Instituto Tecnológico de Costa Rica

Área Ingeniería en Computadores

Desarrollo de Aplicaciones para Dispositivos Móviles

II Proyecto:

Going On

Profesor:

Andrei Fuentes Leiva

Estudiante:

Gia Yao Chen Liang

200940129

II Semestre, 2014

Mobile Service Going On

API Name

- addevent:

Script:

```
exports.post = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.addEvent ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?";

    mssql.query(sql, [request.body.name, request.body.description, request.body.startDate,

    request.body.endDate, request.body.startTime, request.body.endTime, request.body.eventPrice,

    request.body.idTypeEvent, request.body.idTypePrivacyEvent, request.body.idTypeStateEvent,

    request.body.latitude, request.body.longitude,  request.body.Username], {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};

exports.get = function(request, response) {

    response.send(statusCodes.OK, { message : 'Hello World!' });

};
```

API Name

- addUser:

Script:

```
exports.post = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.addUser ?, ?, ?, ?, ?, ?, ?, ?, ?";

    mssql.query(sql, [request.body.name, request.body.password, request.body.userName, request.body.idClassUser, request.body.idTypeUser, request.body.idTypeUserLocal, request.body.description, request.body.latitude, request.body.longitude], {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};


exports.get = function(request, response) {

    response.send(statusCodes.OK, { message : 'Hello World!' });

};
```

API Name:

- classusers:

Script:

```
exports.post = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.spSetClassUsers ?";

    mssql.query(sql, [request.body.name], {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};


exports.get = function(request, response) {

    //response.send(statusCodes.OK, { message : 'Hello World!' });

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.spGetAllClassUsers";

    mssql.query(sql, null, {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};
```

API Name

- geteventinfo:

Script

```
exports.post = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.getEventInfo ?";

    mssql.query(sql, [request.body.idEvent], {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};


exports.get = function(request, response) {

    response.send(statusCodes.OK, { message : 'Hello World!' });

};
```

Api Name

- gettypeusers:

Script

```
exports.post = function(request, response) {

    response.send(statusCodes.OK, { message : 'Hello World!' });

};


exports.get = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "select * from TypeUsers";

    mssql.query(sql, {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};
```

API Name

- loginusers:

Script

```
exports.post = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.spIsUser ?, ?, ?";

    mssql.query(sql, [request.body.Username, request.body.Password, request.body.idClassUser], {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};


exports.get = function(request, response) {

    response.send(statusCodes.OK, { message : 'Hello World!' });

};
```

API Name

- registeruser:

Script

```
exports.post = function(request, response) {
    var params = [request.body.Username, request.body.Password, request.body.name, request.body.firstName,
    request.body.lastName, request.body.description, request.body.idTypeUser, request.body.idTypeUserLocal,
    request.body.idClassUser];

    var mssql = request.service.mssql;
    var sql = "exec GoingOn.spSetUser ?, ?, ?, ?, ?, ?, ?, ?, ?";
    mssql.query(sql, params, {
        success: function(results) {
            if(results.length == 1)
                response.send(200, results[0]);
        }
    })
};


exports.get = function(request, response) {
    var mssql = request.service.mssql;
    var sql = "exec GoingOn.spGetAllTypeUsers";
    mssql.query(sql, null, {
        success: function(results) {
            if(results.length == 1)
                response.send(200, results[0]);
        }
    })
};
```

API Name

- typeusers:

Script

```
exports.post = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.spSetTypeUsers ?";

    mssql.query(sql, [request.body.name], {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};


exports.get = function(request, response) {

    var mssql = request.service.mssql;

    var sql = "exec GoingOn.spGetAllTypeUsers";

    mssql.query(sql, null, {

        success: function(results) {

            if(results.length == 1)

                response.send(200, results[0]);

        }

    })

};
```

API Name

- typeuserslocal:

Script

```
exports.post = function(request, response) {
    var mssql = request.service.mssql;
    var sql = "exec GoingOn.spSetTypeUsersLocal ?";
    mssql.query(sql, [request.body.name], {
        success: function(results) {
            if(results.length == 1)
                response.send(200, results[0]);
        }
    })
};


exports.get = function(request, response) {
    var mssql = request.service.mssql;
    var sql = "exec GoingOn.spGetAllTypeUsersLocal";
    mssql.query(sql, null, {
        success: function(results) {
            if(results.length == 1)
                response.send(200, results[0]);
        }
    })
};
```

Stores Procedures

Store Procedure Name

- addEvent

Parameters

- @name: nvarchar 50
- @description: nvarchar 200
- @startDate: date
- @endDate: date
- @startTime: time
- @endTime: time
- @eventPrice: nvarchar
- @idTypeEvent: int
- @idTypePrivacyEvent: int
- @idTypeStateEvent: int
- @latitude: decimal (18,5)
- @longitude: decimal (18,5)
- @Username: nvarchar 50

Script

```
DECLARE @TransactionName varchar(20) = 'addEvent';

BEGIN TRANSACTION @TransactionName



begin try



Declare @idAddress int

EXEC GoingOn.spSetAddress  ' ',@latitude, @longitude, 1, @idAddress OUTPUT



Declare @idUsers int

EXEC GoingOn.spGetIdUser @Username, @idUsers OUTPUT
```

```
Declare @id int

SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.Event


INSERT GoingOn.Event(name,description,startDate, endDate,startTime, endTime, eventPrice,idTypeEvent,idTypePrivacyEvent,idTypeStateEvent,idUsers,idAddress) VALUES (@name, @description,@startDate, @endDate, @startTime, @endTime,  @eventPrice, @idTypeEvent, @idTypePrivacyEvent, @idTypeStateEvent, @idUsers, @idAddress)

Select @id as idEvent


COMMIT  TRANSACTION @TransactionName

END try

begin catch

SELECT @id = 0

Select @id as idEvent

ROLLBACK TRANSACTION @TransactionName

END catch
```

Store Procedure Name

- addUser

Parameters

- @name: nvarchar 50
- @password: nvarchar 50
- @userName: varchar 50
- @idClassUser: int
- @idTypeUser: int
- @idTypeUserLocal: int
- @description: nvarchar 200
- @latitude: decimal (18,5)
- @longitude: decimal (18.5)

Script

```
DECLARE @TransactionName varchar(20) = 'addUser';

BEGIN TRANSACTION @TransactionName


begin try


Declare @count int

EXEC GoingOn.spCountUser @userName, @idClassUser, @count OUTPUT


IF @count < 1

BEGIN


Declare @idAddress int

EXEC GoingOn.spSetAddress  ' ',@latitude, @longitude, 1, @idAddress OUTPUT
```

```
Declare @idPeople int

EXEC GoingOn.spSetPerson @name, ' ', ' ', @userName, @idAddress , @idPeople OUTPUT


Declare @id int

SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.Users


INSERT    GoingOn.Users(Username,Password,idTypeUser,idTypeUserLocal,idPerson,idClassUser,active,description)    VALUES
(@userName, @password, @idTypeUser, @idTypeUserLocal, @idPeople, @idClassUser, 1, @description)


Select @id as id


COMMIT  TRANSACTION @TransactionName

END


ELSE

BEGIN

SELECT @id = 0

Select @id as id

COMMIT  TRANSACTION @TransactionName

END


END try


begin catch

SELECT @id = 0

Select @id as id

ROLLBACK TRANSACTION @TransactionName

END catch
```

Store Procedure Name

- getEventInfo

Parameters

- @idEvent: int

Script

```
DECLARE @TransactionName varchar(20) = 'getEventInfo';

BEGIN TRANSACTION @TransactionName


begin try

select Event.name, Event.description, Event.startDate, Event.endDate, Event.startTime, Event.endTime, Event.eventPrice, TypeEvent.name as TypeEvent, TypePrivacyEvent.name as TypePrivacyEvent,

TypeStateEvent.name as TypeStateEvent from GoingOn.Event

INNER JOIN GoingOn.TypeEvent On Event.idTypeEvent=TypeEvent.id

INNER JOIN GoingOn.TypePrivacyEvent On Event.idTypePrivacyEvent=TypePrivacyEvent.id

INNER JOIN GoingOn.TypeStateEvent On Event.idTypeStateEvent=TypeStateEvent.id WHERE Event.id=@idEvent

COMMIT  TRANSACTION @TransactionName

end try


begin catch

ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spCountUser

Parameters

- @Username: nvarchar 50
- @idClassUser: int
- @count: int (output)

Script

```
DECLARE @TransactionName varchar(20) = 'spCountUser';

BEGIN TRANSACTION @TransactionName


begin try


SELECT @count = count(*) FROM GoingOn.Users WHERE @Username = Username AND @idClassUser = idClassUser

COMMIT  TRANSACTION @TransactionName


end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spGetAllClassUsers

Parameters

Script

```
DECLARE @TransactionName varchar(20) = 'spGetAllClassUsers';

BEGIN TRANSACTION @TransactionName

begin try


SELECT * FROM GoingOn.ClassUsers

COMMIT TRANSACTION @TransactionName

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spGetAllTypeUsers

Parameters

Script

```
DECLARE @TransactionName varchar(20) = 'spGetAllTypeUsers';

BEGIN TRANSACTION @TransactionName

begin try


SELECT * FROM GoingOn.TypeUsers

COMMIT TRANSACTION @TransactionName

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spGetAllTypeUsersLocal

Parameters

Script

```
DECLARE @TransactionName varchar(20) = 'spGetAllTypeUsersLocal';

BEGIN TRANSACTION @TransactionName

begin try


SELECT * FROM GoingOn.TypeUsersLocal

COMMIT TRANSACTION @TransactionName

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spGetIdUser

Parameters

- @Username: nvarchar 50
- @id: int (output)

Script

```
DECLARE @TransactionName varchar(20) = 'GetIdUser';

BEGIN TRANSACTION @TransactionName


begin try


SELECT @id=id  FROM GoingOn.Users WHERE @Username = Username

COMMIT  TRANSACTION @TransactionName


end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spGetTypeUsersCount

Parameters

- @Username: nvarchar 50
- @id: int

Script

```
begin

  select count(*) as count from GoingOn.TypeUsers

end
```

Store Procedure Name

- spIsUser

Parameters

- @Username: nvarchar 50
- @Password: nvarchar 50
- @idClassUser: int

Script

```
DECLARE @TransactionName varchar(20) = 'spIsUser ';

BEGIN TRANSACTION @TransactionName



begin try

DECLARE @count int, @login int

SELECT @count = count(*)  FROM GoingOn.Users WHERE @Username = Username AND @Password = Password AND active =
1 AND @idClassUser = idClassUser



/* si es menor que 0, NO se da el login*/

IF @count < 1

SELECT @login = 0

ELSE

SELECT @login = 1

SELECT @login as login

COMMIT  TRANSACTION @TransactionName

end try



begin catch

ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spSetAddress

Parameters

- @description: nvarchar 50
- @latitude: decimal (18,5)
- @longitude: decimal (18,5)
- @idCity: int
- @id: int (output)
-

Script

```
DECLARE @TransactionName varchar(20) = 'spSetAddress';

BEGIN TRANSACTION @TransactionName

begin try


SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.Addresses

INSERT GoingOn.Addresses(description,latitude,longitude,idCity) VALUES (@description, @latitude, @longitude, @idCity)

COMMIT TRANSACTION @TransactionName

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spSetCity

Parameters

- @name: nvarchar 50
- @idState: int
- @id: int (output)

Script

```
DECLARE @TransactionName varchar(20) = 'spSetCity';

BEGIN TRANSACTION @TransactionName

begin try


SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.Cities

INSERT GoingOn.Cities(name,idState) VALUES (@name,@idState)

COMMIT TRANSACTION

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spSetClassUsers

Parameters

- @name: nvarchar 50

Script

```
DECLARE @TransactionName varchar(20) = 'spSetClassUsers';

BEGIN TRANSACTION @TransactionName

begin try


DECLARE @id int

SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.ClassUsers

INSERT GoingOn.ClassUsers(name) VALUES (@name)

SELECT @id as ID

COMMIT TRANSACTION @TransactionName

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spSetState

Parameters

- @name: nvarchar 50
- @idCountry: int
- @id: int (output)

Script

```
DECLARE @TransactionName varchar(20) = 'spSetState  ';

BEGIN TRANSACTION @TransactionName

begin try


SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.States

INSERT GoingOn.States(name, idCountry) VALUES (@name, @idCountry)

COMMIT TRANSACTION

end try

begin catch


ROLLBACK TRANSACTION @TransactionName

end catch
```

Store Procedure Name

- spSetPerson

Parameters

- @name: nvarchar 50
- @firstName: nvarchar 50
- @lastName: nvarchar 50
- @email: nvarchar 50
- @idAddress: int
- @id: int (output)
-

Script

```
DECLARE @TransactionName varchar(20) = 'spSetPerson';

BEGIN TRANSACTION @TransactionName


begin try

SELECT @id =  coalesce(MAX(id), 0)+1 FROM GoingOn.People

INSERT  GoingOn.People(name,firstName,lastName,idAddress,email)  VALUES  (@name,  @firstName,  @lastName,  @idAddress,
@email)

COMMIT TRANSACTION @TransactionName

end try


begin catch

ROLLBACK TRANSACTION @TransactionName

end catch
```