרית הזו הTS ידע את סוג היישות עליה מדובר ותתנהג בהתאם.

מחלקת DB מכילה בתוכה את השדה id כך שאין צורך לממש שדה זה במחלקת הישות היורשת.

כל ישות היורשת את המחלקה DB צריכה לממש :

1. הזנת המודל כך : class Entity extent DB<typeof Model.type>
2. מימוש הבנאי באמצעות : super({tableName: "tableNameDB"})

**מחלקות קטגוריה**

מחלקות קטגוריה אלו מחלקות המצינות רשימה. מחלקות הקטגוריה אינן משתמשות במבנה נתונים מורכבים אלא רק במבנים פרמיטיבים בהתאם לDB.

**מחלקת Magic:**

|  |  |
| --- | --- |
| תיאור | מחלקה המכילה את כל הסוגי כשפויות |
| ירושת ממחלקה | DB |
| מודולים: |  |
| מיקום | Models/Categories/Magic.ts |

**שדות:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | - | name | **:** | string |  | **-** |  |  |
| 2 | - | description | **:** | string |  | **-** |  |  |
| 3 | - | freeze |  | boolean |  | **-** |  |  |
| 4 | - | isExist |  | boolean | **fasle** | **-** |  |  |

**מתודות**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם מתודה** | **:** | **ערך החזרה** | **פונקציות נדרשות** | **שדות קשורות** | **תנאי פתיחה** | **תיאור** |
|  |  | constructor(obj? :MagicModel) | **:** | void |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | GetAllItems(sync :boolean) |  | Item[] | Databas.Query |  |  |  |
|  |  | GetAllCard(sync :boolean) |  | Card[] | Databas.Query |  |  |  |
|  |  | GetAllAvatars(sync :boolean) |  | Avatar[] | Databas.Query |  |  |  |
|  |  | GetAllMissions(sync :boolean) |  | Mission[] | Databas.Query |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | GetAllItemsLite(sync :boolean) |  | Item[] | Databas.Query |  |  |  |
|  |  | GetAllCardLite(sync :boolean) |  | Card[] | Databas.Query |  |  |  |
|  |  | GetAllAvatarsLite(sync :boolean) |  | Avatar[] | Databas.Query |  |  |  |
|  |  | GetAllMissionsLite(sync :boolean) |  | Mission[] | Databas.Query |  |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
|  |  | GetMagicById(magicID :number) | **:** | Magic |  | Database.SelectSync |  |
|  |  | GetMagicByName(magicName : string) |  | Magic |  |  |  |
|  |  | GetListMagics() |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  | GetAllByMagic(magic :Magic) |  |  |  |  |  |

**אופן שימוש :**

**מחלקת map:**

|  |  |
| --- | --- |
| תיאור | מחלקה המתארת את כל המפות במשחק. |
| ירושת ממחלקה | DB |
| מודולים: |  |
| מיקום |  |

**שדות:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | - | name | **:** | string |  | **-** |  |  |
| 2 | - | description | **:** | string |  | **-** |  |  |
|  |  | freeze |  | boolean |  | **-** |  |  |
|  |  | isExist |  | boolean | **false** | **-** |  |  |

**מתודות**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם מתודה** | **:** | **ערך החזרה** | **פונקציות נדרשות** | **שדות קשורות** | **תנאי פתיחה** | **תיאור** |
|  | + | constructor(obj? :MapsModel) | **:** |  |  |  |  |  |
|  | + | GetAllLabyrinth() |  | Labyrinth[] |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
|  | + | GetMapById(mapID :number) | **:** | Map |  |  |  |
|  |  | GetMapByName(mapName :string) |  | Map |  |  |  |
|  |  | GetAllMaps() |  | Map[] |  |  |  |
|  |  |  |  |  |  |  |  |

**אופן שימוש :**

**מחלקות כלים :**

מחלקות אשר משמשות את השחקן ככלי.

**מחלקת card:**

|  |  |
| --- | --- |
| תיאור | מחלקת המתאר את כלפי המשחק |
| ירושת ממחלקה | DB |
| מודולים: |  |
| מיקום | Models/Entities/Card/Card.ts |

**שדות:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | -/ | id | **:** | number |  | **-** |  |  |
| 2 | - | name | **:** | string |  | **-** |  |  |
|  | - | description |  | string |  | **-** |  |  |
|  |  | type |  | string |  | **-** |  |  |
|  |  | minAvatarLevel |  |  |  |  |  |  |
|  |  | price |  | Price |  | **-** |  |  |
|  |  | move |  | Move |  | **-** |  |  |
|  |  | attack |  | Attack |  | **-** |  |  |
|  |  | delay |  | number |  | **-** |  |  |
|  |  | upgrade |  | UpgradeCards |  | **-** |  |  |
|  |  | freeze |  | boolean |  | **-** |  |  |
|  |  | isExist |  | boolean | **false** | **-** |  |  |
|  |  | magic |  | Magic |  | **-** |  |  |
|  |  | avatar |  | Avatar | **null** | **-** |  | מכיל את בעל הקלף. |

**מתודות**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם מתודה** | **:** | **ערך החזרה** | **פונקציות נדרשות** | **שדות קשורות** | **תנאי פתיחה** | **תיאור** |
|  |  | constructor(obj? :CardsModel) | **:** | void |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | RankUp(num = 1) |  | this |  |  |  | מעלה את הקלף ברמתו |
|  |  |  |  |  |  |  |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
|  |  | GetCardById() | **:** | Card |  |  |  |
|  |  | GetCardByName() |  | Card |  |  |  |
|  |  | GetCardsByAvatar |  | Promiss |  |  |  |
|  |  | GetCardsByAvatarSync |  | Card[] |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  | GetCardsByMinAvatarLeven |  | Promiss |  |  |  |
|  |  | GetCardsByMinAvatarLevenSync |  | Card[] |  |  |  |
|  |  | GetCardsByMagic |  | Promiss |  |  |  |
|  |  | GetCardsByMagicSync |  | Card[] |  |  |  |
|  |  | GetCardsByType |  | Promiss |  |  |  |
|  |  | GetCardsByTypeSync |  | Card[] |  |  |  |

**אופן שימוש :**

**מחלקת user:**

|  |  |
| --- | --- |
| תיאור | ישות המשתמש. |
| ירושת ממחלקה | מחלקת DB |
| מודולים |  |
| מיקום | ./Entities/user.ts |
| מודל ישות | userModel |

**שדות:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| **0** | -/ | id | **:** | number | null | null | **-** |  |  |
| 1 | **-** | username | **:** | string | null | null | **-** |  |  |
| 2 | - | email | **:** | string | null | null | **-** | **-** |  |
| 3 | - | firstName | **:** | string | null | null | **-** | **-** |  |
| 4 | - | lastName | **:** | string | null | null | **-** | **-** |  |
| 5 | - | birthday | **:** | string | null | null | **-** | **-** |  |
| 6 | - | registerDate | **:** | string | null | null | **-** |  |  |
| 7 | - | banned | **:** | boolean | null | null | **-** |  |  |
| 8 | - | freeze | **:** | boolean | null | null | **-** |  |  |
| 9 | - | token | **:** | string | null | null | **-** |  |  |
| 10 | - | message | **:** | any | null | **-** |  |  |
|  |  |  |  |  |  |  |  |  |
| 11 | - | isExist | **:** | boolean | false | **-** |  |  |
| 12 | - | isLogin | **:** | boolean | false | **-** |  |  |
| 13 | - | isSelectedAvatar | **:** | boolean | false | **-** |  |  |
|  |  |  |  |  |  |  |  |  |
| 14 | - | **avatars** | **:** | Avatar[] | null | **-** |  |  |
| 15 | - | **activeAvatar** | **:** | Avatar | null | null | **-** |  |  |

**מתודות**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם מתודה** |  | **ערך החזרה** | **פונקציות נדרשות** | **שדות קשורות** | **תנאי פתיחה** | **תיאור** |
| 0 | + | constructor() | **:** | void | super({tableName:"users"}) |  |  |  |
|  |  |  | **:** |  |  |  |  |  |
| 1 | + | login(obj:ILogin) | **:** | void | LoginValidation,  **Database**.SelectSync, fillFields  createToken,  **Avatar**.GetAvatarsByUserId, | isLogin, isLogin , messase | if isLogin == true --> return |  |
| 2 | + | logout() | **:** | void | removeToken | id, isLogin | if isLogin == false --> return |  |
| 3 | + | register(obj:IRegister) | **:** | void | RegisterValidation, | isExist , isLogin , messase | if isLogin == true --> return |  |
|  |  |  | **:** |  |  |  |  |  |
| 4 | - | createToken() | **:** | void | randomString, this(DB).UpdateSync | token |  |  |
| 5 | - | removeToken() | **:** | void | this(DB).UpdateSync | token |  |  |
|  | - |  | **:** |  |  |  |  |  |
| 6 | - | UpdateActiveAvatar(AvatarID:number) | **:** | void |  | activeAvatar, |  |  |
| 7 | - | fillFields(data:UserModel) | **:** | void |  | \* |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **+** | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
|  |  | **:** |  |  |  |  |
|  |  | **:** |  |  |  |  |

**אופן שימוש :**

בעת חיבור ראשוני בין הלקוח לשרת נוצר אובייקט USER. אובייקט זה יכיל נתונים בסיסים אשר מהווים נתוני אורח.

בעת חיבור למשחק תופעל המתודה LOGIN שתייבא את הנתונים מבסיס הנתונים ובכך יש בזכרון אובייקט המכיל את כל המידע של השחקן.

בעת התנתקות השרת ימחק את נתוני המשתמש מהזכרון.

האובייקט הנ"ל הוא האובייקט היחידי שאנו נשתמש. כל שאר הישויות והמחלקות מוכלות בתוך הUSER.

**מחלקת Avatar:**

|  |  |
| --- | --- |
| תיאור |  |
| ירושת ממחלקה | DB |
|  |  |
|  |  |

**שדות:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **שם שדה** | **:** | **טיפוס** | **ברירת מחדל** | **GET** | **SET** | **תיאור** |
| 1 | -/ |  | **:** |  |  |  |  |  |
| 2 | - |  | **:** |  |  |  |  |  |

**מתודות**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם מתודה** | **:** | **ערך החזרה** | **פונקציות נדרשות** | **שדות קשורות** | **תנאי פתיחה** | **תיאור** |
|  |  |  | **:** |  |  |  |  |  |

**פונקציות סטטיות:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  | **שם פונקציה** | **:** | **ערך החזרה** |  | **פונקציות נדרשות** | **תיאור** |
|  |  |  | **:** |  |  |  |  |

**אופן שימוש :**