Estructura API

Vamos a usar separación por capas para dividir el código en Business (capa de negocio, donde pondremos las funciones del negocio de la aplicación), Infraestructura (capa de datos, donde pondremos las funciones que contactarán con la base de datos) y Entity (capa de presentación, donde definiremos las clases que utilizaremos y sus atributos).

Estructura Base de Datos

User:
- id: int PK auto_increment
- username: String
- password: String
- email: String
- isAdmin: boolean

```
CREATE TABLE `User` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `isAdmin` tinyint(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8;
```

Profemon:
- id: int PK auto_increment
- name: String
- initialLevel: int

```
CREATE TABLE `Profemon` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `initialLevel` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
```

Capturado:
- id: int PK auto_increment
- idUser: int FK User.id
- idProfemon: int FK Profemon.id
- idLocation: int FK Location.id
- level: int
- date: datetime
- isSuccessful: boolean

```
CREATE TABLE `Capturado` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `idUser` int(11) DEFAULT NULL,
  `idProfemon` int(11) DEFAULT NULL,
  `idLocation` int(11) DEFAULT NULL,
```

```
  `level` int(11) DEFAULT NULL,
  `date` datetime DEFAULT NULL,
  `isSuccessful` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `idProfemon FK` (`idProfemon`),
  KEY `idUser FK` (`idUser`),
  KEY `idLocation` (`idLocation`),
  CONSTRAINT `idLocation` FOREIGN KEY (`idLocation`) REFERENCES `Location` (`id`),
  CONSTRAINT `idProfemon FK` FOREIGN KEY (`idProfemon`) REFERENCES `Profemon` (`id`),
  CONSTRAINT `idUser FK` FOREIGN KEY (`idUser`) REFERENCES `User` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Location:
- id: int PK auto_increment
- latitude: double
- longitude: double
- floor: int

```
CREATE TABLE `Location` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `latitude` double DEFAULT NULL,
  `longitude` double DEFAULT NULL,
  `floor` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

Router:
- id: int PK auto_increment
- BSSID: String
- floor: int

```
CREATE TABLE `Router` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `BSSID` varchar(11) DEFAULT NULL,
  `floor` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

Objetos Java

User:
- id: int PK auto_increment
- username: String
- password: String
- email: String
- isAdmin: boolean
- profemons: List<Profemon>

Profemon:
- id: int
- name: String
- initialLevel: int

Capturado:
- id: int
- idUser: int
- user: User
- idProfemon: int
- profemon: Profemon
- idLocation: int
- location: Location
- date: Date
- level: int

Router:
- id: int
- BSSID: String
- floor: int

Location:
- id: int
- latitude: double
- longitude: double
- floor: int

API para Parte Web (pokeetac/)

Utilizar Session Storage Javascript.

Login:
POST/user/login
{
"username":"samuel",
"password":"1234"
}
Produces:
{
"isSuccessful":"true/false"
}

POST/user/register
{
"username":"samuel",
“password”:”1234",
“email”:”samuel@gmail.com”,
“isAdmin”:”true/false”
}
Produces:
{
“isSuccessful”:”true/false"
}

Profedex:
GET/profemon/all/{filterBy: .*}
Produces:
{
{
"id":"1",
"name":"Tonimon",
"initialLevel":"1"
}
{
"id":"2",
"name":"Juanizard",
"initialLevel":"2"
}
}

My Profile:

GET/user/level/{id del usuario}
Produces:

```
{
        "level":"6"
}
```

GET/user/profemons/{id del usuario}
Produces:

```
{
        {
                "id":"1",
                "name":"Tonimon",
                "level":"1"
        }
        {
                "id":"2",
                "name":"Juanizard",
                "level":"2"
        }
}
```

GET/user/capturados/successfulByDay/{id del usuario}
Produces:

```
{
        "capturadosOneDayAgo":"3",
        "capturadosTwoDaysAgo":"2",
        "capturadosThreeDaysAgo":"3",
        "capturadosFourDaysAgo":"5",
        "capturadosFiveDaysAgo":"1",
        "capturadosSixDaysAgo":"0",
        "capturadosSevenDaysAgo":"3"
}
```

GET/user/capturados/successfulPercentage/{id del usuario}
Produces:

```
{
        "successfulPercentage":"60"
}
```

GET/user/capturados/{id del usuario}
Produces:

```
{
        “profemonId”:”1”,
        “name":"tonimon",
        “locationId”:”1”,
        "latitude":"43.1",
        “longitude”:"41.1",
        “floor”:”0”
}
```

Settings (for Admin):
POST/profemon
{
"name":"Tonimon",
“level":"1"
}

GET/profemon/all (Igual que la definida en Profedex)

DELETE/profemon/{id del profemon a borrar}

GET/user/all
{
"username":"samuel",
“password”:”1234",
“email”:”samuel@gmail.com”,
“isAdmin”:”true/false”
}

API para Parte Android

Utilizar objeto Application para pasarle cosas al server.

Login y Register:
Igual que para la web.

MapActivity:
GET/user/level/{id del usuario}
Produces:

```
{
        "level":"6"
}
```

GET/profemon/location/all
Produces:

```
{
        {
                “profemonId”:”1”,
                “name":"tonimon",
                “locationId”:”1”,
                "latitude":"43.1",
                “longitude”:"41.1",
                “floor”:”0”
        }
        {
                “profemonId”:”1”,
                “name":"tonimon",
                “locationId”:”2”,
                "latitude":"42.1",
                “longitude”:"40.1",
                “floor”:”1”
        }
}
```

POST/user/location/floor

```
{
        {
                “BSSID”:”RouterString",
                “signalLevel”:”34.5”
        }
        {
                “BSSID”:”Router2String",
                “signalLevel”:”31.5”
        }
}
```

Produces:

```
{
        "floor":"2"
}
```

Al acabar los mini juegos:

POST/capturado

```
{
        "idUser":"14",
        "idProfemon":"4",
        "idLocation":"7",
        "isSuccessful":"true/false"
}
```