

UN PROYECTO DE

UNIVERSIDAD DE MÁLAGA

SoftPro

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA



REPOSITORIO: <https://github.com/chemafdezuma/softpro.git>

ÁLVARO YUSTE MORENO
JOSE LUIS LÓPEZ RUIZ
PABLO ALARCÓN CARRIÓN
ADRIÁN CAMACHO FERREZUELO
MARTA GONZÁLEZ PALMERO

YUSTE@UMA.ES
PEPEFXLR@UMA.ES
PAPROKA@UMA.ES
ADRICAM@UMA.ES
MGONZALEZPALMERO@UMA.ES

PLANIFICACIÓN DEL PROYECTO

ÁLVARO VALENCIA VILLALÓN
RAFAEL CEBALLOS MARTÍNEZ
ROCÍO GÓMEZ MANCEBO
JOSE MARÍA FERNÁNDEZ CANTÓN

VALEAL@UMA.ES
RAFACEBAMAR@UMA.ES
ROCIOGOOMEZ27@UMA.ES
JOSEMARIACANTON@UMA.ES



ÍNDICE

Introducción	2
Roles	2-3
Aplicación	3
Uso de Scrum	3-4
Gestión del riesgo	5
Herramientas usadas	6
Requisitos	7-16
Casos de uso	16-26
Diagrama de Clases	27
Pruebas JUnit	28-29



INTRODUCCIÓN

EXPLICACIÓN DEL PROBLEMA

Málaga es una ciudad que hoy en día recibe a miles de turistas, estudiantes, empresarios... Cada uno llega con un propósito específico, pero una vez aquí resulta inevitable querer conocer un poco más los sitios y oportunidades que puede ofrecer. Nuestro objetivo es llevar a cabo una aplicación que, a través de ciertas preguntas como número de personas para el plan, tipo de plan, dinero que se quiere gastar, el tiempo que hace, ubicación... Proponga al usuario el mejor plan que puede ofrecer la ciudad.

ROLES DEL EQUIPO

Los diferentes roles que se han asignado son, básicamente, 6:

- PO (Product Owner): se encargará principalmente de guiar el diseño del producto, con una meta clara y concisa. Representa a los usuarios y clientes del producto en cuestión
- SM (Scrum Master): permite que los miembros del equipo tengan las herramientas necesarias para trabajar correctamente, y modera las reuniones que tenga el equipo. Además, hace de coach y de administrador del proyecto.
- Equipo de desarrollo
 - Code (Programador): como su nombre indica, se encarga de la programación front-end y back-end del proyecto en cuestión, así como todo lo complementario a código que sea necesario.
 - Testing (Pruebas): se encarga principalmente de poner a prueba las diferentes versiones que tenga el producto, para así encontrar posibles bugs y fallas del mismo. Este rol es esencial dentro de la calidad a presentar del producto final.
 - Graphic Design: Se encarga de diseñar los elementos gráficos (entre ellos, la presentación en diapositivas).
 - App Design: Su función es la organización, percepción visual y elementos adicionales que formarán parte de la percepción visual de la aplicación.
 - Risks (Posibles riesgos): El atributo que se le asigna se basa en buscar los riesgos principales que tenga cada fase del desarrollo, y notificar acerca de posibles soluciones a los mismos, en caso de que ocurriesen.
 - Analyst (Analista): Identificar los requisitos que tendrá la aplicación coordinándose con el Product Owner.
 - Structural Designer: Idea y desarrolla los modelos necesarios para llevar a cabo el producto a bajo nivel.



Ya explicados, los miembros del grupo poseen los siguientes roles:

- Rocío Gómez Mancebo: PO / Testing
- Alvaro Yuste Moreno: Analyst / Structural Design
- Álvaro Valencia Villalón: Scrum Master / Graphic Design
- Marta González Palmero: Scrum Master / Testing
- Adrián Camacho Ferrezuelo: Analyst / Structural Design
- Pablo Alarcón Carrión: Code Specialist / Testing
- José María Fernández Cantón: Code Specialist / Structural Design
- José Luis López Ruiz: Code Specialist / App Design
- Rafael Ceballos Martínez: Graphic Design / Risks

Debido a la alta demanda de desarrolladores de código y teniendo en cuenta el equipo que tenemos, todos pertenecemos al departamento "code". También dandonos cuenta nuestra inexperiencia diseñando y que cada uno tiene ciertas habilidades al respecto, hemos decididos fraccionar el departamento de "design" como viene reflejado arriba.

APLICACIÓN LOGO Y NOMBRE

El nombre 'where2go' se ha elegido teniendo en cuenta la funcionalidad principal de la aplicación, encontrar un plan acorde a tus necesidades.

El diseño del logotipo reúne todas las partes del nombre de la aplicación, en forma de símbolo abstracto y moderno.



'Where2go'

USO DE SCRUM Y POR QUÉ LO HEMOS ELEGIDO

Creemos que este es el método más adecuado para este proyecto porque fomenta el trabajo en equipo, nos permite realizar cambios durante el proceso del proyecto y favorece el contacto con el cliente en todo momento.

Seguramente una vez terminado el proyecto habremos aprendido mucho sobre la metodología, las necesidades y habilidades de desarrollo de un proyecto y las capacidades necesarias para el trabajo en grupo, la colaboración y la planificación de actividades complejas.



Concretando, nuestra estrategia será la siguiente:

- Sprints de 1 semana de duración
- Sprint Planning (reunión grande) los lunes.
- Sprint Review Meeting (reunión de 15 minutos aproximadamente) los domingos

The screenshot shows a Product Backlog board with several columns representing different sprints and a sprint backlog. The columns are:

- Sprint 2**: Contains tasks like "INFORME SECRETO (INDIVIDUAL)" and "Desarrollar los requisitos y clasificarlos en RF y RNF".
- Sprint 3**: Contains tasks like "Hacer diagramas MagicDraw de los casos de uso" and "Poner las historias de usuario en tarjetas".
- Sprint 4**: Contains tasks like "Subir el documento actualizado" and "Actualizar historias de usuario y tarjetas de requisito".
- Sprint 5**: Contains tasks like "NOTA" and "CASOS DE USO".
- Sprint 6 // Corrección**: Contains tasks like "Comprobar que Diagrama coincide con partes del código" and "ESTAR ANIMADOS PORQUE NUESTRA APP VA GENIAL".
- Sprint 7**: Contains tasks like "Actualizar el documento" and "Actualizar diagrama de secuencia CUI1".

Each task card includes details such as status (e.g., TERMINADO, EN ESPERA), due dates, and assignees (e.g., RG, JC, MG, RM, JR). A sidebar on the right shows an "Índice de contenidos" (Table of Contents) and a "Crear presentación del trabajo" (Create presentation of the work) button.



GESTIÓN DEL RIESGO

TIPO, PROBABILIDAD, EFECTOS Y MITIGACIÓN

Tipo	Riesgos	Probabilidad	Efectos	Mitigación
Proyecto	Poca experiencia organizativa	Muy alta	Tolerable	Aprender lo máximo posible del profesor
Proyecto	Mala planificación del tiempo invertido en tareas	Alta	Serio	Tratar de evitar las máximas dependencias posibles y reducir el camino crítico en la planificación
Proyecto	Imposibilidad de implementación de las herramientas CASE	Alta	Tolerable	Tener varias herramientas disponibles por si una falla
Proyecto y producto	Cambio de diseño para implementar las funciones requeridas	Moderada	Serio	Optimizar lo máximo el código para que no suponga una pérdida de tiempo grande
Producto	La base de datos usada en el sistema no es capaz de procesar tantas transiciones	Moderada	Serio	Plantear la compra de una base de datos con mas capacidad y rendimiento
Proyecto y producto	Cambios repentinos en los requisitos	Moderada	Serio	Evaluar el impacto de los requisitos para ver si conviene o no
Negocio	Otra compañía lanza una aplicación con el mismo objetivo	Baja	Tolerable	Tratar de implementar más funciones llamativas sin que afecten al tiempo de entrega
Proyecto	Personal clave está enfermo o no esta disponible en momentos críticos	Moderada	Tolerable	Reorganizar el equipo para que haya más miembros que conzcan el trabajo del resto
Proyecto y producto	Componentes reutilizados limitan la funcionalidad	Muy baja	Insignificante	Modificar el código reutilizado para adaptarlo a las necesidades



HERRAMIENTAS USADAS

HASTA LA FECHA

- Métodos de comunicación
 - Discord (Reuniones)
 - WhatsApp
- Trabajo colaborativo
 - Git/GitHub
 - Trello
- Elaboración de documentos
 - Microsoft Word
 - Pages
- Diseño gráfico
 - Adobe Photoshop
 - Affinity Designer
 - Pencil 2D
 - Procreate
- Diagrama de requisitos, clases y secuencia
 - MagicDraw
- Programación de la aplicación
 - Eclipse (Spring/Java)
 - Visual Studio Code (CSS y HTML)
- Diseño de presentación
 - Key note
 - Jitter



REQUISITOS

FUNDAMENTALES PARA EL PROYECTO

Los requisitos son descripciones de los servicios que un sistema debe proporcionar y las restricciones a su modo de operación.

Un requisito es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Se usa en un sentido formal en diversas ingenierías.

En este apartado vamos a mostrar los requisitos en forma de historias de usuario, ya que proporcionan un “enfoque ligero” para gestionar los requisitos de un sistema, son una pequeña pieza de funcionalidad que añade valor al negocio.

Los acompañamos con pruebas de aceptación, que detallan ejemplos de cómo interactuará cierto sujeto con la aplicación centrándose en el uso que le dará al requisito en cuestión.

FUNCIONALES (RF)

Describen la funcionalidad o los servicios que se espera que el sistema suministre

RF1

Título del lugar

Como usuario **quiero** saber el nombre del lugar **para** poder identificarlo y buscarlo.

Pruebas de aceptación

- El usuario busca “Parque del Oeste” por su nombre en el buscador y este sitio le aparece.



RF2

Descripción del lugar

Como usuario **quiero** ver una descripción del lugar **para** ver con más detalle como es el sitio.

Pruebas de aceptación

- El usuario ve la siguiente descripción del restaurante "La Tagliatella": "Restaurante de comida italiana de calidad".

RF3

Características del lugar

Como usuario **quiero** saber información extra del sitio **para** poder confirmar si el lugar se adapta a lo que busco

Pruebas de aceptación

- El usuario ve las siguientes características del "Centro comercial Vialia Málaga": Tiene parking, dispone de un supermercado Mercadona en su interior, dentro hay un cine...



RF4

Valoraciones

Como usuario **quiero** dar una valoración sobre un lugar al estilo de "5 estrellas"**para** dar mi opinión del lugar.

Pruebas de aceptación

- El usuario clickea un lugar y le aparece la opción de añadir valoración.

RF5

Comentarios

Como usuario **quiero** poder dar mi opinión de un lugar mediante un comentario **para** aportar a la descripción del mismo.

Pruebas de aceptación

- Cuando el usuario ha añadido la valoración del lugar también le aparece la opción de añadir un comentario, escribe su opinión y publica el comentario de manera que sea visible para el resto de los usuarios.



RF6

Gestión de contenidos

Como moderador **quiero** poder eliminar los comentarios que vea inapropiados (comentarios que no estén relacionados con el lugar) **para** que los comentarios que haya sobre un lugar no hablen de cosas ajena a este.

Pruebas de aceptación

- El moderador selecciona un comentario, selecciona la opción de eliminar y este deja de ser visible por el resto de los usuarios.

RF7

Imagen del sitio

Como usuario **quiero** poder ver imágenes del lugar que he seleccionado **para** ver el aspecto del lugar.

Pruebas de aceptación

- El usuario busca un lugar y le aparecen diferentes imágenes del establecimiento, pero las imágenes han de ser de ese local en concreto. De esta manera aporta a la descripción del sitio.



RF8

Búsqueda del lugar

Como usuario **quiero** buscar un nombre de un lugar **para** que me aparezcan sitios destacados a los que visitar en ese sitio.

Pruebas de aceptación

- El usuario realiza una búsqueda de Marbella y aparecen sitios con una puntuación buena y que puedan adaptarse al usuario

RF9

Búsqueda de tipo de lugar

Como usuario **quiero** poder buscar un tipo de lugar (restaurante, cine, recreativos) **para** que me aparezcan ese tipo de lugar en un radio acordado.

Pruebas de aceptación

- El usuario realiza una búsqueda de "Cine" y le aparecen todos los cines que existan en el perímetro marcado.



RF10

Restringir el precio

Como usuario **quiero** poder señalar un rango de precios **para** que me aparezcan sitios que se encuentren en ese margen de precios preestablecidos.

Pruebas de aceptación

- El usuario realiza una búsqueda de "Cine" y marca un precio de 5-8 euros y aparecen únicamente aquellos cines que cumplen esta característica.

RF11

Tiempo

Como propietario de la app **quiero** que aparezca en todo momento el tiempo y de forma actualizada el tiempo que hace en la ubicación del usuario, **para** que en función de este el usuario pueda decidir mejor que sitio elegir.

Pruebas de aceptación

- El usuario puede observar que el tiempo se mantiene actualizado.
- El usuario puede ver el tiempo del lugar en el que se encuentra actualmente.
- El usuario puede acceder sin problemas y de manera rápida al tiempo.



RF12 Mapa

Como programador **quiero** que la aplicación tenga una redirección a la aplicación de mapas **para** simplificar mucho nuestro trabajo y para que sea mucho más fácil guiarse con una aplicación local.

Pruebas de aceptación

- El usuario podrá ver en la parte inferior del sitio seleccionado un mapa pequeño con la ubicación del lugar, o bien un botón que, cuando lo pulse, lo redirigirá a su aplicación predeterminada de mapas.

RF13 Ubicación

Como usuario **quiero** que mi aplicación detecte mi ubicación actual **para** que me recomiende lugares según esta característica.

Pruebas de aceptación

- El usuario realiza distintas búsquedas, en las que pone como requisito que los lugares estén a menos de dos kilómetros. La aplicación recomienda lugares que satisfagan esa característica.



RF14

Lugares de interés

Como usuario **quiero** que los sitios que se me recomiendan estén acordes a los filtros que he puesto **para** poder elegir entre uno de esos sitios sin tener que buscar manualmente cuáles me conviene.

Pruebas de aceptación

- El usuario al meter los filtros puede elegir entre los lugares que aparecen rápidamente sabiendo que todos ellos cubren sus necesidades, por lo que el usuario tiene una buena experiencia usando la aplicación.

NO FUNCIONALES (NRF)

Restricciones sobre los servicios o funciones ofrecidas por el sistema.

RNF1

Diseño visual atractivo

Como diseñador gráfico **quiero** poseer un diseño minimalista pero atractivo de la aplicación mediante los distintos diseños realizados **para** que el usuario se sienta cómodo visualmente hablando mientras utiliza nuestra aplicación.

Pruebas de aceptación

- El usuario se registra y la pantalla de carga es adecuada.
- Al realizar búsquedas los distintos lugares de acceso de la app se encuentran acordes y con colores que crean una armonía visual.
- El usuario usa la aplicación y no se cansa gracias a las características visuales.



RNF2

Fácil uso de la app

Como programador **quiero** conseguir que el usuario sea capaz de utilizar la aplicación de manera intuitiva maximizando las funcionalidades de la aplicación sin perder la esencia de sencillez **para** conseguir que el usuario esté conforme con la aplicación

Pruebas de aceptación

- Se registra un cliente nuevo y es capaz de hacerlo de manera rápida y sencilla.
- Al navegar por la aplicación los tiempos de carga son mínimos.
- Se representa de manera visual las distintas opciones que puede realizar el usuario a la hora de buscar un lugar.

RNF3

Tamaño

Como programador **quiero** que la aplicación ocupe el menor espacio posible **para** que los usuarios puedan descargarla más fácilmente y sea más liviana para la memoria interna del dispositivo.

Pruebas de aceptación

- El usuario que descargue la aplicación verá reflejado un bajo peso de la misma por la optimización de la resolución de las imágenes.
- Cada vez que ocurran cambios de datos de la aplicación, el usuario podrá ver esa actualización, evitando así crear nuevas versiones de la aplicación.



RNF4

Rendimiento

Como programador **quiero** que la aplicación tenga mejor rendimiento **para** que pueda funcionar correctamente en todos los dispositivos, y que sea fluida.

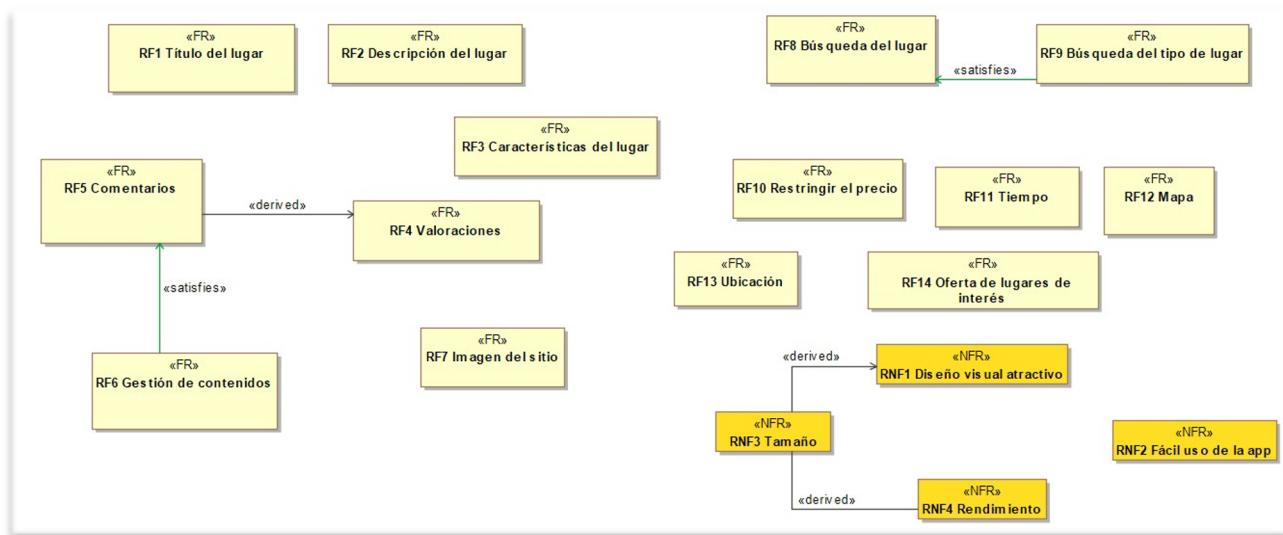
Como usuario **quiero** que la aplicación tenga mejor rendimiento **para** que sea más cómoda y rápida de usar.

Pruebas de aceptación

Este requisito va conjunto al tamaño, por ende:

- El usuario dispondrá de imágenes de menor calidad para que funcionen con mayor fluidez, con efectos reducidos y funciones mínimas.
- El usuario verá la aplicación con una cantidad pequeña de texto para su fácil lectura, así como mejor rendimiento.
- Internamente, la aplicación procesa menos complejidades para una optimización de la misma.

ESQUEMA DE LOS REQUISITOS



Esquema en MagicDraw de los requisitos y las relaciones entre ellos



CASOS DE USO DEL SOFTWARE

Los diagramas de casos de uso muestran la funcionalidad del sistema desde el punto de vista de un observador externo.

ACTORES

- Usuario (el que usa la app)
- Administrador (Encargados de los aspectos técnicos)
- Moderadores (Conocedores de la ciudad, encargados de sugerir lugares y eliminar comentarios ofensivos)

CASOS DE USO

- Usuario
 - Añadir una valoración o comentario
 - Añadir características del lugar, cuestionarios como google
 - Buscar nuevos planes para un día concreto
 - Sugerir nuevos lugares no puestos anteriormente (Han de ser aprobados)
- Administrador
 - Solucionar problemas puntuales
 - Aprobar y banear moderadores
- Moderador:
 - Añadir sugerencias semanales (Posibles campañas publicitarias).
 - Semanalmente, revisar lugares, comentarios y planes nuevos.
 - Actualizar sobre problemas o eventos en el lugar (Ej.: Obras de teatro, reparaciones en el lugar, etc.)



CUI1

Identificador único: Añadir una valoración o comentario.

Contexto de uso: Cuando un usuario quiere agregar su opinión sobre un sitio que ya haya visitado

Precondiciones y activación: El usuario está conectado y ha visitado el lugar recientemente.

Garantías de éxito: Se agrega una opinión con las huellas (estrellas) con o sin comentario de un lugar.

Escenario principal:

1. El usuario visita un lugar
2. El usuario quiere recomendar un sitio
3. El usuario busca el sitio
4. El usuario opina
5. Se publica

Escenarios alternativos: El moderador al cierto tiempo ve el comentario inadecuado y lo elimina.

CUI2

Identificador único: Añadir características del lugar, cuestionarios como google.

Contexto de uso: Un usuario recibe una notificación en el móvil para responder ciertas preguntas relacionadas con el sitio, tales como si tiene aparcamientos cerca, etc.

Precondiciones y activación: El usuario ha visitado el lugar recientemente y recibe una notificación para llenar una encuesta

Escenario principal:

1. El usuario visita un lugar
2. El usuario recibe una notificación de encuesta
3. El usuario realiza la encuesta

Escenarios alternativos: El usuario decide no hacer la encuesta



CUI3

Identificador único: Buscar nuevos planes para un día concreto.

Contexto de uso: El usuario quiere hacer planeas un día concreto y se mete en la app para buscar ideas.

Precondiciones y activación: El usuario desea buscar un lugar que visitar.

Garantías de éxito: La aplicación recomienda sitio al usuario.

Escenario principal:

1. El usuario entra en la APP.
2. Busca un tipo de sitio (restaurante, concierto o búsqueda general).
3. El usuario observa las distintas posibilidades.
4. El usuario va al sitio.

Escenarios alternativos: La app no va y no da resultados, el usuario no se decide por ningún plan que hay propuesto.

CUI4

Identificador único: Sugerir nuevos lugares no puestos anteriormente (Han de ser aprobados).

Contexto de uso: El usuario agrega un lugar al que visitar no registrado en nuestra base de datos.

Precondiciones y activación: El usuario desea agregar ese lugar y le da a la pestaña de proponer sitio.

Garantías de éxito: Se recibe una petición de añadir ese sitio a la base de datos.

Escenario principal:

1. El usuario visita un sitio
2. Abre la app
3. Trata de aplicar para agregar un sitio
4. El moderador la recibe y la procesa
5. Se agrega el sitio

Escenarios alternativos: El lugar es un sitio falso, el lugar ya ha sido agregado.



CUI5

Identificador único: Sugerir nuevos planes a un lugar (Han de ser aprobados).

Contexto de uso: El usuario agrega un plan que realizar no registrado en nuestra base de datos.

Precondiciones y activación: El usuario desea agregar ese plan y le da a la pestaña de proponer lugar.

Garantías de éxito: Se recibe una petición de añadir ese plan a la base de datos.

Escenario principal:

1. El usuario visita un sitio.
2. Abre la app.
3. Trata de aplicar para agregar un sitio.
4. El moderador la recibe y la procesa.
5. Se agrega el sitio.

Escenarios alternativos: Sea un plan falso, ya este ese sitio agregado o sea un plan muy aburrido y el moderador no lo acepte.

CUI6

Identificador único: Solucionar problemas puntuales.

Contexto de uso: Se notifica un error en la aplicación web y el administrador se dispone a solucionarlo.

Precondiciones y activación: Hay un error notable.

Garantías de éxito: Ese error no se repite más.

Escenario principal:

1. Se recibe una advertencia de error.
2. El moderador la procesa.
3. Se soluciona el error.

Escenarios alternativos: los administrador no son capaces de resolver ese error.



CUI7

Identificador único: Aprobar y banear moderadores.

Contexto de uso: Se necesitan moderadores para tener la app en buen estado.

Garantías de éxito: Se buscan moderadores suficientes.

Escenario principal:

1. Nos damos cuenta de que hay muchas peticiones y propuestas.
2. Buscamos moderadores.
3. Se lee las diferentes aplicaciones desde nuestra app.
4. Se aceptan ciertas solicitudes.

Escenarios alternativos: No encontramos moderadores aptos para la tarea.

CUI8

Identificador único: Añadir sugerencias semanales (Posibles campañas publicitarias).

Contexto de uso: Promocionar nuevas actividades para la app.

Garantías de éxito: La app tiene más movimiento..

Escenario principal:

1. El moderador abre la app.
2. Crea ciertos eventos (en base a valoraciones anteriores o a raíz de campañas publicitarias).
3. la gente consulta esas sugerencias y las toma.

Escenarios alternativos: Las sugerencias son deficientes o al moderador se le olvida realizar estas sugerencias.

CUI9

Identificador único: Semanalmente, revisar lugares, comentarios y planes nuevos.

Contexto de uso: Revisar las diferentes peticiones de los usuarios.

Garantías de éxito: La bandeja de peticiones este vacía.

Escenario principal:

1. Cada ciertos días el moderador abre su bandeja de propuestas.
2. Las analiza y ve cuales son factibles y cuales no.
3. Las agrega a la app.
4. Acepta las opiniones.
5. Descarta las opiniones o lugares o planes malos por diferentes motivos (ya mencionados antes).

Escenarios alternativos: Sean tantas las sugerencias que sea imposible analizarlas todas.



CUI10

Identificador único: Actualizar sobre problemas o eventos en el lugar (Ejemplos: Obras de teatro, reparaciones en el lugar, etc.)

Contexto de uso: Había un evento externo el cual impedía acceder a cierto sitio.

Garantías de éxito: Se actualizó.

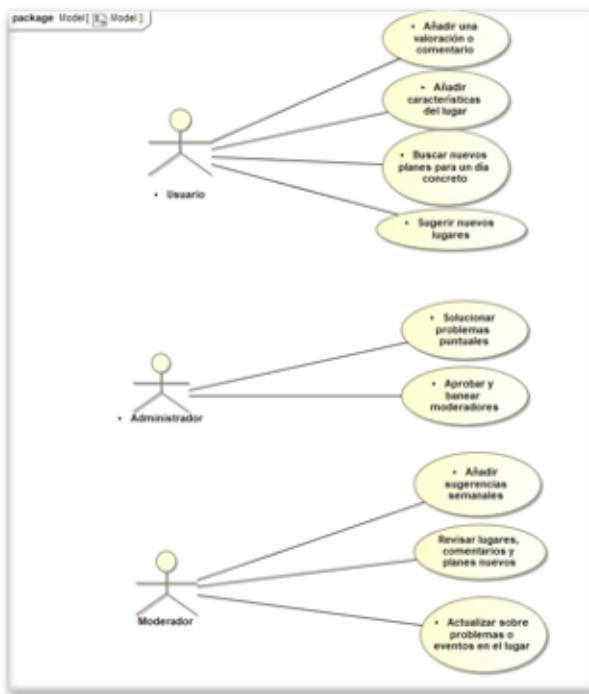
Escenario principal:

1. previamente se había agregado un problema que impedía realizar cierto plan.
2. Se acaba este problema.
3. El moderador entra y borra el evento el cual molestaba.
4. Se puede volver a realizar ese plan.

Escenarios alternativos: Se le olvide borrar al moderador.

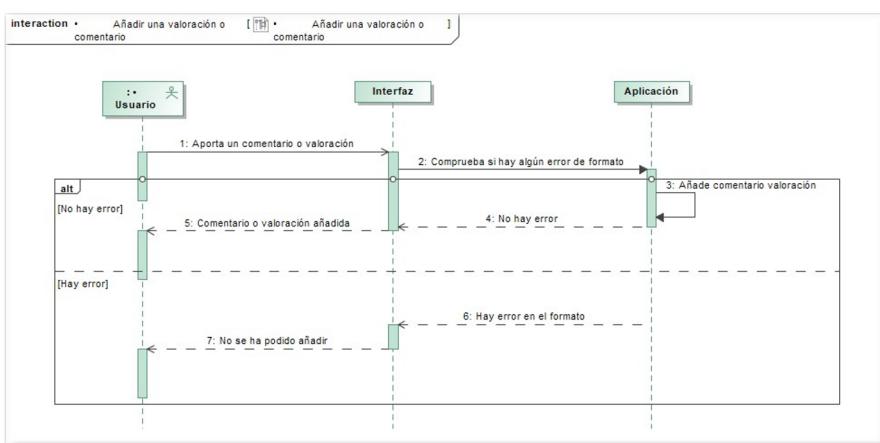


DIAGRAMA DE CASOS DE USO

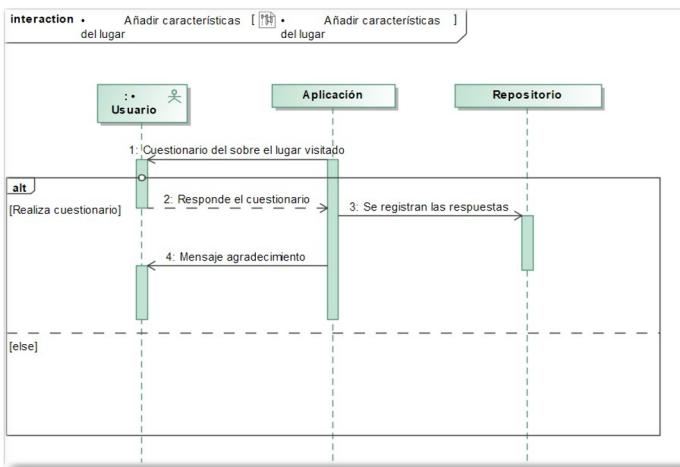


DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia muestran la interacción de un conjunto de objetos enfatizando el orden en el tiempo. Permite refinar los casos de uso incluyendo detalles de implementación como los objetos y clases usados y los mensajes intercambiados entre los mismos.



(CUI 1) El usuario accede al lugar visitado recientemente y en el apartado de valoraciones y comentarios dejara sus respectivas opiniones sobre el lugar, siendo la valoración permitida de 1 a 5. Si hay algún error de formato como que el usuario intente añadir un comentario vacío, este se descartará

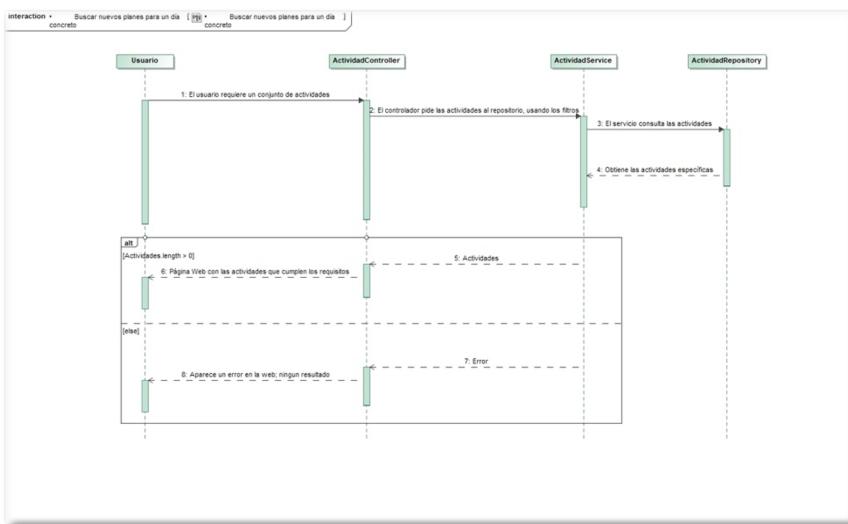


(CUI 2) El usuario recibe una notificación del lugar que ha visitado gracias a las recomendaciones de la aplicación,

se le facilita un cuestionario en el que hay preguntas sobre el lugar visitado.

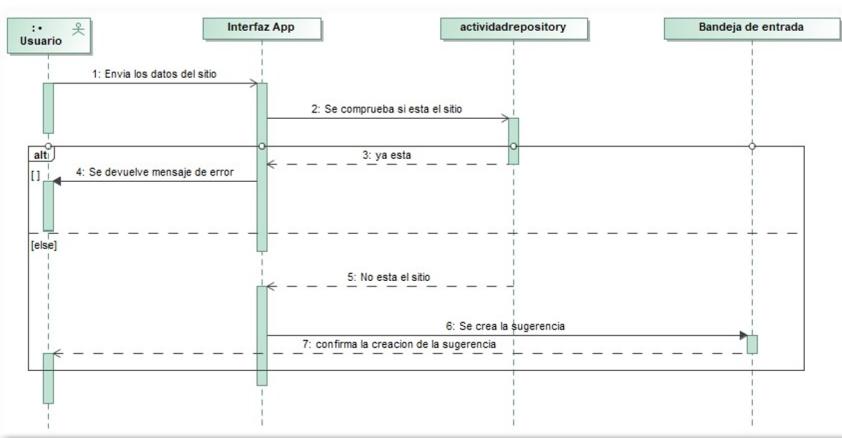
Si el usuario responde correctamente, se registran las respuestas y se agradece la colaboración al usuario,

por el contrario, si el usuario ignora la notificación, no se hará nada al respecto.



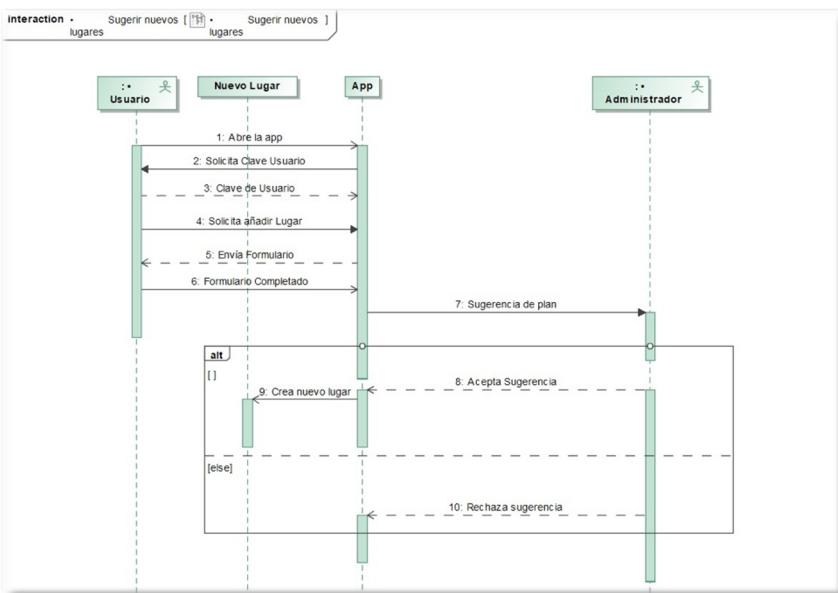
(CUI 3) El usuario pedirá al ActividadController que obtenga las actividades, solicitando a ActividadService que el repositorio de actividades (ActividadRepository) le devuelva un conjunto de actividades que cumplan el filtro elegido por el usuario, en este caso cumpliendo el requerimiento de que sea apto para ese día.

Si el conjunto devuelto es vacío, se mostrará un error por pantalla pero si por el contrario hay elementos, se mostraran en ella.



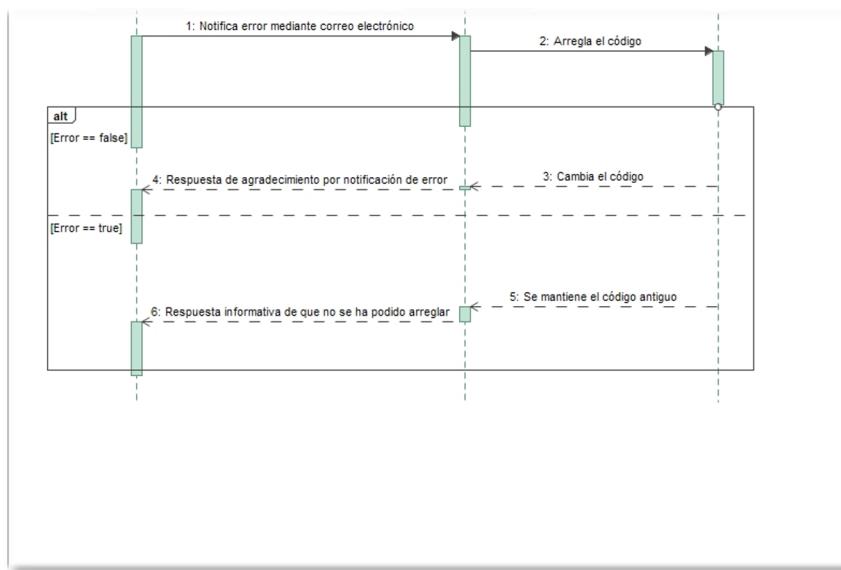
(CUI 4) 1. El usuario enviará los datos por medio de la interfaz de la app, desde aquí se consulta en la base si el sitio ya está agregado o no, el cual más tarde se envía a la bandeja de entrada de solicitudes que más tarde se administrará

2. Si ya está agregado se le dice al usuario que ese sitio ya está.



(CUI 5) El usuario inicia sesión en la app para añadir un nuevo lugar en el que ha estado,

la app le proporciona un formulario y cuando este lo rellena el administrador decide si aceptar o no la sugerencia. Si es aceptada, la app crea el nuevo lugar.

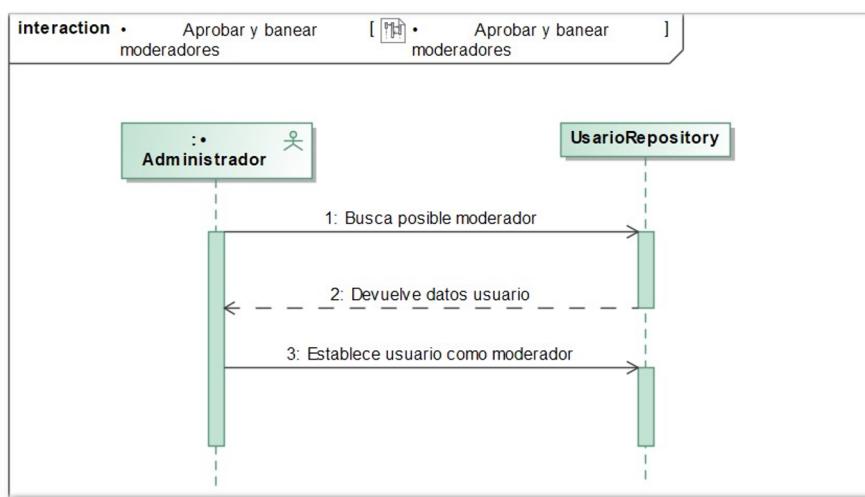


(CUI 6) El usuario notifica un error que haya encontrado en la aplicación mediante correo electrónico.

El correo llega al administrador, que intenta arreglar el código de la app. En caso exitoso,

tras probar que funciona, se le responde al cliente con un correo agradeciendolo su notificación.

En caso contrario, le informamos de que no hemos sido capaces de arreglar el error.



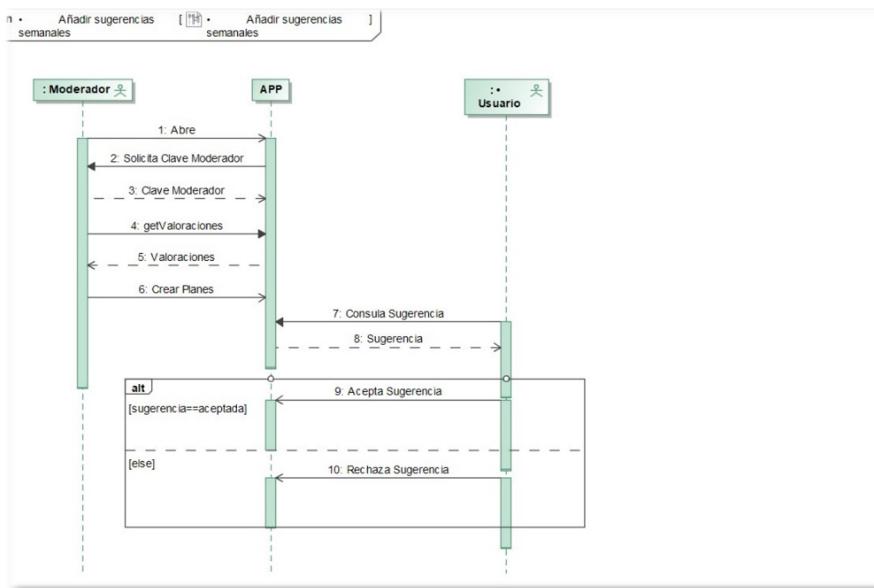
(CUI 7) El administrador pide un usuario, el cual le entregará el UsuarioController, que para encontrarlo

tiene que pedirle al UserService que acceda al repositorio de usuarios y encuentre al susodicho.

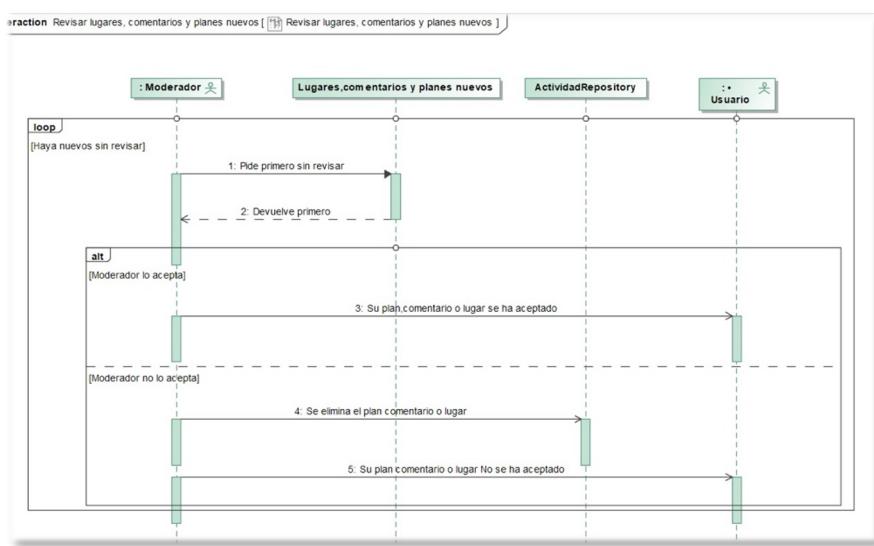
Una vez encontrado (o no) entrega al UserController el usuario. Si le manda al administrador el

usuario pedido, éste procederá a crear o eliminar (según el contexto de la situación) una cuenta de

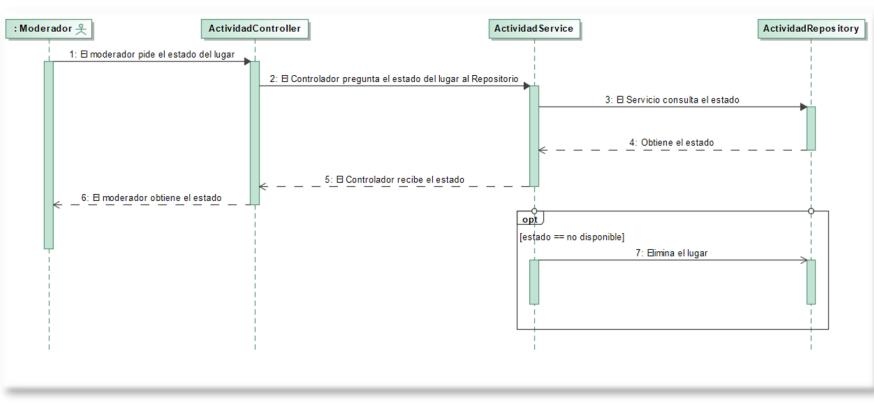
moderador. En caso de que no le llegue ningún usuario, el administrador no podrá hacer nada.



- (CUI 8) 1. El moderador entra a la aplicación y en función de las sugerencias recibidas creará planes.
 2. El cliente consulta las sugerencias creadas.
 3. El cliente decidirá si aceptar o no las sugerencias recomendadas.



- (CUI 9) 1. Es un bucle, el moderador solicita el primer mensaje de la bandeja de entrada, esta se la devuelve, el administrador comprueba si es válida, la implementa en los datos, devuelve que todo está correcto.
 2. Si no es correcta no se agrega y se le envía un mensaje al usuario sobre el motivo.
 3. Si no hay suficiente espacio se le dice al moderador



- (CUI 10) 1. El moderador pide el estado del lugar al ActividadController
 2. El Controlador pregunta por el estado al Repositorio
 3. El ActividadService consulta en el Repositorio el estado
 4. Se obtiene el estado
 5. El Controlador recibe el estado y se lo pasa al Moderador
 6. El Moderador recibe el estado y de esta manera sabe lo que tiene que hacer
 7. Si el estado es "no disponible" se borra el lugar del repositorio, sino no se hace nada



DIAGRAMA DE CLASES

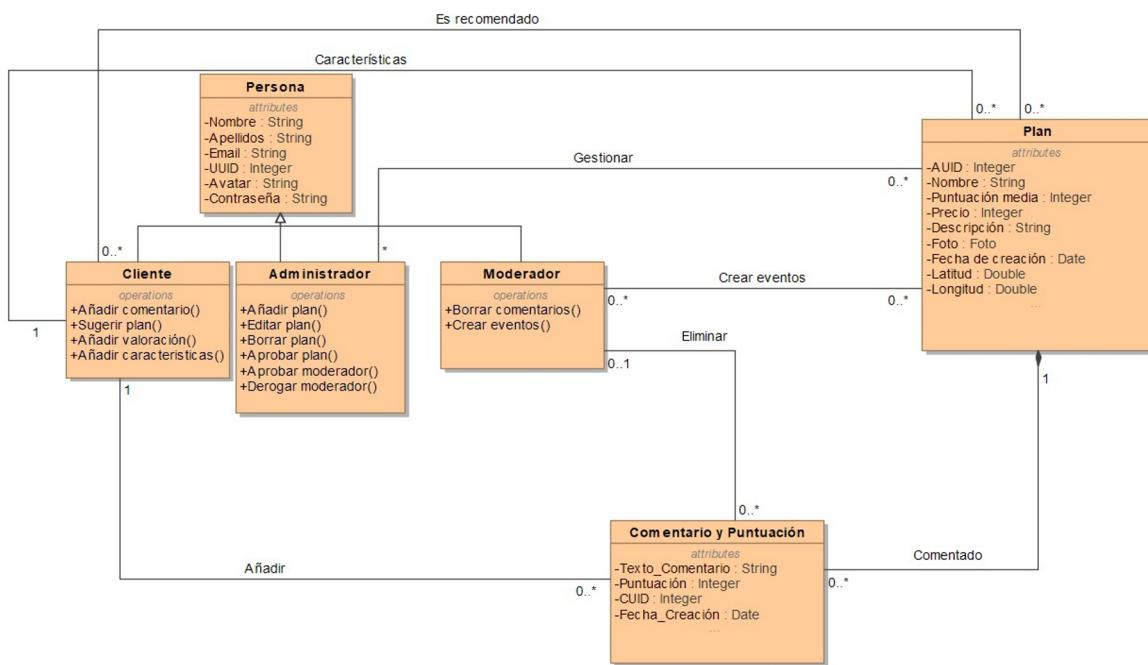


Diagrama de estructura estática que muestra las clases del sistema y sus interrelaciones

PRUEBAS JUNIT

Para comprobar el correcto funcionamiento de una unidad del código de nuestras clases principales (actividad, comentario y usuario) hemos realizado las siguientes pruebas JUnit. Además para algunas de ellas hemos utilizado Mockitos, para simular la creación de algunos objetos.



```

class Where2goApplicationTests {
    UsuarioController user;
    ComentarioController comment;
    ActividadController activity;

    @BeforeEach
    public void init() {
        user = new UsuarioController();
        comment = new ComentarioController();
        activity = new ActividadController();
    }

    @AfterEach
    public void terminated() {
        user = null;
        comment = null;
        activity = null;
    }

    @Test
    public void SiSeEliminaUnaActividadHabiendoCeroLanzaExcepcion() {
        Exception ex = assertThrows(Exception.class, ()→activity.deleteActivity());
        assertEquals("No hay actividades", ex.getMessage());
    }

    @Test
    public void SiSeAnyadeUnaActividadTeniendoMaxElevaExcepcion() throws Exception {
        int x = 10;
        for (int i=0; i<x; i++) {
            activity.anyadirActividad();
        }
        assertEquals(activity.getActividadCounter(),x);
        Exception ex = assertThrows(Exception.class, ()→activity.anyadirActividad());
        assertEquals("Pasa el limite de actividades", ex.getMessage());
    }

    @Test
    public void SiNoHayActividadSeElevaExcepcion() {
        Exception ex = assertThrows(Exception.class, ()→activity.hayActividad());
        assertEquals("No hay actividades", ex.getMessage());
    }

    @Test
    public void InicialmenteElNumeroDeActividadesEsCero() {
        assertEquals(activity.getActividadCounter(), 0, "Mensaje error");
    }

    @Test
    public void SiSeEliminaUnUsuarioHabiendoCeroLanzaExcepcion() {
        Exception ex = assertThrows(Exception.class, ()→user.deleteUsuario());
        assertEquals("No hay usuarios", ex.getMessage());
    }

    @Test
    public void SiNoHayUsuariosSeElevaExcepcion() {
        Exception ex = assertThrows(Exception.class, ()→user.hayUsuario());
        assertEquals("No hay usuarios", ex.getMessage());
    }

    @Test
    public void InicialmenteElNumeroDeUsuariosEsCero() {
        assertEquals(user.getUsuarioCounter(), 0, "Mensaje error");
    }

    @Test
    public void SiSeAnyadeUnUsuarioTeniendoMaxElevaExcepcion() throws Exception {
        int x = 100;
        for (int i=0; i<x; i++) {
            user.anyadirUsuario();
        }
        assertEquals(user.getUsuarioCounter(),x);
        Exception ex = assertThrows(Exception.class, ()→user.anyadirUsuario());
        assertEquals("Pasa el limite de usuarios", ex.getMessage());
    }

    @Test
    public void SiSeAnyadeUnComentarioTeniendoMaxElevaExcepcion() throws Exception {
        int x = 50;
        for (int i=0; i<x; i++) {
            comment.anayadirComentario();
        }
        assertEquals(comment.getComentarioCounter(),x);
        Exception ex = assertThrows(Exception.class, ()→comment.anayadirComentario());
        assertEquals("Pasa el limite de comentarios", ex.getMessage());
    }

    @Test
    public void SiSeGuardaUnComentarioTeniendoMaxElevaExcepcion() throws Exception {
        int x = 50;
        for (int i=0; i<x; i++) {
            Comentario commentMock = mock(Comentario.class);
            comment.guardarComentario(commentMock);
        }
        assertEquals(comment.coments.size(),x);

        //Creamos un nuevo mock
        Comentario commentMock = mock(Comentario.class);

        Exception ex = assertThrows(Exception.class, ()→comment.guardarComentario(commentMock));
        assertEquals("Pasa el limite de comentarios guardados", ex.getMessage());
    }

    @Test
    public void InicialmenteElNumeroDeComentariosEsCero() {
        assertEquals(comment.getComentarioCounter(), 0, "Mensaje error");
    }

    @Test
    public void SiSeEliminaUnComentarioHabiendoCeroLanzaExcepcion() {
        Exception ex = assertThrows(Exception.class, ()→comment.deleteComentario());
        assertEquals("No hay comentarios que borrar", ex.getMessage());
    }
}

```