Desafío Técnico – Santander Tecnología

Challenge Meetups

Documentación General

Consigna

En Santander Tecnología queremos armar las mejores meetups y para eso planeamos hacer una App que nos ayude a lograr que no falte lo fundamental…¡Birras!

¿Cuál es el problema?

Tenemos un proveedor que nos vende cajas de 6 unidades de birras. El problema es que: si hace entre 20 y 24 grados, se toma una birra por persona; si hace menos de 20 grados, se toma 0.75; y si hace mucho calor (más de 24 grados), se toman 2 birras más por persona. Y siempre preferimos que sobre y no que falte.

Cada historia de usuario una tiene su parte front y back, las dos primeras son obligatorias y cuantas más hagas ¡Mejor!

- Como admin quiero saber cuántas cajas de birras tengo que comprar para poder aprovisionar la meetup.
- Como admin y usuario quiero conocer la temperatura del día de la meetup para saber si va a hacer calor o no.
- Como usuario y como admin quiero poder recibir notificaciones para estar al tanto de las meetups.
- Como admin quiero armar una meetup para poder invitar otras personas.
- Como usuario quiero inscribirme en una meetup para poder asistir.
- Como usuario quiero hacer check-in en una meetup para poder avisar que estuve ahí.

Objetivo

Presentar la solución propuesta para el problema “Meetups – Birras”, utilizando recursos como diagramas para poder brindar distintos puntos de vista de la API.

Introducción

Siguiendo la regla de presentar una solución con una implementación se realizó una API:

Meetup

URL: https://meetups-santander.herokuapp.com/

Especificación

Tecnologías

Las principales tecnologías que se utilizaron para construir la solución son:

- Java 11
- Gradle
- Springboot
- JPA
- PostgreSQL

API Clima utilizada:

https://www.weatherbit.io/

Servicio utilizado:

https://www.weatherbit.io/api/weather-forecast-16-day

Descripción de los servicios

La API se encuentra hosteada en heroku, que es un servicio de hosting que brinda opciones para hostear APIS de manera gratuita. Para más información puede ingresar a www.heroku.com.

Meetup

La API expone los siguientes servicios

Controlador Usuario

- */usuarios/crear_usuario_admin*

URL: https://meetups-santander.herokuapp.com/meetups/api/v1/usuarios/crear_usuario_admin

Este servicio acepta llamadas POST siendo la estructura del body:

```
{
    "nombre": "UsuarioAdmin",
    "apellido":"UsuarioAdmin",
    "usuario": "Uadmin",
    "password": "admin123",
    "email": "Admin@santander.com"
}
```

El response es con un código 200(OK) y la estructura del response es:

```
{
    "idUsuario": 1,
    "nombre": "UsuarioAdmin",
    "apellido": "UsuarioAdmin",
    "usuario": "Uadmin",
    "password": "admin123",
    "email": "Admin@santander.com",
    "idPrivilegio": 1,
    "privilegio": null,
    "meetups": null,
    "meetupUsuarios": null
}
```

Indicándonos que el id del usuario es “1” en este caso y tiene el privilegio “1” (Administrador).

En caso de no poder crear el usuario se retorna un response de código 404 (NOT FOUND)

- */usuarios/crear_usuario*

URL: https://meetups-santander.herokuapp.com/meetups/api/v1/usuarios/crear_usuario

Este servicio acepta llamadas POST siendo la estructura del body:

```
{
    "nombre": "UsuarioComun",
    "apellido":"UsuarioComun",
    "usuario": "Ucomun",
    "password": "comun123",
    "email": "comunn@santander.com"
}
```

El response es con un código 200(OK) y la estructura del response es:

```
{
    "idUsuario": 2,
    "nombre": "UsuarioComun",
    "apellido": "UsuarioComun",
    "usuario": "Ucomun",
    "password": "comun123",
    "email": "comunn@santander.com",
    "idPrivilegio": 2,
    "privilegio": null,
    "meetups": null,
    "meetupUsuarios": null
}
```

Indicándonos que el id del usuario es “2” en este caso y tiene el privilegio “2” (Usuario común).

En caso de no poder crear el usuario se retorna un response de código 404 (NOT FOUND)

Controlador Meetup

- /meetups/crear_meetup

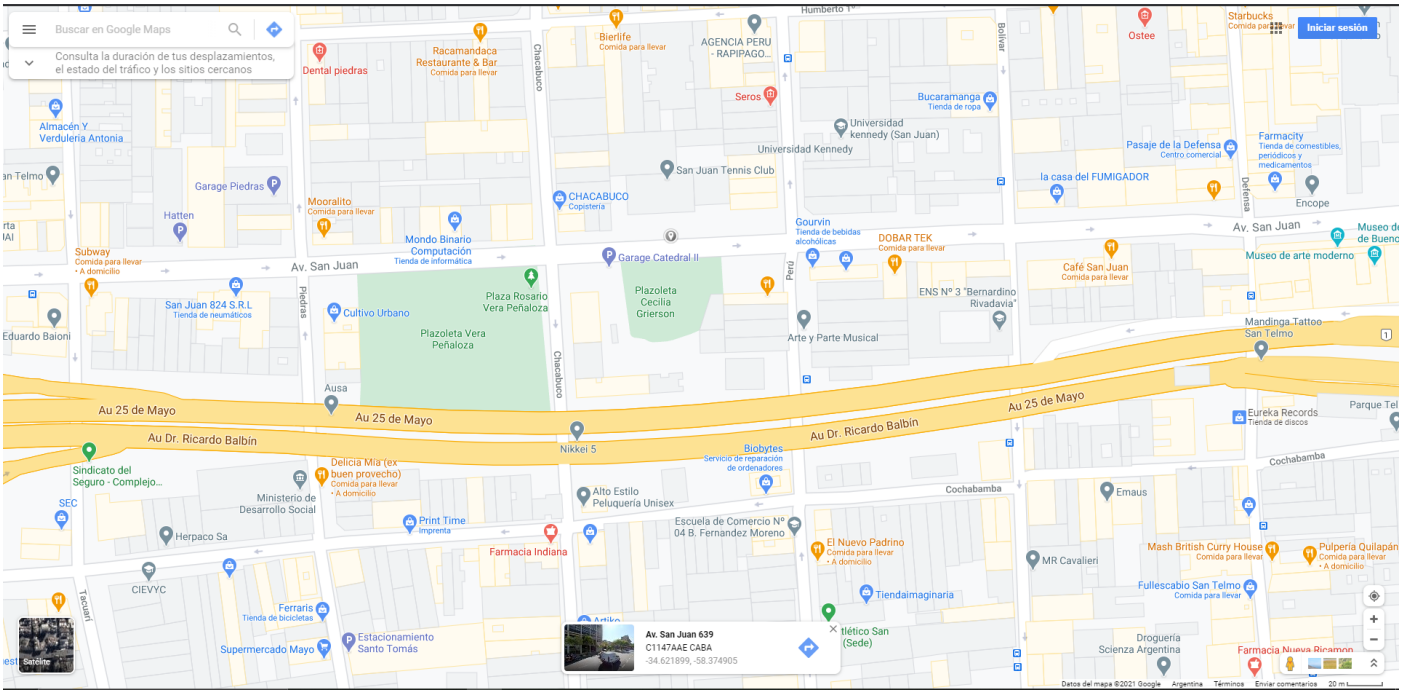
URL: https://meetups-santander.herokuapp.com/meetups/api/v1/meetups/crear_meetup

Este servicio acepta llamadas POST siendo la estructura del body:

```
{
    "idUsuarioCreador": 1,
    "fecha": "2021-02-22T16:10:00",
    "latitud": -34.673160,
    "longitud": -58.541422
}
```

Body:

- idUsuarioCreador, debe ser un usuario de tipo administrador existente.
- La fecha valida que sea mayor a “ahora”, y no sobrepase los 16 días.
- Se debe enviar la latitud y longitud de donde se realizará la meetup. Esto se puede obtener utilizando Google maps:



Marcando cualquier punto, nos dará las coordenadas de latitud y longitud:

El response es con un código 200(OK) y la estructura del response es:


```
{
    "idMeetup": 1,
    "idUsuarioCreador": 1,
    "fecha": "2021-02-22T16:10:00",
    "temperatura": 27.9,
    "cantidadDeCerveza": 1,
    "latitud": -34.67316,
    "longitud": -58.541422,
    "usuario": null,
    "meetupUsuarios": null
}
```


Retornando la temperatura de la meetup y la cantidad de packs de cerveza que se deberían comprar hasta el momento. Ya que el único "invitado/anfitrión" es el usuario creador.

- /meetups/consultar_temperatura/{idMeetup}

URL: https://meetups-santander.herokuapp.com/meetups/api/v1/meetups/consultar_temperatura/{idMeetup}

Este servicio acepta llamadas GET. (No debería llevar un body pero por falta de tiempo olvidé borrarlo)


```
{
    "idUsuario":1
}
```


El response es con un código 200(OK) y la estructura del response es:


```
27.9
```


En caso de no poder obtener la temperatura se retorna un response de código 404 (NOT FOUND)

- /meetups/consultar_cantidad_cervezas/{idMeetup}

URL: https://meetups-santander.herokuapp.com/meetups/api/v1/meetups/consultar_cantidad_cervezas/{idMeetup}

Este servicio acepta llamadas GET y debe llevar un usuario de tipo administrador en el body.


```
{
    "idUsuario":1
}
```

El response es con un código 200(OK) y la estructura del response es:

```
1
```

En caso de no poder obtener la cantidad de cerveza se retorna un response de código 404 (NOT FOUND)

Controlador MeetupUsuario

- /meetups_usuarios/inscribirme_meetup/{idMeetup}

URL: https://meetups-santander.herokuapp.com/meetups/api/v1/meetups_usuarios/inscribirme_meetup/{idMeetup}

Este servicio acepta llamadas POST siendo la estructura del body:

```
{
   "idUsuario": 2,
   "usuario": "Ucomun"
}
```

El response es con un código 200(OK) y la estructura del response es un String:

```
Inscripción a la meetup realizada correctamente.
```

En caso de no poder realizar la inscripción se retorna un response de código 404 (NOT FOUND)

Se válida que el usuario que se envía NO sea administrador y que exista.

Que la meetup exista y que no haya concluido ya.

Este servicio realiza nuevamente el cálculo de las cervezas luego de alojar un nuevo invitado en la meetup.

- /meetups_usuarios/checkin_meetup/{idMeetup}

URL: https://meetups-santander.herokuapp.com/meetups/api/v1/meetups_usuarios/checkin_meetup/{idMeetup}

Este servicio acepta llamadas POST siendo la estructura del body:

```
{
   "idUsuario": 2,
   "usuario": "Ucomun"
}
```

El response es con un código 200(OK) y la estructura del response es:

```
Check-in realizado correctamente.
```

En caso de no poder realizar la inscripción se retorna un response de código 404 (NOT FOUND)

- Se válida que el usuario que se envía NO sea administrador y que exista.
- Que la meetup exista y que haya concluido ya.

Diagrama de arquitectura

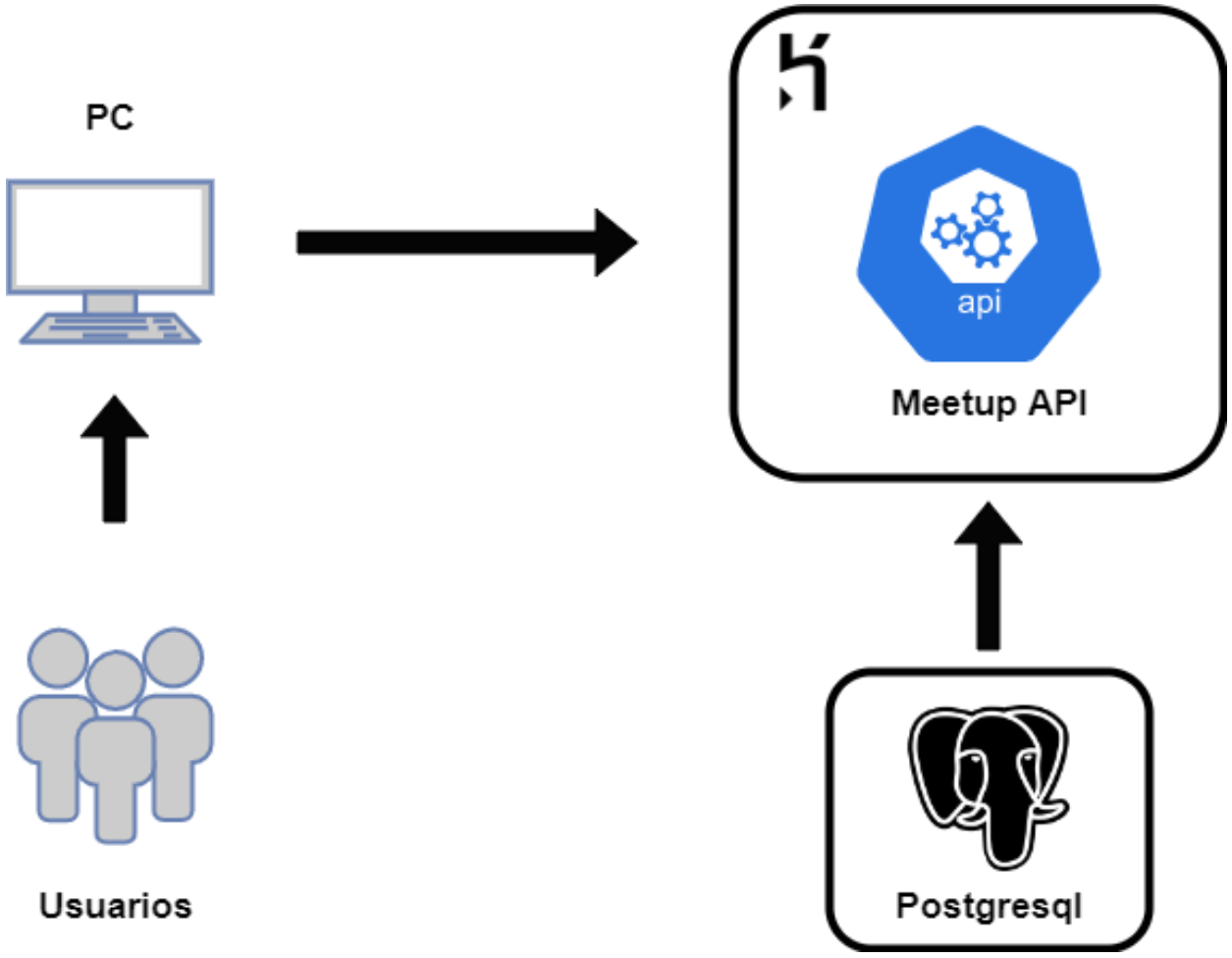


Diagrama Entidad Relación

Base de datos Meetup

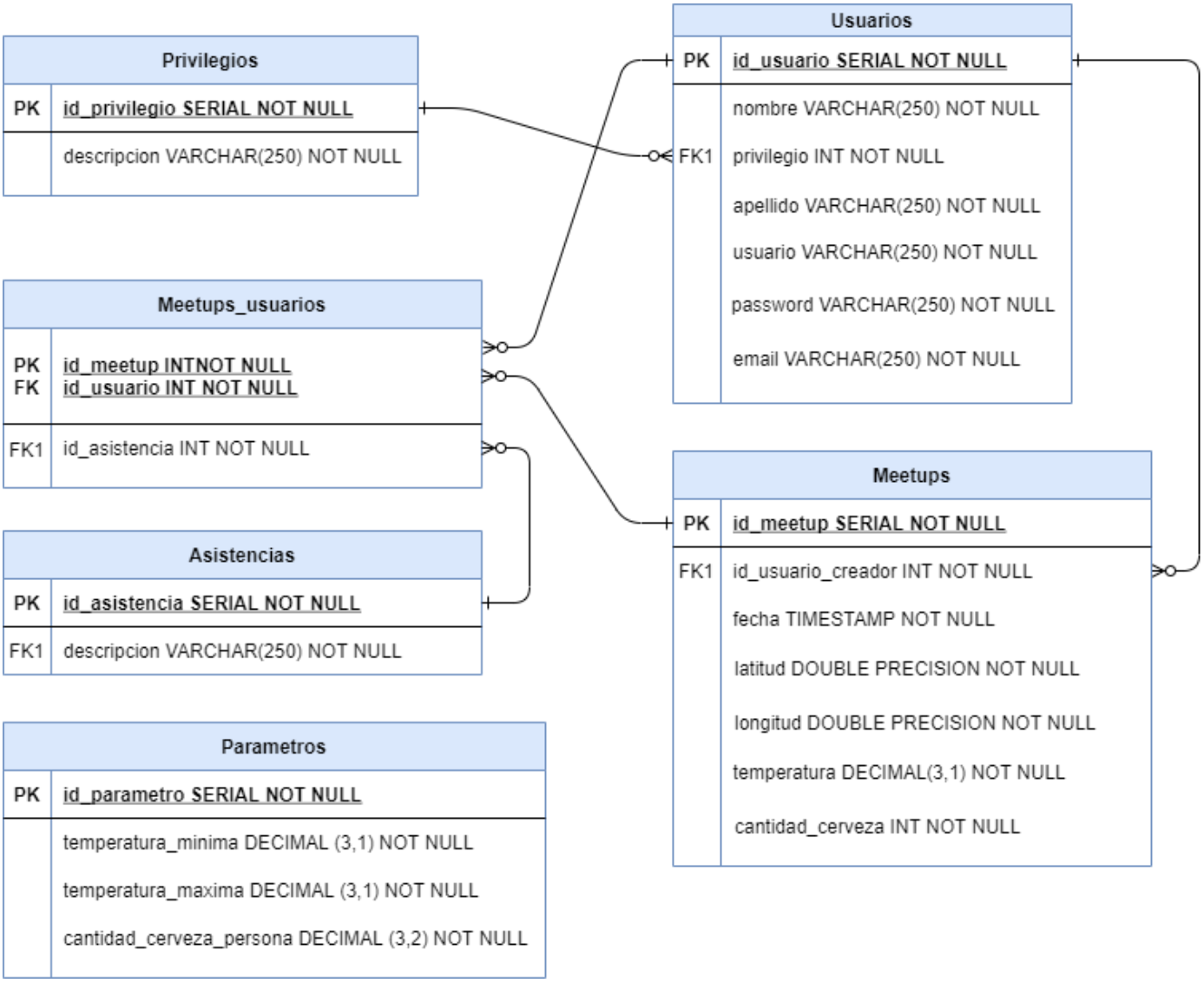
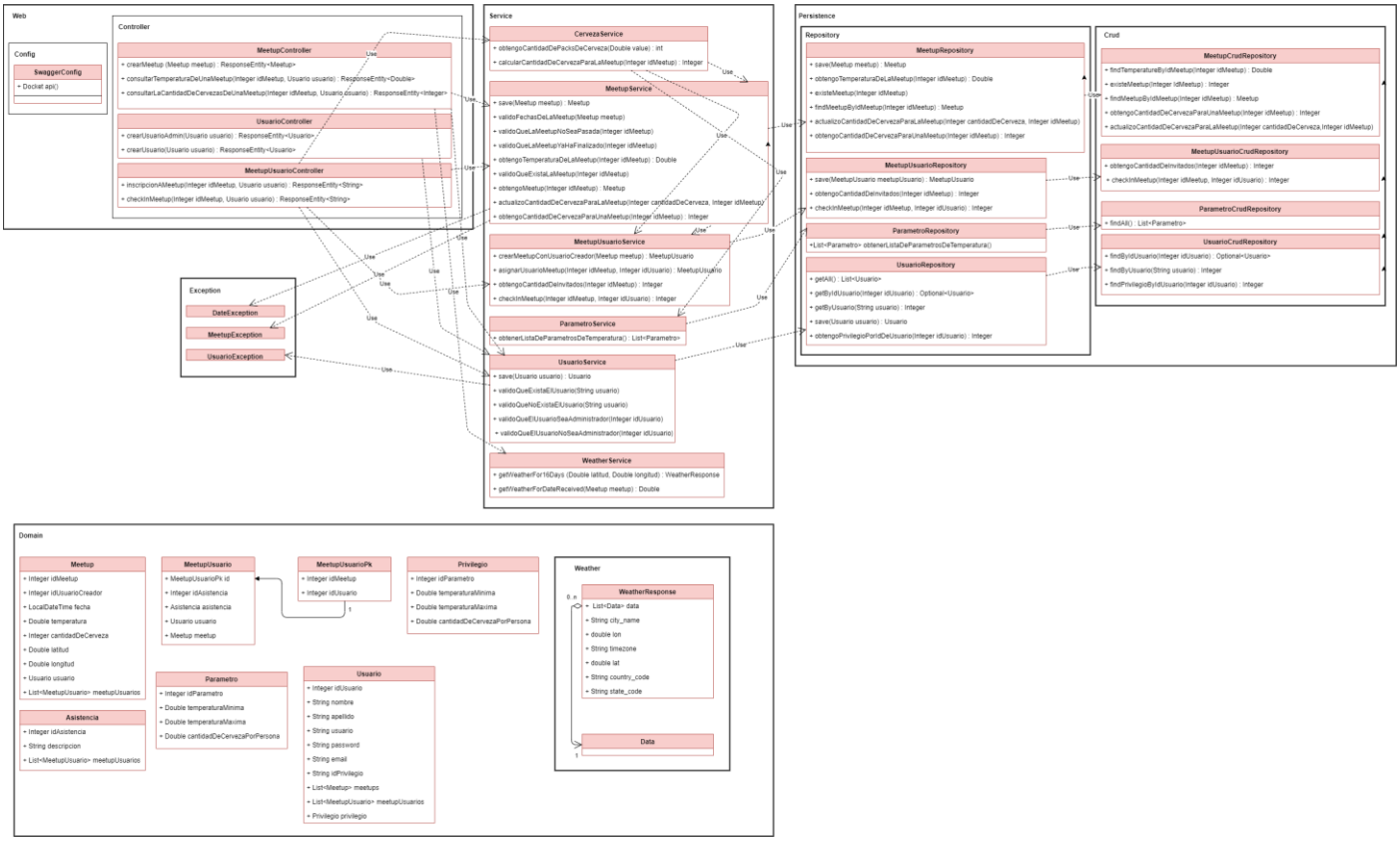


Diagrama de clases



Ejemplos para probar la API

Una de las herramientas más famosas para probar a alto nivel de manera fácil y rápida es Postman.

Nota: Quizas la primer request que se realice a cada API falle por timeout. ¿Por qué sucede esto? Se debe al servicio de hosting gratuito de heroku, este controla las APIs y cuando una pasa a estar inactiva por, aproximadamente, 5 minutos, pasa esa API a un estado "asleep" (dormida). Esto provoca que, cuando una API que se encuentra en ese estado reciba un request, heroku cambia el estado de la misma a "awake". Este cambio de estado conlleva el levantar todo el contexto de la API, lo que lleva un tiempo significativo

Conclusión

A modo de cierre, me gustaría compartir que cosas hubiese agregado y corregido ya que calculé mal el tiempo, y cuando me di cuenta solo me faltaban 2 horas para que finalice el hackerrank.

- En algunas clases hice la inyección del mismo service sin darme cuenta (no lo pude corregir)
- Una entidad la cual tiene dos atributos de fecha, también olvidé eliminarlo.
- Hay servicios que devuelven un String (obtener temperatura, hacer check-in y asignarme a una meetup) hubiera sido mejor devolver un objeto con más información (timestamp,message,etc).
- La idea del usuario y el password fueron implementadas para crear una entidad credencial, luego crear un controlador de autorización y más tarde generar un JWT para que el usuario se comunique con un token. No llegué a hacerlo, y algunos servicios reciben el idUsuario o el Usuario para funcionar, quedó bastante desprolijo para ser cierto.
- La entidad usuario posee un atributo e-mail, estaba pensado para en caso de modificación o creación de meetups se les envíe un e-mail a los usuarios. (Punto notificaciones)
- Hay un servicio aclarado más arriba donde se espera un Usuario y no debería esperarse.
- Utilicé nombres en castellano lo más descriptivos posibles basados en lecturas de Clean Code.
- Intenté utilizar a mi criterio el principio de responsabilidad única en todo el proyecto.
- Tampoco llegué a hacer test unitarios en la sección de tests.
- Faltó la utilización de logs a nivel aplicación como a la BD.

Por ultimo. ¡Muchas gracias!