FACULTAD DE INGENIERÍA

Obligatorio

Ingeniería de Software en la Práctica

Docente: Luis Barrague

Nicolas Damiani - 192489
Sebastian Rodriguez - 192412
Gonzalo Kurin - 192488

1. Indice:

1. Indice: **2**

2. Introduccion **4**

2.1. Análisis del problema y solución propuesta. 4

3. Especificación de requerimientos **6**

3.1. Alcance del producto 6

3.2. Actores del sistema 6

3.3 Requerimientos Funcionales 7

3.3.1. RF01: Registro de nuevo usuario con Facebook 7

3.3.2. RF02: Registro de nuevo usuario sin Facebook 7

3.3.3. RF03: Inicio/Cierre de sesión 7

3.3.4. RF04: Ver perfil de un usuario 7

3.3.5. RF05: Modificar los datos del perfil de un usuario 7

3.3.6. RF06: Crear publicación 8

3.3.7. RF07: Ver mapa de publicaciones 8

3.3.8. RF08: Ver publicación 8

3.3.9. RF09: Modificar publicación 8

3.3.10. RF10: Borrar publicación 8

3.3.11. RF11: Seguir publicación 8

3.3.12. RF12: Comentar publicación 9

3.3.13. RF13: Notificaciones 9

3.3.14. RF14: Exportar fotos de la publicación 9

3.4. Requerimientos no funcionales: **10**

3.4.1. Restricciones generales 10

3.4.2. Restricciones de diseño 10

3.4 Casos de uso: **11**

3.4.1. Registro de nuevo usuario con Facebook 11

3.4.2 Ver Perfil de Usuario 12

3.4.3 Ver Mapa de Publicaciones 13

3.3.4. Modificar Perfil de Usuario 14

3.4.5. Crear publicación 14

3.4.6. Modificar publicación 16

3.4.7. Comentar Publicación 17

3.4.8. Borrar Publicación 18
3.4.9. Ver Publicación 19
3.4.10. Notificaciones 20
3.4.11. Exportar fotos de publicaciones 21
3.4.12. Seguir publicación 21

4. Planificación 22
4.1. Metodología de trabajo 22
4.2. Cronograma de trabajo 23
4.3. Plan de releases 24
4.4. Control de versiones 25

5. Gestión de Riesgos 26
5.1. Identificación y especificación 26
5.2. Análisis cualitativo 27

6. Gestión de la Calidad 29
6.1. Actividades preventivas 29
6.2. Actividades correctivas 29

7. Mockups Moviles 32
7.1. Flujo de navegación 32

8. Evaluación del proceso de desarrollo 33
8.1 Introducción 33
8.2 Alcance 33
8.2.1 Requerimientos funcionales 33
8.2.3 Requerimientos no funcionales 34
Estandares de Clean Code 34
Control de versiones 36
Heurísticas de Nielsen 37
8.3 Cumplimiento de cronograma 39
8.4 Cumplimiento del plan de release 39
8.5 Evaluación de riesgos 39
8.6 Actividades preventivas 41
8.7 Actividades correctivas 41
8.8 Actividades de control 43
8.9 Gestion de cambios 43
8.10 Gestion de defectos 44
8.11 Diseño 46
8.11.1 Backend 46
8.11.1 Base de datos 48

8.11.2 Frontend - Codigo Android 49
8.11.3 UI/UX 50

9. Datos de prueba 52

10. Manual de instalación 53

2. Introduccion

En este documento se buscará definir todos los elementos necesarios para llevar adelante el desarrollo del sistema del obligatorio. Para esta primera entrega, este documento contará con la especificación de los requerimiento funcionales y no funcionales y tendrán un caso de uso asociado a cada uno. A su vez, estará detallada toda la planificación de cómo se va a desarrollar este proyecto, incluyendo la metodología de trabajo, el cronograma de trabajo, planes de releases y el control de versiones.
Se presenta además dos secciones en las cuales describiremos en una como ser la gestión de los riegos y en la otra la gestión de calidad. Por último se tendrá los flujos de navegación de los prototipos de pantallas para front-end mobile.

2.1. Análisis del problema y solución propuesta.

Se busca solucionar el problema de la falta de comunicación entre los habitantes de una zona o un barrio de una ciudad. Hoy en día la gente vive acostumbrada a tener toda la información actualizada minuto a minuto en la palma de su mano, ya sean las noticias, el clima, resultados de deportes o cualquier otro tipo de información. Al analizar las apps disponibles en el mercado

actual, no logramos encontrar ninguna que brinde la posibilidad de que los habitantes de un barrio se compartan información pertinente entre los mismos, sin la necesidad de comunicarse directamente entre ellos.
Debido a este vacío en el mercado, se decidió optar por la realización de una app que brinde esta posibilidad a la gente, la posibilidad de mantenerse comunicado con sus vecinos del barrio y compartirse información que pueda ser útil y que ayude a los demás.

Así nace **TuBarrio**. **TuBarrio** es una plataforma donde se puede realizar publicaciones dirigidas a los vecinos de un barrio o una zona en particular. Dichas publicaciones pueden ser de una amplia gama de tipos, como por ejemplo avisos de perros perdidos, información acerca de calles en mal estado o calles donde sea peligroso transitar, novedades del barrio, como lo puede ser un nuevo sector para perros en una plaza por ejemplo, o cualquier otra cosa que se quiera compartir y que pueda ser de utilidad para los vecinos.

La plataforma permite a los usuarios crear publicaciones geolocalizadas, asi como tambien subir fotos junto con la publicación. Los demás usuarios las van a poder ver representadas en un mapa, y podrán realizar comentarios en las mismas y también seguir una publicación. Al seguir una publicación, el usuario podrá recibir notificaciones en caso de que la publicación sea actualizada de alguna manera, lo que es de gran utilidad en caso de querer mantenerse informado acerca de algún acontecimiento en el barrio.

3. Especificación de requerimientos

En este seccions se especificarán todos los requerimientos para el sistema a desarrollar. Entre ellos se encuentra una lista de requerimientos funcionales y una lista de requerimientos no funcionales del sistema. Los funcionales estarán representados por medio de casos de uso, acompañados de mockups cuando corresponda.

3.1. Alcance del producto

El alcance del producto a desarrollar será proveer un sistema que permita a los usuarios gestionar los eventos/sucesos que ocurran en su barrio. El usuario podrá registrar publicaciones en el mapa de su barrio así como también visualizar publicaciones publicadas por otros usuarios.

Se desarrollara una aplicación mobile donde los usuarios podrán crear publicaciones. En los mismos incluirán información relevante y podrán agregar fotos a la misma. La misma quedar publicada sobre el mapa del barrio donde allí otros usuarios podrán acceder a ella. Las fotografías de un evento se podrán exportar y guardar como un álbum en la galería de fotos.

3.2. Actores del sistema

Se han identificado dos tipos de actores con relación al sistema. Se presentan a continuación sus características.

Actor	Caracteristicas	Primario	Secundario
Usuario registrado	Es cualquier persona que accede a la aplicación y la utiliza. Para ello debe haberse registrado previamente en la aplicación.	✓	
Usuario no registrado	Es cualquier persona que accede a la aplicación sin tener previamente un usuario creado. Interactuara con la pantalla de inicio de sesión y con alguna de las formas de registrarse. Una vez se registre, pasará a ser un usuario registrado.	✓	

3.3 Requerimientos Funcionales

3.3.1. RF01: Registro de nuevo usuario con Facebook

Descripción: La aplicación deberá permitir a las personas registrarse en el sistema utilizando su cuenta de facebook.
Prioridad: Alta
Especificacion: CU 1

3.3.2. RF02: Registro de nuevo usuario sin Facebook

Descripción: La aplicación solicitará a los usuarios ingresar su nombre, apellido, numero de celular, mail, usuario y contraseña.
Prioridad: Alta
Especificacion: CU 2

3.3.3. RF03: Inicio/Cierre de sesión

Descripción: La aplicación deberá permitir a los usuarios iniciar sesión en la misma, así como cerrar la sesión iniciada previamente.
Prioridad: Alta
Especificacion: CU 3

3.3.4. RF04: Ver perfil de un usuario

Descripción: Los usuarios podrán acceder a su propio perfil para ver los datos que han ingresado y su foto de perfil.
Prioridad: Alta
Especificacion: CU 4

3.3.5. RF05: Modificar los datos del perfil de un usuario

Descripción: Los usuarios podrán acceder a su propio perfil para ver los datos que han ingresado y su foto de perfil.
Prioridad: Media
Especificacion: CU 5

3.3.6. RF06: Crear publicación

Descripción: Los usuarios podrán crear publicaciones ingresando su nombre, descripción, categoría y su localización.
Prioridad: Alta
Especificacion: CU 6

3.3.7. RF07: Ver mapa de publicaciones

Descripción: Los usuarios podrán ver un mapa del barrio en el cual se mostrarán todas las publicaciones suyas y de los demás usuarios.
Prioridad: Alta
Especificacion: CU 7

3.3.8. RF08: Ver publicación

Descripción: Los usuarios podrán seleccionar la publicación del mapa y se mostrará toda la publicación en detalle.
Prioridad: Alta
Especificacion: CU 8

3.3.9. RF09: Modificar publicación

Descripción: Los usuarios podrán editar sus propias publicaciones posteadas.
Prioridad: Media
Especificacion: CU 9

3.3.10. RF10: Borrar publicación

Descripción: Los usuarios podrán borrar sus propias publicaciones posteadas.
Prioridad: Media
Especificacion: CU 10

3.3.11. RF11: Seguir publicación

Descripción: Los usuarios podrán seguir las publicaciones que más le interesen.
Prioridad: Media
Especificacion: CU 11

3.3.12. RF12: Comentar publicación

Descripción: Los usuarios podrán hacer comentarios sobre cualquier publicación posteada
Prioridad: Media
Especificacion: CU 12

3.3.13. RF13: Notificaciones

Descripción: Los usuarios recibirán notificaciones de aquellas publicaciones las cuales hayan seguido cuando estas cambien su estado.
Prioridad: Media
Especificacion: CU 13

3.3.14. RF14: Exportar fotos de la publicación

Descripción: Los usuarios podrán descargar a su celular las fotos que contengan las publicaciones.
Prioridad: Media
Especificacion: CU 14

3.4. Requerimientos no funcionales:

3.4.1. Restricciones generales

RNF01: Se respetará el estándar de Clean Code para realizar la codificación de la aplicación.
RNF02: Se realizará el control de versiones a través de GitHub.
RNF03: La aplicación debe poder correr en un dispositivo Android.
RNF04: Se respetarán las heurísticas de Nielsen para realizar la interfaz de usuario de la aplicación.

3.4.2. Restricciones de diseño

RNF05: Se utilizará SQL Server para la persistencia de datos.
RNF06: Se utilizará la herramienta Entity Framework para el manejo de los datos.
RNF07: Se utilizará la plataforma de .NET, ASP Web Api para el desarrollo del Back-End.
RNF08: La aplicación contará con un front-end nativo de Android.
RNF09: El front-end se desarrollará utilizando Android Studio.

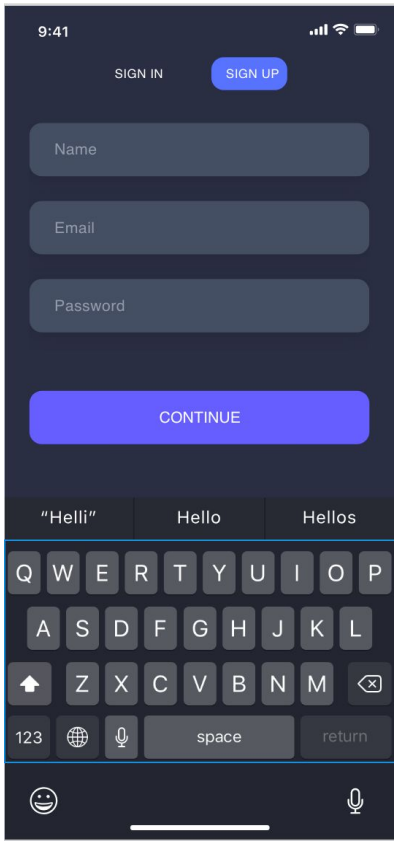
3.4 Casos de uso:

3.4.1. Registro de nuevo usuario con Facebook

Nombre: Registro de nuevo usuario con Facebook
Actor: Usuario
Precondiciones: Tener una cuenta de Facebook
Curso Básico:

1. En caso de no haber sesión, el usuario selecciona la opción de “Iniciar sesión con Facebook”.
2. Se abre modulo de aplicacion de Facebook, donde se ingresan los datos de Facebook
3. Se verifica si ya existe el usuario en nuestro servidor, y se crea usuario nuevo en caso que no exista.
4. El sistema pide al usuario permiso para enviarle notificaciones.

Curso Alternativo
2.1 El usuario no desea ingresar datos y cancela/cierra la aplicación.
3.1 Ya existe usuario de facebook, se inicia sesión pero no se crea nuevo usuario.
3.2 No existe usuario de facebook, se despliega mensaje de error.
Pantalla asociada:

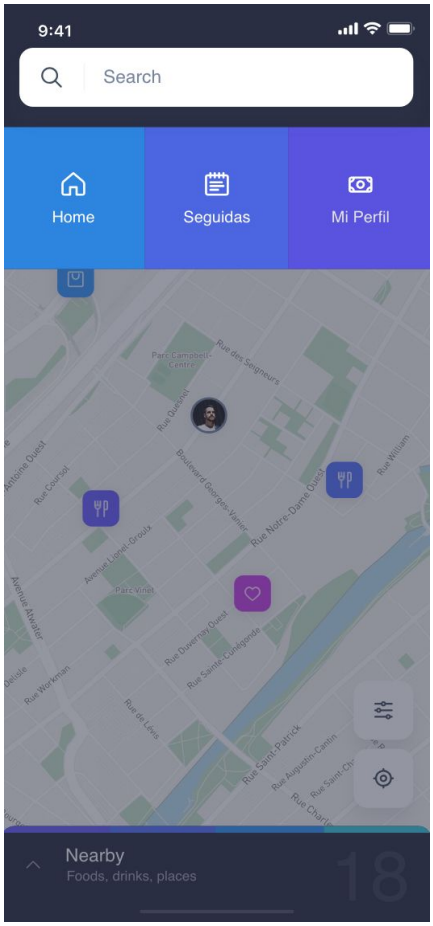


3.4.2 Ver Perfil de Usuario

Nombre: Ver perfil de un usuario
Actor: Usuario registrado
Precondiciones: Usuario iniciado sesión en la aplicación
Curso básico:

1. El usuario selecciona la opción “Perfil” del menú hamburguesa lateral.
2. El sistema muestra la información asociada al perfil del usuario que esté iniciado sesión.

Curso alternativo:
No existe.
Pantalla asociada:

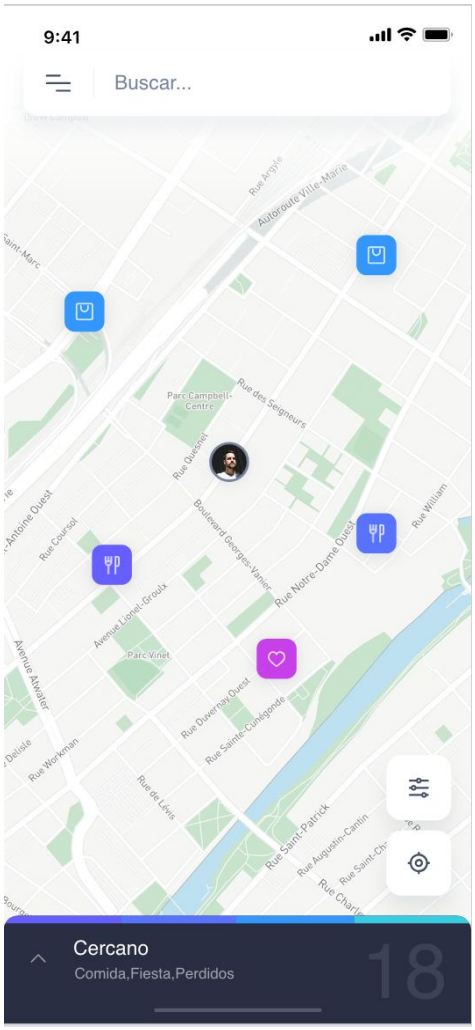


3.4.3 Ver Mapa de Publicaciones

Nombre: Ver mapa de publicaciones
Actor: Usuario registrado
Precondiciones: Usuario iniciado sesión en la aplicación
Curso básico:

3. El usuario selecciona la opción “Mapa” del menú hamburguesa lateral.
4. El sistema muestra mediante iconos, todas las publicaciones que se encuentran dentro del rango del mapa seleccionado.

Curso alternativo:
No existe.
Pantalla asociada:



3.3.4. Modificar Perfil de Usuario

Nombre: Modificar datos de perfil de Usuario
Actor: Usuario registrado
Curso Básico:

1. El usuario selecciona la opción “Perfil” del menú hamburguesa lateral.
2. El sistema muestral perfil de usuario. Todos los campos aparecen editables.
3. El usuario modifica los campos y selecciona “Guardar”. Si agrega una foto de perfil y es la primera vez que se usa la cámara/galería desde la aplicación, el sistema pedirá permisos para utilizarla.
4. El sistema autentica los datos y guarda los cambios.

Curso Alternativo:
4.1 Nombre y/o Apellido vacíos. Sistema muestra mensaje de error.

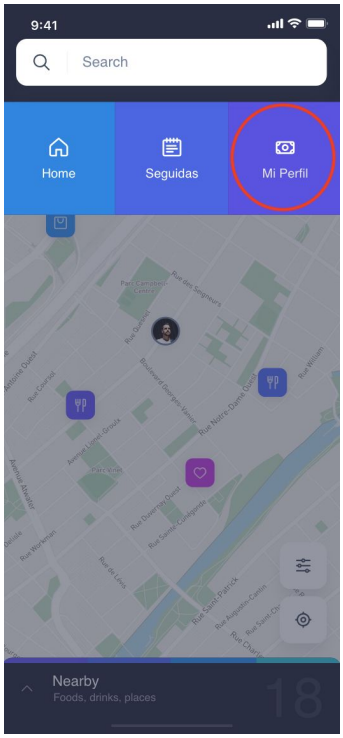
3.4.5. Crear publicación

Nombre: Crear publicación.

Actor: Usuario.
Precondición: Usuario logueado en la aplicación.
Curso básico:

1. El usuario presiona el botón de Crear Publicación en el mapa.
2. El sistema muestra la pantalla de Crear Publicación.
3. El usuario ingresa los datos de la publicación (nombre, descripción, ubicación, fotos).
4. El usuario selecciona la categoría de la publicación.
5. El usuario presiona el botón de Confirmar Publicación.
6. El sistema valida los datos de la publicación.
7. Se crea el evento y se puede visualizar en el mapa.

Curso alternativo:
5.1 El usuario no ingresa algún dato y presiona Confirmar Publicación. El sistema le mostrará un mensaje indicando los datos que debe llenar.
5.2 El usuario ingresa algún dato inválido y presiona Confirmar Publicación. El sistema le mostrará un mensaje indicando los errores que debe corregir.
5.3 El usuario no selecciona una categoría y presiona Confirmar Publicación. El sistema le mostrará un mensaje indicando que debe seleccionar una categoría.
Pantalla asociada:

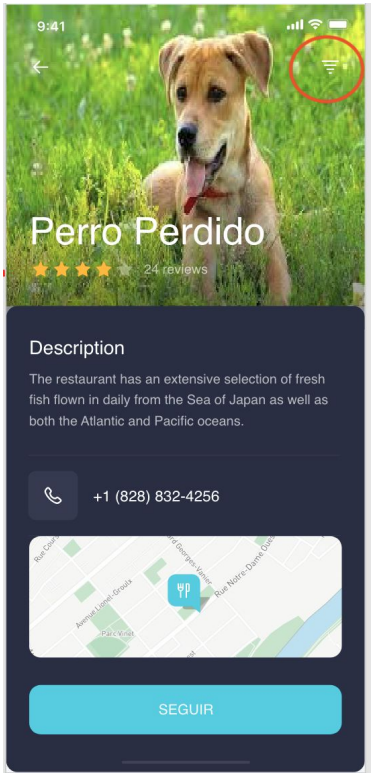


3.4.6. Modificar publicación

Nombre: Modificar publicación.
Actor: Usuario.
Precondición: Usuario logueado en la aplicación. Usuario creador de la publicación
Curso básico:

1. El usuario selecciona la opción Mis Publicaciones.
2. El usuario selecciona una de las publicaciones.
3. El sistema muestra la pantalla de la publicación.
4. El usuario presiona el botón Modificar Publicación.
5. El usuario ingresa los nuevos datos de la Publicación

Curso alternativo:
5.1 El usuario no ingresa algún dato y presiona Confirmar Publicación. El sistema le mostrará un mensaje indicando los datos que debe llenar.
5.2 El usuario ingresa algún dato inválido y presiona Confirmar Publicación. El sistema le mostrará un mensaje indicando los errores que debe corregir.
5.3 El usuario no selecciona una categoría y presiona Confirmar Publicación. El sistema le mostrará un mensaje indicando que debe seleccionar una categoría.
Pantalla asociada:

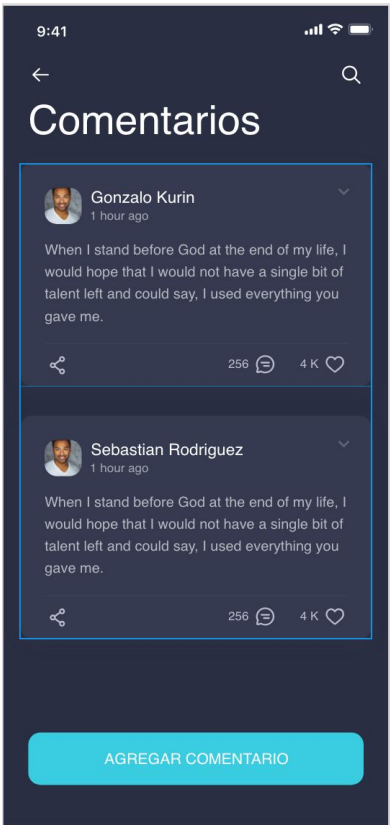


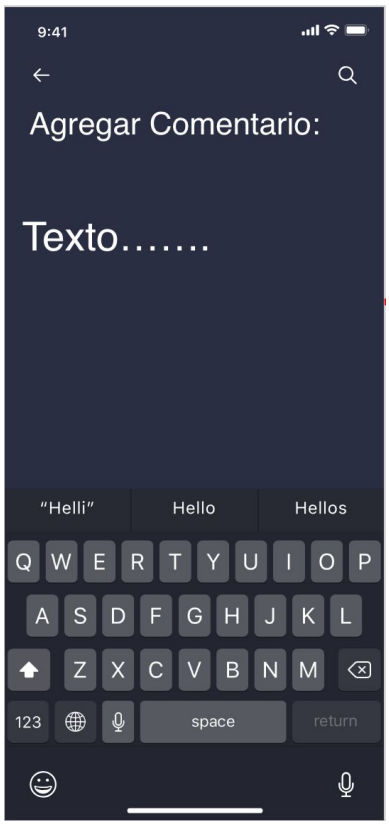
3.4.7. Comentar Publicación

Nombre: Comentar Publicación.
Actor: Usuario.
Precondición: Usuario logueado en la aplicación.
Curso básico:

1. Usuario presiona en una publicación en el mapa.
2. El sistema muestra la pantalla de la publicación.
3. El usuario presiona el botón Comentar Publicación.
4. El sistema muestra la pantalla de Comentar Publicación.
5. El usuario escribe el comentario.
6. El usuario presiona el botón Comentar.
7. Se agrega el comentario a la publicación.
8. Se vuelve a la pantalla de la publicación.

Curso alternativo:
6.1 El usuario no escribe ningún comentario, y presiona el botón Comentar. El sistema le mostrará un mensaje indicando que el comentario no puede estar vacío.
Pantalla asociada:



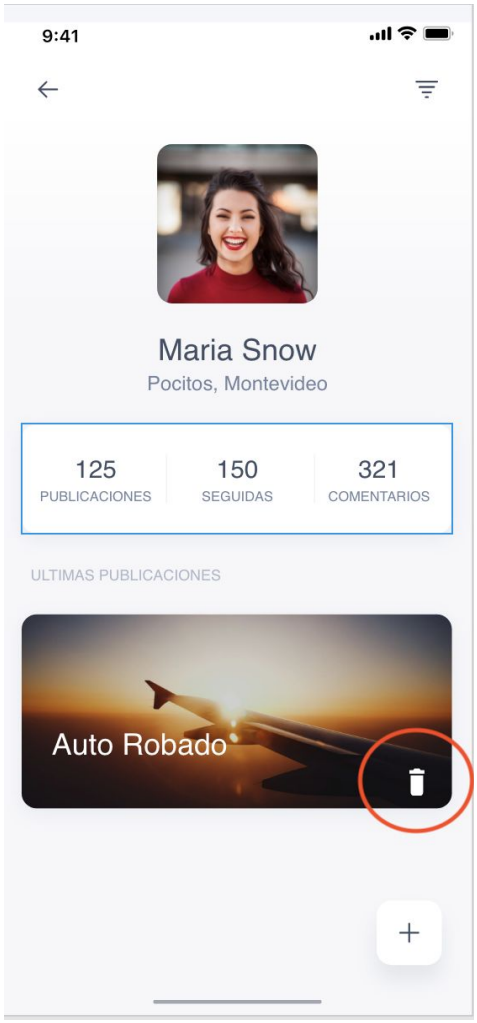


3.4.8. Borrar Publicación

Nombre: Borrar Publicación.
Actor: Usuario.
Precondición: Usuario logueado en la aplicación.
Curso básico:

1. El usuario entra a la ventana de “Mis publicaciones”.
2. El usuario selecciona una de sus publicaciones.
3. El sistema le muestra una pantalla con su publicación.
4. El usuario presiona el botón Eliminar.

Curso alternativo:
7.1 El usuario accede a su publicación a través del mapa del barrio.
7.2 El usuario no selecciona una de sus publicaciones. El sistema le indicará que solo puede eliminar una de sus publicaciones.
Pantalla asociada:

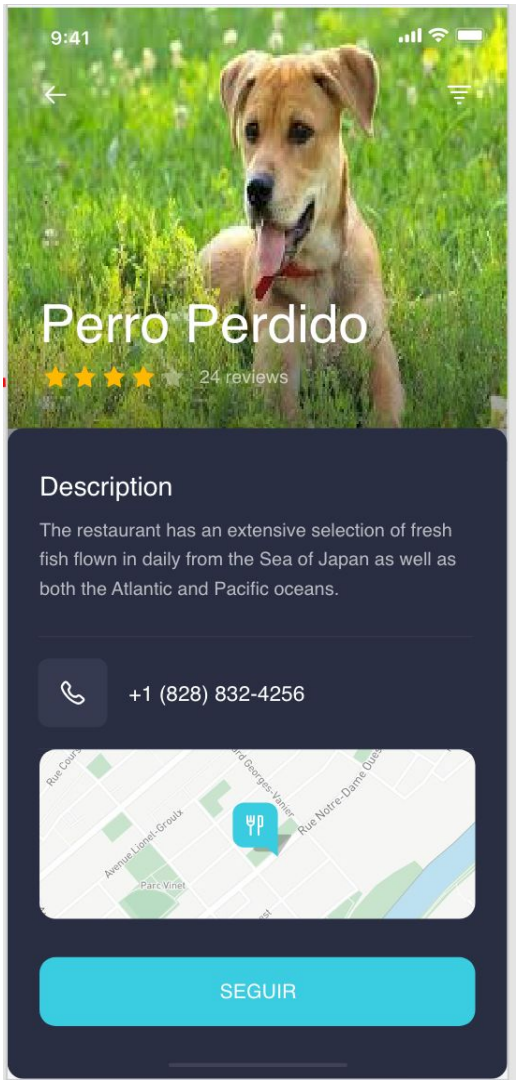


3.4.9. Ver Publicación

Nombre: Ver Publicación.
Actor: Usuario.
Precondición: Usuario logueado en la aplicación.
Curso básico:

1. El usuario presiona en una publicación en el mapa.
2. El sistema muestra la pantalla de la publicación.

Pantalla asociada:

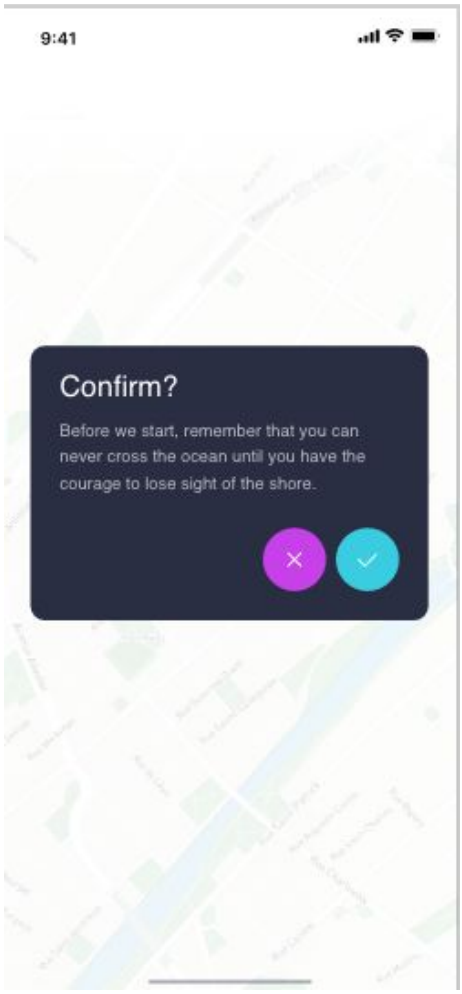


3.4.10. Notificaciones

Nombre: Notificaciones.
Actor: Sistema
Precondicion: Usuario siguiendo alguna publicación.
Curso basico:

1.El usuario recibirá un popup en el cual se le notificara que una de las publicaciones que sigue ha cambiado de estado.
2. El usuario oprime el botón de Ok.

Pantalla asociada:



3.4.11. Exportar fotos de publicaciones

Nombre: Exportar fotos
Actor: Usuario
Precondicion: Usuario logueado en la aplicación.
usuario logueado en la publicación.
Curso basico:

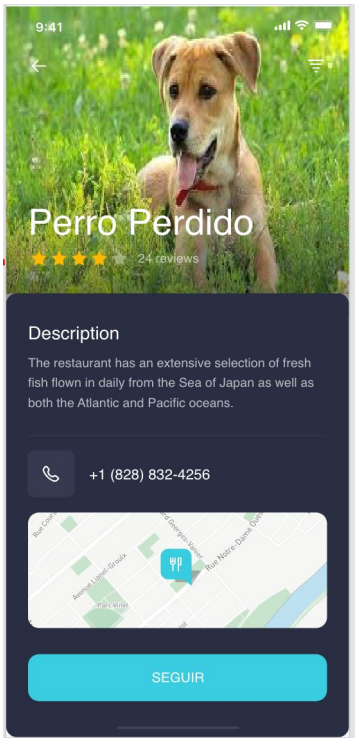
1. El usuario presiona en una publicación del mapa.
2. El sistema muestra la pantalla de la publicación.
3. El usuario selecciona una foto de la publicación.
4. El usuario oprime el botón Descargar Foto.

3.4.12. Seguir publicación

Nombre: Seguir publicación
Actor: Usuario
Precondicion: Usuario logueado en la aplicación.
Curso basico:

1. El usuario presiona en una publicación del mapa.
2. El sistema muestra la pantalla de la publicación.
3. El usuario oprime el botón de Seguir.

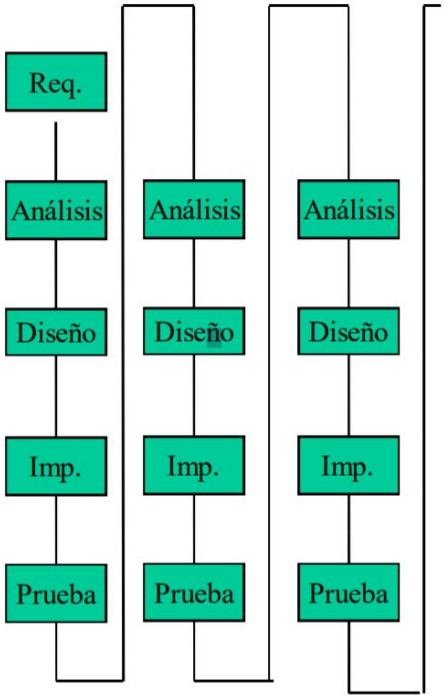
Curso Alternativo:
12.3. El usuario ya sigue esa publicación. Si oprime de nuevo dejará de seguir la publicación.
Pantalla asociada:



4. Planificación

4.1. Metodología de trabajo

Al poder definir los distintos requerimientos de manera clara y debido a que creemos que los mismos no van a cambiar casi durante el desarrollo del sistema, nuestra elección de ciclo de vida será un ciclo de vida incremental iterativo. De esta manera al principio se definirán los distintos requerimientos y el diseño del sistema, y una vez realizado esto, se pasará a ejecutar las distintas iteraciones como se muestra a continuación:



Al tener este tipo de planificación, se pueden realizar distintas versiones que permitan ir mostrando el sistema de manera incremental. Es decir, luego de cada iteración, se crea una versión que contiene las mismas funcionalidades que la versión anterior, pero se agregan aún más. Esto permite que el cliente pueda ir viendo durante todo el proceso, como se va mostrando el sistema. Estar tan pegado al cliente permite que el mismo pueda ir viendo cada detalle e ir ajustando los mismos, hasta que le parezca el adecuado. Principalmente lo que se evita es que el sistema no tenga que ser modificado radicalmente una vez que gran parte del mismo esté desarrollado, sino que a partir de cada iteración se corrige y ajusta lo suficiente para la próxima.

4.2. Cronograma de trabajo

Como mencionamos anteriormente, el desarrollo del proyecto se dividirá en módulos de funcionalidad que se irán desarrollando a lo largo de cada interacción. A continuación se muestra el cronograma definida:

N. Tarea	Tarea	Horas	Precondicion
ITERACIÓN 0 (11/03- 23/04)			
Tarea 1	Definición de metodología de trabajo	1	-
Tarea 2	Especificación de requerimientos	8	Tarea 1 finalizada
Tarea 3	Planificación y documentación inicial	15	Tarea 1 finalizada
Tarea 4	Realización de Mock-Ups de front-end	6	Tarea 2 finalizada
ITERACIÓN 1 (25/04 -08/05)			
Tarea 1	Desarrollo de entidades del sistema y su lógica correspondiente	25	-
Tarea 2	Persistencia en BD de las entidades desarrolladas	10	Tarea 1 empezada
Tarea 3	Desarrollo de los endpoints de la API	30	Tarea 1 finalizada Tarea 2 finalizada
ITERACIÓN 2 (10/05 - 20/05)			
Tarea 1	Desarrollo en front end de inicio de sesión, registro de usuario y cierre de sesión	15	-
Tarea 2	Desarrollo en front end de menú de navegación	3	-
Tarea 3	Desarrollo en front end de perfil de usuario	10	-
Tarea 4	Desarrollo en front end the modificación de usuario	7	Tarea 3 finalizada
ITERACIÓN 3 (22/05 - 01/06)			
Tarea 1	Desarrollo en front end de mapa de publicaciones	15	-

Tarea 2	Desarrollo en front end de crear publicación	8	-
Tarea 3	Desarrollo en front end de modificar publicación	6	Tarea 2 finalizada
Tarea 4	Desarrollo en front end de eliminar publicación	6	Tarea 2 finalizada
Tarea 5	Desarrollo en front end de comentar publicación	5	Tarea 2 finalizada
ITERACIÓN 4 (2/06 - 16/06)			
Tarea 1	Desarrollo en front end de seguir publicación	5	-
Tarea 2	Desarrollo en front end de dejar de seguir una publicación	4	Tarea 1 finalizada
Tarea 3	Desarrollo en front end de notificaciones de publicación	5	Tarea 1 finalizada
Tarea 4	Desarrollo en front end de exportar imágenes de una publicación	10	-

4.3. Plan de releases

Los releases del proyecto se harán de la siguiente manera:

Numero de release	Contenido	Fecha de Comienzo	Fechas de finalización
1	Iteracion 0	11/03/19	23/04/19
2	Iteraciones 1 a 4	25/04/19	16/06/19

4.4. Control de versiones

Se utilizará un repositorio en Bitbucket para realizar el control de versiones del proyecto. El mismo podrá ser accedido por todos los miembros del equipo y por el docente. Al utilizar esta herramienta, se mantienen organizados los avances y los cambios realizados al proyecto.

Se va utilizar Git y Git Flow, la utilización de los mismos se ve reflejada en las ramas del sistema. En la rama master se efectuarán los releases, y en la rama develop se registrarán los cambios durante el desarrollo de cada iteración.

5. Gestión de Riesgos

A continuación se realizará la gestión de riesgos. En la misma van a ser identificados y especificados los posibles riesgos que puedan surgir a lo largo del proyecto. Se va a presentar una tabla con el análisis cualitativo de cada uno de los riesgos identificados, indicando su impacto, probabilidad de ocurrencia, magnitud, plan de respuesta y plan de contingencia.

5.1. Identificación y especificación

Nombre	Condición	Consecuencias	Contexto
Errores en la estimación	El equipo no cuenta con mucha experiencia en estimaciones de proyectos	Se puede atrasar/adelantar el desarrollo del proyecto y hay que volver a realizar la estimación	Los errores en la estimación son especialmente graves ya que la fecha de entrega del proyecto es inamovible y no pueden haber retrasos
Errores en el manejo de la Web Api	El equipo ya conoce la modalidad de trabajo, pero igualmente realiza un error significante al momento de desarrollar la API para el back end	Se debe invertir tiempo no contemplado para solucionar los problemas emergentes, lo que retrasa el desarrollo del proyecto	Impacta negativamente debido a que se retrasa la realización del proyecto, y no hay tiempo de sobra para retrasos
Problemas con el manejo de Android Studio	El equipo no conoce la herramienta utilizada para desarrollar el front end	Hay que invertir tiempo en investigación y estudio de la herramienta	Presenta una desventaja ya que requiere dedicar tiempo del cronograma al estudio de la herramienta
Algún integrante del equipo abandona la materia	Por alguna razón indiferente uno de los integrantes del equipo abandona la	Los demás integrantes del equipo se ven afectados debido a	Es un impacto importante debido a que hay que reorganizar y volver a

	materia y por ende el equipo de trabajo	que deben recuperar el tiempo dedicado al integrante que abandonó	planificar el proyecto teniendo en cuenta que hay un desarrollador menos
Los entregables de cada iteración no contienen lo especificado en el cronograma	El equipo por error desarrolla funcionalidades en una iteración en la cual no fueron planificadas	Se pierde tiempo reorganizando el cronograma y acomodando las funcionalidades para poder desarrollarlas todas ordenadamente	Presenta una desventaja debido a que hay que reorganizar y se pierde tiempo de desarrollo

5.2. Análisis cualitativo

Para el análisis cualitativo se utilizarán las siguientes escalas:

Impacto	Probabilidad de ocurrencia
0 - Ninguno	0.0 - No probable
1 - Marginal	0.2 - Poco probable
2 - Poco importante	0.4 - Probable
3 - Importante	0.6 - Muy probable
4 - Crítico	0.8 - Altamente probable
5 - Catastrófico	1 - Se convierte en problema

Riesgo	Impacto	Probabilidad	Magnitud	Plan de Respuesta	Plan de Contingencia
Problemas con el manejo de Android Studio	3	0.8	2.4	Utilizar las guías provistas por el docente. Leer documentación o realizar tutoriales en internet	Solicitar ayuda al docente

Errores en la estimación	3	0.4	1.2	-	Volver a estimar
Algún integrante del equipo abandona la materia	4	0.2	0.8	-	Consultar con el docente como proseguir. Seguramente se repiense el alcance del sistema para poder llegar a entregar
Errores en el manejo de la Web Api	2	0.2	0.4	El equipo ya conoce la modalidad de trabajo por obligatorios anteriores	Solicitar ayuda al docente o a algún compañero
Los entregables de cada iteración no contienen lo especificado en el cronograma	2	0.2	0.4	Repasar el cronograma previamente a iniciar una iteración para tener claras las funcionalidades a desarrollar	Corregir el cronograma y redistribuir las actividades en las iteraciones

6. Gestión de la Calidad

Para gestionar la calidad del sistema durante el desarrollo se llevarán a cabo distintas prácticas que se describirán a continuación.

6.1. Actividades preventivas

- Utilización de estándares y buenas prácticas de codificación. Al utilizar estándares y prácticas conocidas de codificación se asegura que se va a obtener un código prolijo y conforme a formatos universales. De esta manera se asegura que si el sistema debe ser mantenido por otros desarrolladores en un futuro, no sería tan difícil debido a que el código es prolijo y cumple con estándares.
- Revisiones. Se realizarán revisiones de a pares para detectar errores y evitar que los mismos se transformen en problemas.

6.2. Actividades correctivas

Las actividades correctivas se realizan para detectar disconformidades con el software, y para corregir la causa del problema. Para nuestro sistema, se implementarán pruebas unitarias que testeen todas las clases del dominio. Además, se llevarán a cabo pruebas funcionales, las cuales se verificarán y validarán el funcionamiento del back-end. También se realizarán pruebas de integración con el sistema front-end, así como pruebas de usabilidad.

6.3. Gestión de cambios

Al elegir un ciclo de vida iterativo, no se espera que los requerimientos del proyecto cambien de manera impactante. Aún así, se definirá un proceso que se debe cumplir para realizar un cambio en el sistema.

1. Identificar el cambio a ser realizado
2. Emitir una una solicitud de cambio
3. Priorizar y categorizar el cambio
4. Analizar si se realizará o no (a partir de tiempo, costo e importancia)
5. Implementación del cambio
6. Probar el sistema para verificar la correcta implementación del cambio
7. Almacenar la nueva versión. Esto se llevará a cabo con el control de versiones.

6.4. Gestión de defectos

Al igual que para los cambios, se define un proceso para corregir los defectos que surjan al trabajar en el proyecto.

1. Identificar el defecto
2. Documentar detalladamente el defecto
3. Analizar la viabilidad de solución del defecto. En caso de decidir por solucionarlo, se llevarán a cabo los pasos del 4 al 7 del proceso de “Gestión de Cambios”. En caso contrario, se documentará y justificará adecuadamente

Diagrama de clases

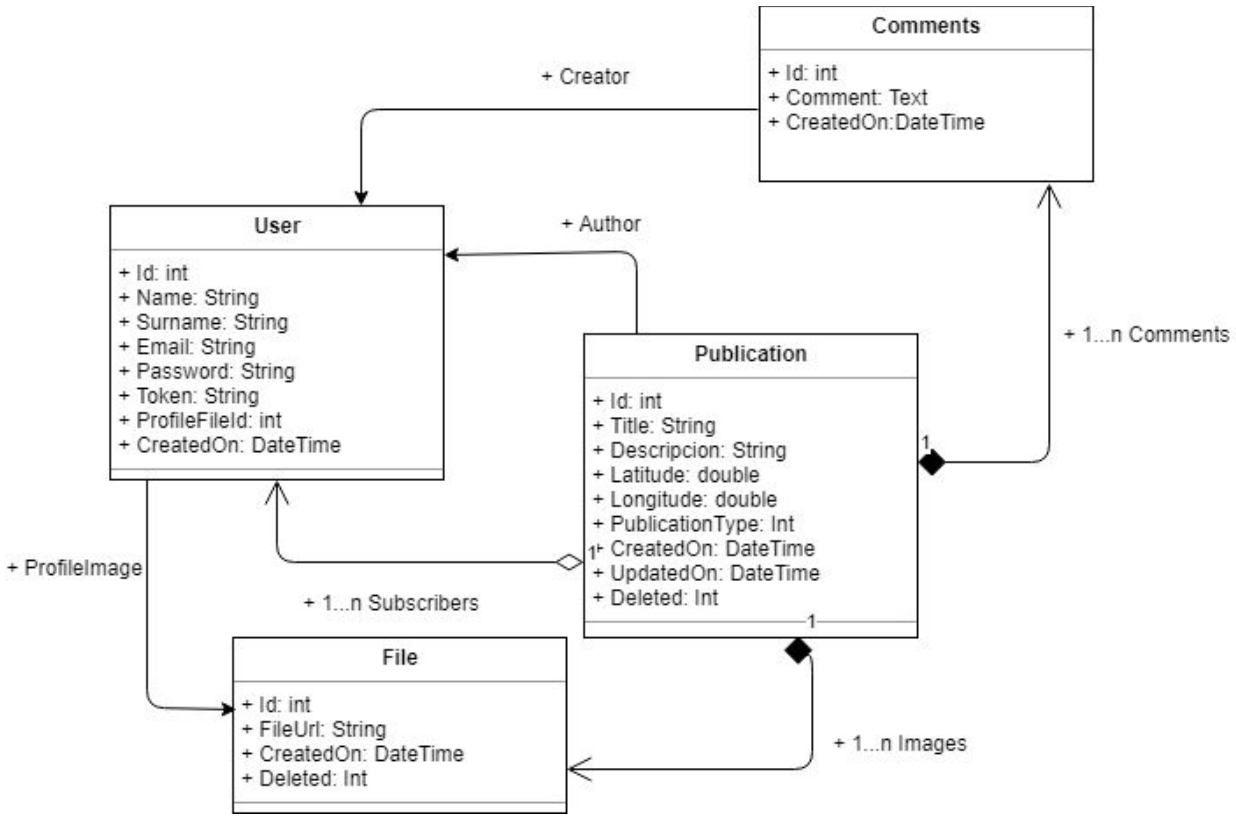
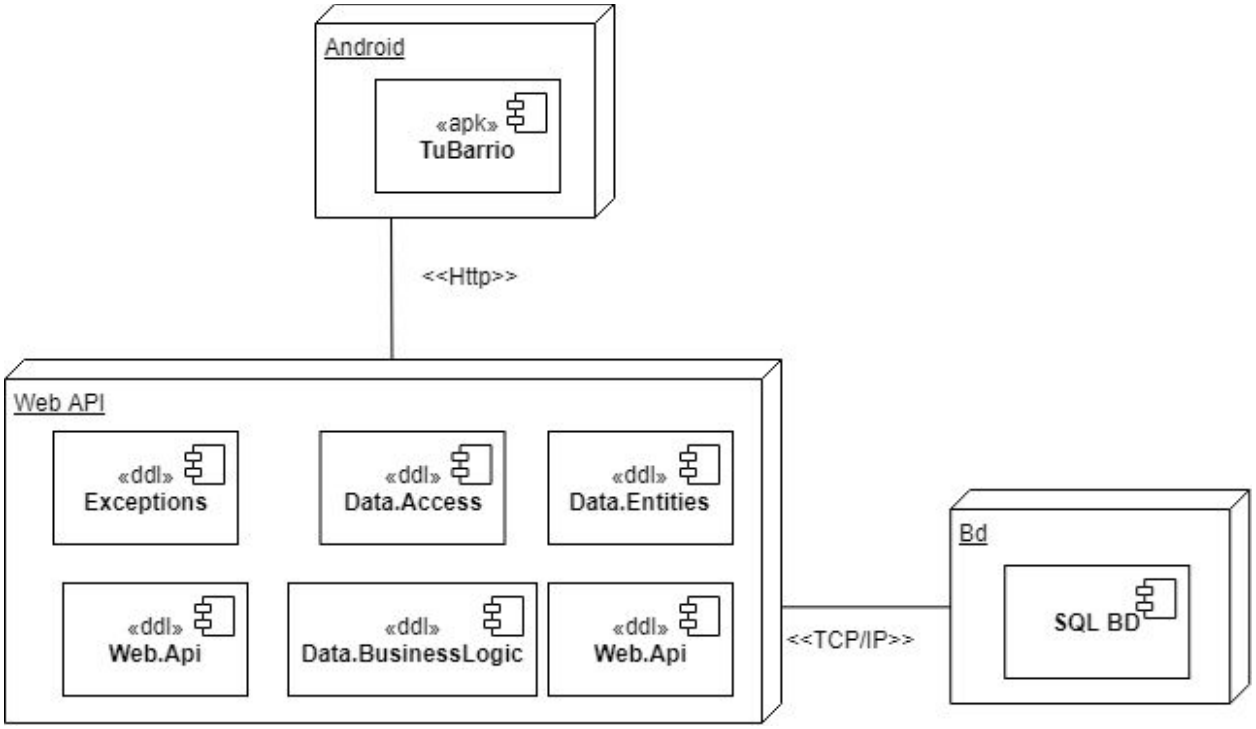
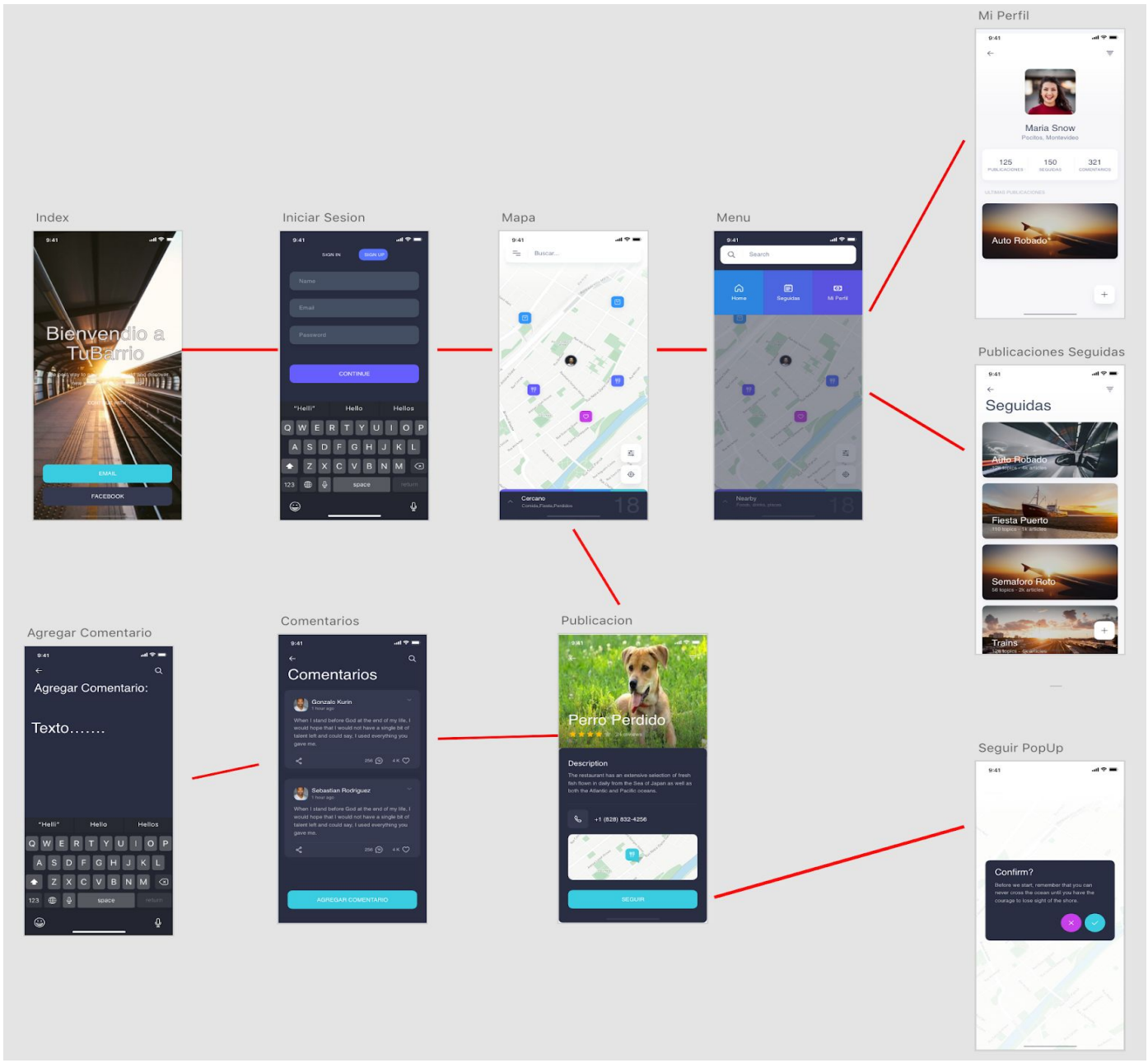


Diagrama de despliegue



7. Mockups Moviles

7.1. Flujo de navegación



8. Evaluación del proceso de desarrollo

8.1 Introducción

Esta sección del trabajo tiene como fin explicar la realización del proyecto. Si bien al comienzo del mismo se realizó una etapa de planificación, en la cual se definió como iba a ser el trabajo y que prácticas de ingeniería se utilizarían, en esta sección se va a comprobar si el trabajo fue realizado de esa manera o no.

Se va a analizar si fueron implementados todos los requerimientos. En caso de que alguno no haya sido terminado, o se haya modificado por algún motivo, se va a justificar la decisión tomada. Además se va a hacer énfasis en mostrar el cumplimiento de las tareas definidas en el plan de proyecto. Se prestará especial atención al cumplimiento de los plazos pautados en el cronograma. También se van a analizar si los riesgos planteados se convirtieron en amenazas, y si la gestión de cambios y defectos fue llevada a cabo como fue definida. Se van a dar evidencias de la gestión de calidad, y se incluirán diagramas actualizados del diseño de la aplicación, indicando, en caso de existir, modificaciones de los mismos.

8.2 Alcance

8.2.1 Requerimientos funcionales

No hubo cambios de alcance entre la primera y segunda entrega. Esto significa que no se realizaron cambios en cuanto a requerimientos funcionales en el proyecto.

Por momentos a lo largo del desarrollo del proyecto, se observó que habían situaciones en las cuales las iteraciones se finalizaban sin haber logrado cumplir todas las tareas planificadas para la iteración, lo que resultaba en una necesidad de reestructurar el cronograma. Esto llevó a la conclusión de que se iba a tener que realizar una priorización de los requerimientos, de esta manera se lograría identificar qué funcionalidades serían claves para avanzar en el proyecto y cuáles se podían postergar para iteraciones posteriores. Dicha priorización se realizó de acuerdo a qué funcionalidades eran de mayor relevancia para el sistema final. Se realizó de manera individual y luego se realizó una puesta en común para decidir entre todos los integrantes.

Al llegar a la fecha de entrega, se logró desarrollar todas las funcionalidades del sistema, a excepción de la funcionalidad del envío de notificaciones desde el server a los usuarios. Este requerimiento se dejó para el final, y por cuestiones de tiempo no se logró implementar.

8.2.3 Requerimientos no funcionales

Por otro lado, en cuanto a requerimientos no funcionales, muchos de ellos se pueden evidenciar en el proyecto (uso de base de datos SQL, uso de Entity Framework, Web Api y Android, entre otros). Sin embargo, hay algunos RNF cuyo cumplimiento se demostrará a continuación.

- Estandares de Clean Code

Se buscó respetar a lo largo de todo el sistema los estándares de Clean Code. Dichos estándares permiten obtener un código prolijo, fácil de entender y de mantener. Este requerimiento es importante debido a que en la letra del trabajo se especifica que el código va a recibir cambios y modificaciones en un futuro, y el Clean Code ayuda en su mantenibilidad. Además, se utilizaron prácticas de Clean Code en la codificación de pruebas unitarias y mocks de manera de simplificar el entendimiento de estas a cualquier persona que haga el futuro mantenimiento y testing. Se incluirán capturas de pantalla a continuación para ejemplificar:

Backend:

```
9 referencias | 0 cambios | 0 autores, 0 cambios | 0 excepciones
public Publication GetPublicationById(int id, string token)
{
    User author = authenticationLogic.GetUserWithToken(token);
    Publication publicationToGet = publicationRepository.GetPublicationById(id);
    if (author.Equals(publicationToGet.Author))
    {
        return publicationToGet;
    }
    else
    {
        throw new PublicationException("No tiene permiso para llamar a la publicacion");
    }
}
```

El código anterior corresponde a la clase Publication Logic en el backend de la aplicación. Aquí se pueden observar buenas prácticas de codificación como lo son:

- Nombres de clases, variables y métodos significativos ▪ Variables locales privadas
- Métodos con cantidad reducida de parámetros
- Manejo de excepciones con mensajes significativos
- Validaciones
- Uso de interfaces

Pruebas unitarias:

```
[TestMethod]

| 0 referencias | 0 cambios | 0 autores, 0 cambios | 0 excepciones
public void GetUserWithTokenCorrectTest()
{
    string tokenRecieved = authenticationLogic.LogIn(user.Email);
    User userRecieved = authenticationLogic.GetUserWithToken(tokenRecieved);
    Assert.AreEqual(user, userRecieved);
}

[TestMethod]
[ExpectedException(typeof(InvalidCredentialException))]
| 0 referencias | 0 cambios | 0 autores, 0 cambios | 0 excepciones
public void GetUserWithTokenIncorrectTest()
{
    User userRecieved = authenticationLogic.GetUserWithToken("invalid token");
}
```

En cuanto al código de las pruebas unitarias, también se intentó manejar un código limpio que fuera fácil de comprender. Algunas buenas prácticas que se pueden encontrar en el código anterior son:

- Uso de variables privadas
- Nombres significativos en métodos y variables
- Verificación de métodos que devuelven excepciones
- Utilización de un método de setup que define datos comunes a las pruebas
- Métodos de prueba sencillos con pocas líneas

Front end:

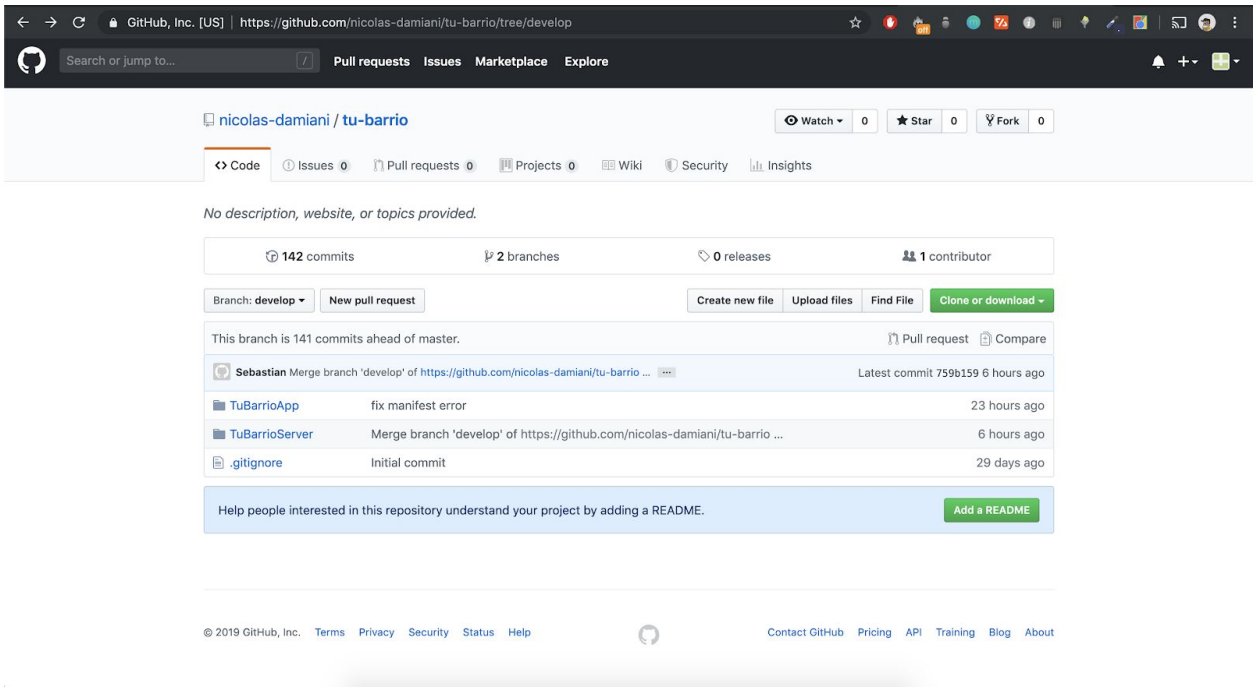
```
public class Publication implements Serializable{

    private int id;
    private String title;
    private String description;
    private Double longitude;
    private Double latitude;
    private Date updatedOn;
    private User creator;
    private ArrayList<User> followers;

    public User getCreator() { return creator; }

    public void setCreator(User creator) { this.creator = creator; }

    private String publicationImage;

    public Publication(JSONObject jsonObj) {
        try {
            //TODO
            this.id = jsonObj.has( name: "Id")?jsonObj.getInt( name: "Id"):0;
            this.title = jsonObj.has( name: "Title")?jsonObj.getString( name: "Title"):"";
            this.description = jsonObj.has( name: "Description")?jsonObj.getString( name: "Description"):"";
            this.longitude= jsonObj.has( name: "Longitude")?jsonObj.getDouble( name: "Longitude"):0;
            this.latitude= jsonObj.has( name: "Latitude")?jsonObj.getDouble( name: "Latitude"):0;
            this.publicationImage= jsonObj.has( name: "PublicationImage") && !jsonObj.getString( name: "PublicationImage").equals("null")?jsonObj.getString( name: "PublicationImage"):"";
            this.creator = new User(new JSONObject(jsonObj.getString( name: "Creator")));
            this.followers = new ArrayList<>();
            JSONArray jsonArray = new JSONArray(jsonObj.getString( name: "Followers"));

            if (jsonArray.length() != 0) {
                ArrayList<User> userList = new ArrayList<>();
                for (int i = 0; i < jsonArray.length(); i++) {
                    JSONObject userObj = jsonArray.getJSONObject(i);
                    User user = new User(userObj);
                    userList.add(user);
                }
```

Al igual que en el backend, en la parte de código de Android también se procuró utilizar estándares de Clean Code. Aquí se observa:

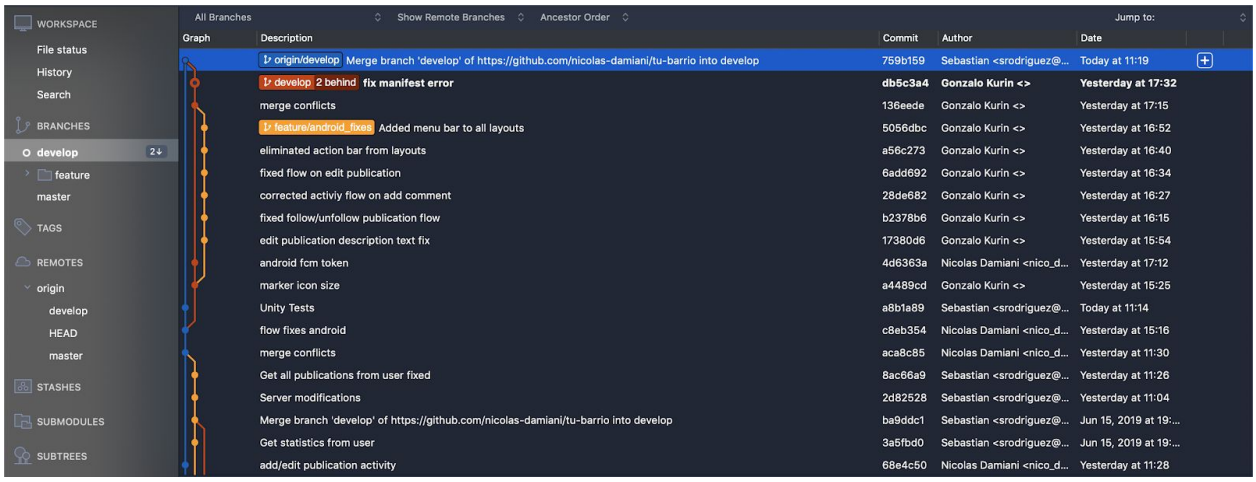
- Variables privadas
- Nombres significativos de clase, variables y métodos
- Manejo de excepciones con mensajes claros que indiquen el error
- Métodos con poca cantidad de parámetros
- Métodos cortos

- Control de versiones

El control de versiones se llevó a cabo en un repositorio en GitHub, tal como fue especificado en la letra del trabajo. El mismo puede ser accedido por el docente, y en él se pueden ver los commits de los distintos integrantes del equipo.



Para manejar los commits y el repositorio se utilizó la herramienta Git y el flujo de trabajo Git Flow. En la rama master y develop se pueden ver los releases y el desarrollo del proyecto respectivamente. Luego, cada integrante del equipo utilizó diferentes ramas a nivel local al momento de desarrollar nueva funcionalidad o de arreglar algún defecto. La siguiente captura de Git muestra el trabajo en simultáneo:



- Heurísticas de Nielsen

Un requerimiento no funcional indicaba que la interfaz de usuario debía respetar las heurísticas de Nielsen al menos a un 70%. Nuevamente, al igual que con la priorización de requerimientos, cada integrante del equipo evaluó la interfaz de usuario de la aplicación contra las heurísticas y luego se realizó un promedio obteniendo un resultado final.

Uno de los requerimientos no funcionales del proyecto indicaba que la interfaz de usuario debería respetar las heurísticas de Nielsen en al menos un 70%. Al igual que la metodología que se utilizó para priorizar los requerimientos, cada integrante evaluó la interfaz de usuario de la aplicación teniendo en cuenta las heurísticas, y se realizó un promedio de los valores.

El porcentaje se asigna de acuerdo a la siguiente tabla:

Porcentaje	Significado
0%	No se cumple con la heurística
50%	Se cumple en alguna medida con la heurística
100%	Se cumple con la heurística

Heurística	Porcentaje de cumplimiento	Justificación
Visibilidad	100%	Se muestran tanto mensajes de error como de verificación al usuario al realizar acciones
Relación con la realidad	100%	Se utiliza un lenguaje actualizado y la aplicación es completamente relacionable con la realidad, se adecua a un problema real
Control y libertad	100%	Los usuarios pueden modificar y rehacer los datos ingresados en la aplicación
Consistencia y estándares	100%	Se mantiene el lenguaje utilizado y colores uniformes a lo largo de toda la aplicación
Prevención de errores	50%	Se presentan avisos antes de realizar algunas acciones, pero hay acciones que se realizan sin confirmación (ejemplo seguir/dejar de seguir una publicación)
Reconocimiento	100%	No se fuerza al usuario a memorizar ningún

		tipo de dato para utilizar la aplicación
Flexibilidad	50%	Se guarda el usuario en sesión para que el mismo no tenga que realizar login cada vez que use la app
Estética y minimalismo	100%	Solamente se muestran por pantalla elementos e información que sean de utilidad para el usuario en el momento
Recuperarse de errores	100%	Los mensajes de error permiten a los usuarios recuperarse de errores al momento de ingresar datos (ejemplo un campo vacío)
Ayuda y documentación	0%	No se incluye ayuda en la app

Realizando un cálculo promedio de la suma de los valores de cada heurística, se obtiene aproximadamente un 80% de cumplimiento de las mismas. Este valor supera el valor de 70% que se puso como objetivo en los requerimientos no funcionales, pero igualmente es algo en lo que se podría mejorar para futuras versiones. Por ejemplo, incluyendo una sección de ayuda para la app, y realizando una mejor prevención de errores.

8.3 Cumplimiento de cronograma

Durante la primer entrega del obligatorio, el equipo realizó un cronograma en el cual se indico cómo serían realizadas las tareas y en qué fecha debería completarse. También se estimó el tiempo que se necesitaría para poder terminar cada tarea.

Al principio hubo un leve retraso en las primeras iteraciones dado que se demoró más de lo planeado llevar a cabo algunas tareas. Sin embargo, las tareas siguientes nos llevaron menos tiempo del estimado lo que resultó favorable para ponernos al dia nuevamente

Como conclusión se puede decir que cumplimos con el cronograma en tiempo y forma. A pesar de algún retraso en alguna tarea puntual, el hecho de que otras tareas nos hayan llevado menos tiempo del estimado significó que no hubo un atraso en el cronograma que haya dificultado la completitud del proyecto

8.4 Cumplimiento del plan de release

En el plan de release podemos ver reflejado lo antes dicho acerca del cronograma, ya que cumplimos con las dos fechas pactadas para el lanzamiento de cada release. Nos planteamos en el plan dos fechas para los releases, siendo cada una de ellas una de las entregas del obligatorio. Para el primer release cumplimos con la fecha pactada la cual fue el 23 de junio. Para este segundo, nos habíamos planteado terminarlo para el 16/06 pero tuvimos un retraso de un dia. Esta misma se vio retrasada en comparación a lo indicado en el plan dado que aún nos quedan detalles minimos por solucionar.

La fecha del release final se verá en el repositorio una vez entregado y realizadas las modificaciones restantes

8.5 Evaluación de riesgos

Un aspecto importante de la etapa de planificación del trabajo es la planificacion y gestion de riesgos. Es de vital importancia tenerlos en cuenta ya que durante el desarrollo del proyecto pueden surgir imprevistos que podrían afectar el resultado final del trabajo e incluso provocar la cancelación del mismo.

A continuación, se muestra un análisis de los riesgos que se definieron durante la etapa de planificación, explicando claramente cuales se convirtieron en problemas reales y cuales no, justificando por qué, y cómo se manejaron.

Riesgo	Evolución y solución
Errores en la estimación	Si bien se habían realizado algunos trabajos de estimación previos en otras materias, siempre consideramos la estimación como un riesgo ya que no tenemos mucha experiencia haciéndolo. Sucedio que alguna tareas llevaron mas tiempo del pensado y otras menos. Debido a este balance entre las tareas, pudimos cumplir con el cronograma en tiempo y forma
Errores en el manejo de la Web Api	Este riesgo no se convierte en problema ya que la experiencia previa que se tenía de trabajar con esta tecnología favorece el correcto desarrollo de la parte de Web Api de la aplicación
Problemas con el manejo de Android Studio	No hubo grandes problemas al momento de manejar Android Studio. Si sucede que al ser una herramienta nueva las primeras tareas llevaron más tiempo de desarrollo dado que había que ir aprendiendo a la par que se iban realizando las funcionalidades de la aplicación.
Algún integrante del equipo abandona la materia	No sucedio. Todos los integrantes continúan en la materia y en el equipo.
Los entregables de cada iteración no contienen lo especificado en el cronograma	Sucedió en alguna iteración que los entregables esperados no contenían todas las tareas pautadas. Lo que se decidió hacer fue transferir esas tareas no realizadas en una iteración a la iteración siguiente o a alguna posterio.

Un riesgo que no se tuvo en cuenta en la etapa inicial de documentación y que sucedió, fue el hecho que los integrantes del equipo estén cursando además otras materias y por ende tiene que cumplir con entregas e instancias de estudio. Como forma de solucionarlo, se dedicaron unos días de trabajo más intensos que sirvieron para recuperar el tiempo perdido.

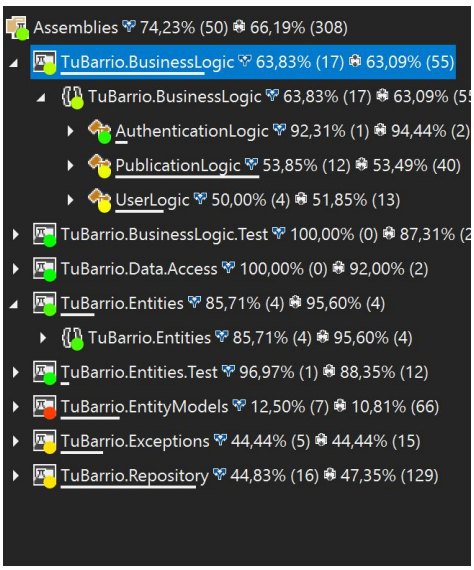
8.6 Actividades preventivas

Tal como se planificó, a lo largo del proyecto se utilizaron distintas prácticas que funcionaron como actividades preventivas. En primer lugar, se destaca la utilización de Clean Code y los lineamientos allí definidos para tener un código mantenible y fácil de entender. Esto facilitó en gran medida el trabajo en equipo dado que al tener cierta uniformidad en el código, cada integrante pudo entender rápidamente que los otros habían desarrollado y viceversa. La evidencia de uso de Clean Code.

Por otro lado, el uso de clean code y estandares de codificacion facilito tambien la realizacion de las revisiones. Dado los pocos recursos y el tiempo para realizar el proyecto, se decidió hacer revisiones ad hoc. Esas son sencillas, se pueden hacer de forma rápida y no requieren gran inversión de tiempo ni de recursos. En cuanto a sus beneficios, sirvieron para corregir defectos en etapas tempranas del desarrollo permitiendo la colaboración entre los miembros del equipo.

8.7 Actividades correctivas

Como actividades correctivas decidimos realizar distintos tipos de pruebas.
A continuación adjuntamos evidencia de las pruebas, si bien sea deseaba obtener un porcentaje mayor, debido al poco tiempo que teníamos para finalizar el proyecto, decidimos enfocarnos en los requerimientos funcionales y dejar de lado alguna pruebas unitarias no tan importantes. En el caso del BusinessLogic, obtuvimos un 63% de cobertura de las ramas, lo cual no es un porcentaje alto, pero al menos indica que se probaron más de la mitad de las ramas del paquete. En el caso de las otras clases el porcentaje nunca baja de 85%, lo cual indica que el código del servidor está relativamente bien cubierto.



Por otro lado, se llevaron a cabo pruebas funcionales para validar y verificar el cumplimiento de los requerimientos funcionales definidos en primera instancia. Sirvió para corroborar que se había logrado implementar los requerimientos establecidos y también para asegurarse que el resultado de las acciones era efectivamente el esperado. La tabla a continuación demuestra todos los escenarios probados y los resultados obtenidos:

Módulo de funcionalidad	Datos ingresados	Resultado esperado	Resultado obtenido
Registrar Usuario	Nombre:”” Apellido:” Telefono:” Email:”” Contraseña:””	Mostrar mensaje de error en los campos vacíos solicitando que se ingrese.	Ok
	Nombre:””	Mostrar mensaje de que se	

Registrar Usuario	Apellido:” Telefono:” Email:”seba@gmail.com” Contraseña:””	debe ingresar la contraseña para loguearse con google	Ok
Registrar Usuario	Nombre:”” Apellido:” Telefon:” Email:”seba@seba” Contraseña:”sebseba”	Mostrar mensaje que indique que el email es incorrecto	Ok
Registrar Usuario	Nombre:”Seba” Apellido:”Rodriguez” Telefon:”099112233” Email:”seba@gmail.com” Contraseña:”sebseba”	Registrar usuario e iniciar sesión con el nuevo usuario	Ok
Menu Principal	Opción profile	Mostrar la pantalla con información del usuario	Ok
Menu Principal	Todas las publicaciones	Mostrar la pantalla con todas las publicaciones	Ok
Menu Principal	Publicaciones seguidas	Mostrar la pantalla con todas las publicaciones seguidas por el usuario	Ok
Editar Perfil	Seleccionar Nombre, Apellido o Telefono	Se muestra una pantalla la cual le permite cambiar la opción seleccionada al usuario	Ok
Editar Perfil	Seleccionar opción cambiar imagen	Se activa la cámara y el usuario puede tomar una nueva fotografía	Ok
Modificar Datos	Presionar la flecha hacia atrás	Se muestra la pantalla de perfil del usuario con los datos actualizados	Ok
Cambiar imagen	Presionar el botón para sacar una foto	Se activa la cámara y el usuario puede tomar una nueva fotografía	Ok
Sacar foto	Presionar el botón para sacar una foto	Se activa la cámara y el usuario puede tomar una nueva fotografía	Ok
Publicacion	Crear nueva publicación	Se muestra la pantalla de	

		creación de publicación	Ok
Publicacion	Selecciona una publicación de la lista	Se muestra una pantalla con la publicación seleccionada	Ok
Publicacion	Seleccionar información de la publicación	Mostrar la información de la publicación seleccionada	Ok
Publicacion	Seguir publicación	Se muestra en la pantalla que estás siguiendo la publicación	Ok
Publicacion	Dejar de seguir publicación	Se muestra en pantalla que la publicación fue dejada de seguir	Ok
Publicacion	Ver comentarios de una publicación	Se muestra en pantalla una lista con todos los comentarios de la publicación	Ok
Publicacion	Agregar Comentario	Se muestra una pantalla en la cual el usuario puede ingresar un nuevo comentario.	Ok

8.8 Actividades de control

En la etapa de planificación de la primera entrega no se hizo referencia a ninguna actividad de control. Sin embargo, se considero necesario y valioso tener una instancia de reflexión al final de cada iteración donde se pudiese evaluar el trabajo realizado para ver si estaba de acuerdo con lo que se había planificado. De esta forma, se pudo detectar problemas en la estimación, atrasos/adelantos en el cronograma y tareas cumplidas/ sin cumplir al final de la iteración.

8.9 Gestion de cambios

Para este proyecto no fue necesario tener un plan de gestión de cambios ya que los requerimientos estaban claramente establecidos al principio y sabíamos que no iban a cambiar. Tampoco se le agregaron funcionalidades nuevas por lo que no fue necesario realizar gestión de cambios

8.10 Gestion de defectos

Mientras iban transcurriendo las iteraciones y durante el proceso de desarrollo de las mismas iba surgiendo defectos en la aplicación, es decir, elementos que no coinciden con como se había especificado los requerimientos funcionales en el inicio. Dichos defectos, una vez identificados, se registraron en una tabla y se priorizan para su posterior corrección, A continuación, se da evidencia de dicha gestión:

Id	Nombre	Explicacion	Prioridad	Se soluciona
1	Volver a la pantalla principal no funciona	Al intentar volver para atrás de cualquier pantalla y que te redirija a la pantalla principal la cual contiene el mapa de publicaciones no está funcionando	Alta	Si
2	Error con botones del menu	Cuando se abre el menú, los botones no están dirigiendo a las pantallas que se esperaría.	Alta	Si
3	Descripción y nombre de una publicacion estan cambiadas	Cuando se crea una publicación, se guarda como nombre de la publicación la descripción de la misma y como la descripción se guarda le nombre	Alta	Si
4	Problemas para agregar followers a una publicación	Cuando se un usuario quiere seguir una publicación se cae la aplicación	Alta	Si
5	Error al dejar de seguir una publicación	Cuando se deja de seguir una publicación se muestra un mensaje de publicación dejar de seguir satisfactoriamente pero el usuario sigue siguiendo esa publicación	Media	Si
6	No hay foto de publicacion	Al crear una publicación nueva, no figura la opción de agregarle una foto a la misma	Alta	Si
7	Problema al crear comentario	La aplicación deja de funcionar al momento de ingresar un comentario nuevo	Alta	Si

8	Al registrar un usuario se lo lleva a la pantalla de login	Al registrar un usuario, si dicha acción es correcta, se debe loguearse automáticamente al usuario y dirigirlo a la pantalla principal	Media	Si
9	Crear un comentario vacío	La aplicación permite crear un comentario vacío. No tiene sentido que funcione así, debe corregirse	Baja	Si
10	Falta de foto de perfil en lista de comentarios	En el listado de los comentarios de una publicación junto al mensaje se debe ver la foto del perfil del usuario que realizó dicho comentario	Baja	Si
11	Teléfono de usuario permite ingresar cualquier texto	Al ingresar el numero de telefono de un usuario no se verifica que sean únicamente números sino que se puede agregar cualquier carácter	Media	Si
12	Error al acceder a una publicación	Cuando se desea acceder a una publicación desde el mapa, la aplicación se cae	Alta	Si

8.11 Diseño

8.11.1 Backend

El diseño del backend se mantuvo igual a la planificación inicial. Se realizaron algunos cambios de pequeño porte a nivel de entidades que fueron surgiendo con el desarrollo del proyecto. A continuación se muestran los diagramas:

Diagrama de paquetes:

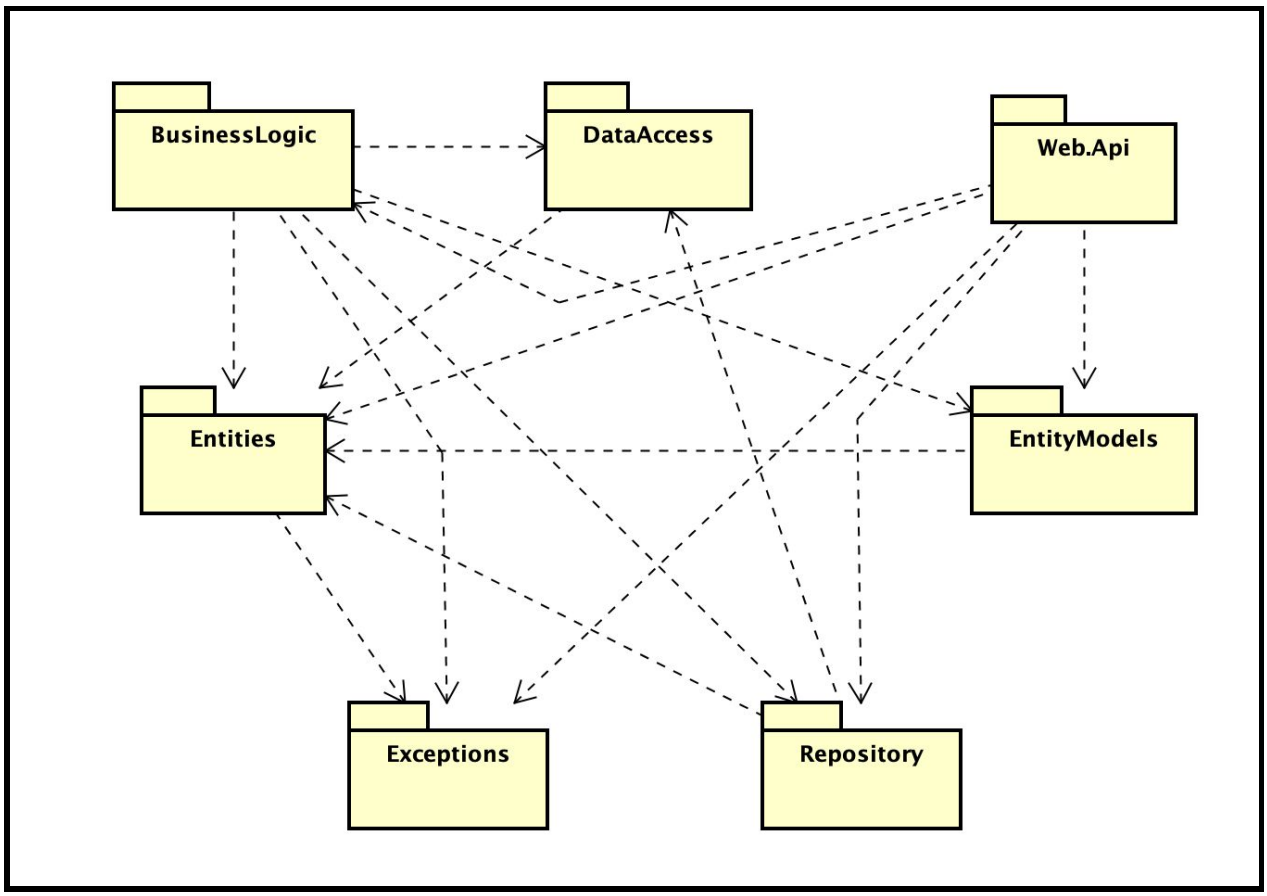
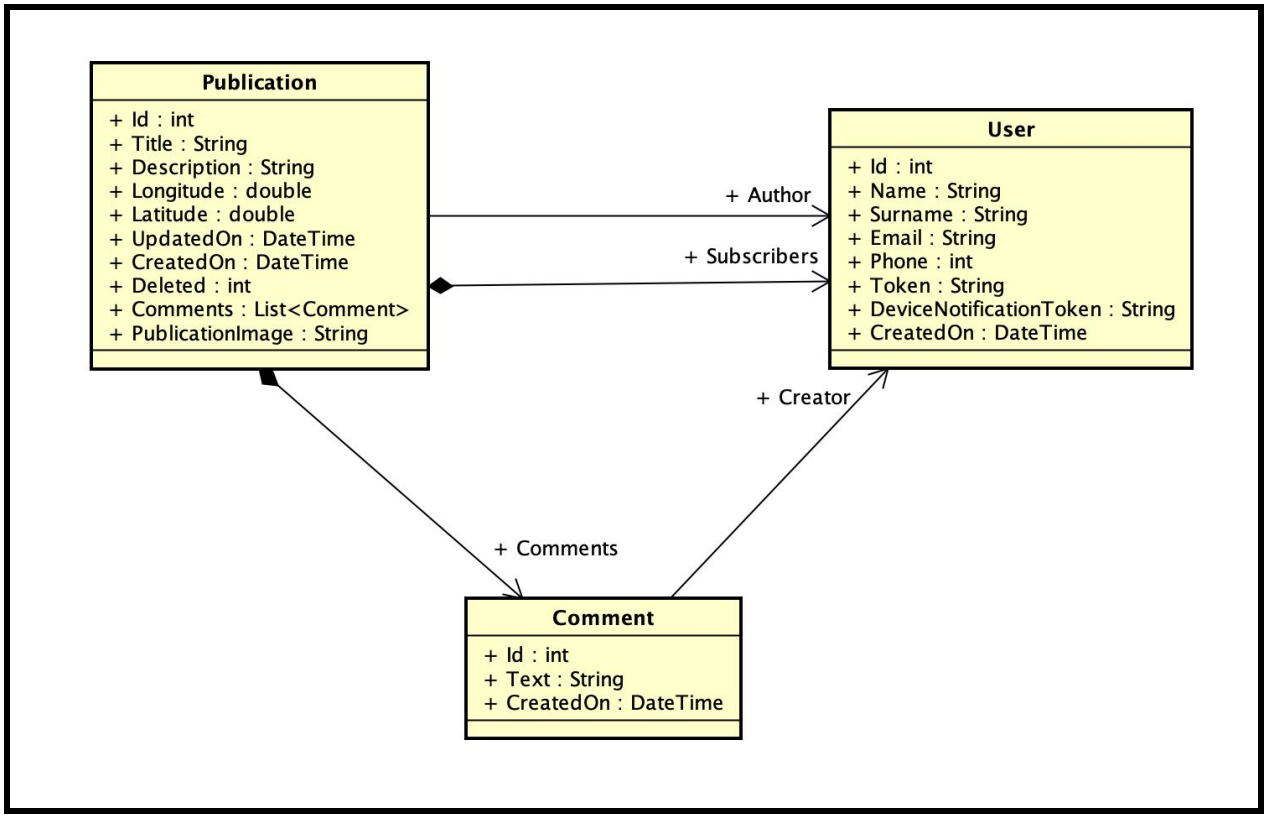


Diagrama de clases:

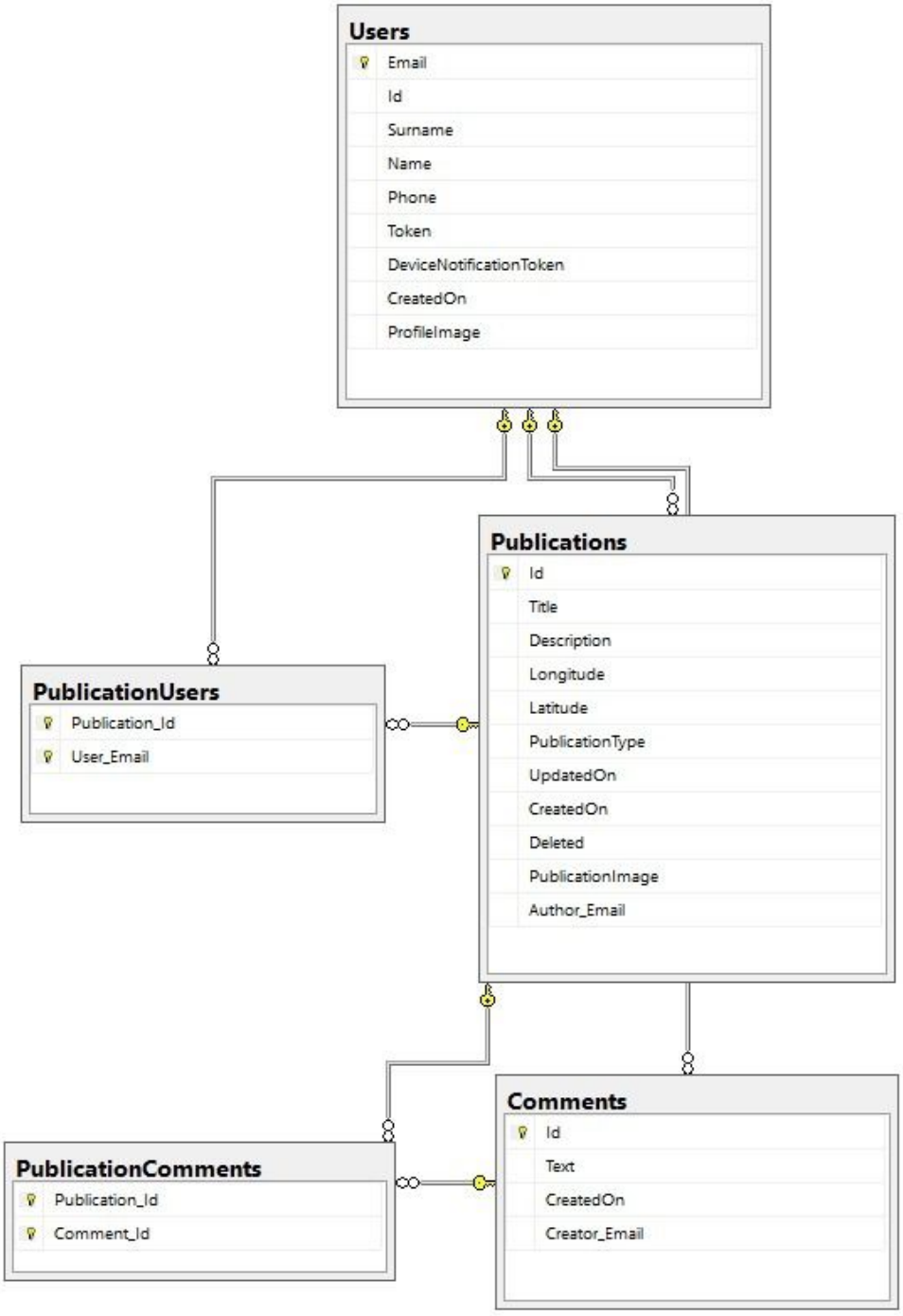
Se realizaron algunos cambios en el diagrama de clases en comparación con la primer entrega, a continuación de presenta el diagrama de clases actualizado.



En cuanto al diagrama de despliegue, se mantiene el mismo que en la primera entrega.

8.11.1 Base de datos

A continuación se presenta un diagrama del modelo de tablas de la base de datos.



8.11.2 Frontend - Codigo Android

En cuanto a la sección Android de la aplicación se tomaron algunas consideraciones que permitan organizar mejor la solución. Se dividió la solución en varias carpetas:

- Adapters: Contiene los adapters utilizados para desplegar información en la aplicación.
- Entities: Contiene todas las clases que representan sus respectivas entidades de dominio, las mismas se crearon para manejar los datos que vienen del backend de manera encapsulada.
- Firebase: Contiene las clases que permiten el manejo de notificaciones.
- Fragments: Contiene el fragment utilizado para los mapas.
- Layouts: Contiene las vistas de la aplicación, así como los items de la misma.
- Requests: Contiene las clases con las llamadas al server, cada llamada en una clase distinta.

Todas estas carpetas están dentro de la carpeta de la aplicación. Tener las clases distribuidas de esta manera permite que haya una mayor organización y se vuelve más fácil encontrar los elementos dentro de la solución.

Se buscó también favorecer el reúso del código, por este motivo se creo cada request como una clase separada, cuando se quiere hacer una llamada se llama a dicha clase, en vez de repetir el código innecesariamente.

8.11.3 UI/UX

Para diseñar la aplicación se hizo énfasis en la experiencia de los usuarios al momento de interactuar con el sistema. El diseño de las pantallas y el flujo de navegación que se realizó no se desvirtuó significativamente de lo planificado, con la excepción de algunos de los botones o posicionamiento de los elementos que en el momento se consideró.

Para ayudar con el diseño de la aplicación se utilizaron como guías las heurísticas de Nielsen. Se buscó utilizar un lenguaje natural y en sintonía con el problema de negocio, de manera que sea fácil de entender. Además, se intentó que el flujo de navegación resulte natural, y se lo organizó a propósito de esa manera. Todas las pantallas incluyen la información necesaria para el usuario, teniendo cuidado de no inundar las vistas de información.

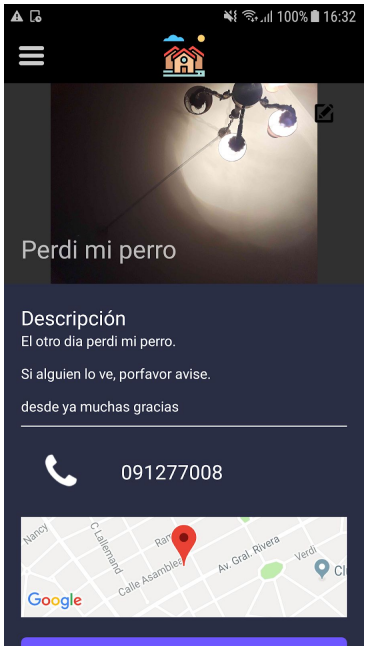
Se hace un gran uso de los mensajes explicativos a lo largo de la aplicación, esto permite que el usuario sepa el estado del sistema y logre comprender los errores cuando suceden eventos que no son normales. Un ejemplo de esto se aplica cuando el usuario está guardando una nueva publicación y deja algún dato vacío.

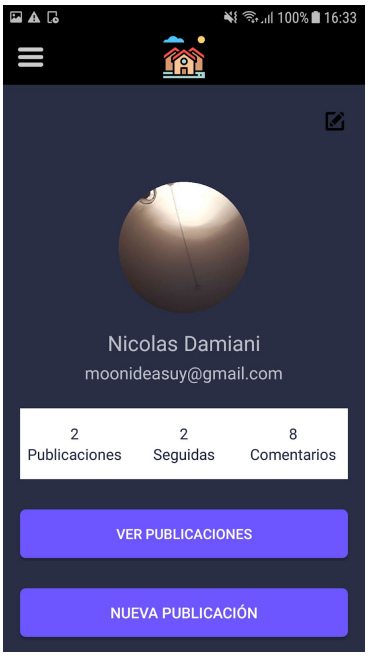
Con respecto a los colores de la aplicación, se buscó mantener una paleta de colores sencilla y sobria, así como minimalista. Esta decisión se tomó con el fin de que los colores no distraigan al usuario de la aplicación y lo permitan centrarse en el objetivo de la aplicación.

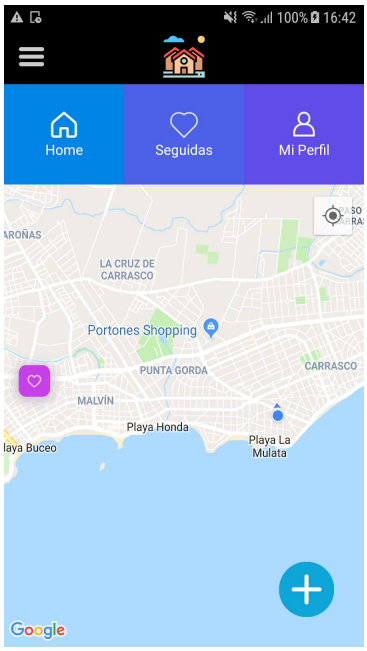
A lo largo de toda la aplicación se buscó mantener un mismo estilo de diseño en todas las pantallas, de esta manera se mantiene la cohesión y se le da un sentido de unidad a toda la aplicación. Se buscó realizar un diseño que tenga alguna semejanza con sistemas conocidos por los usuarios, de esta manera se reduce el tiempo de aprendizaje de la app.

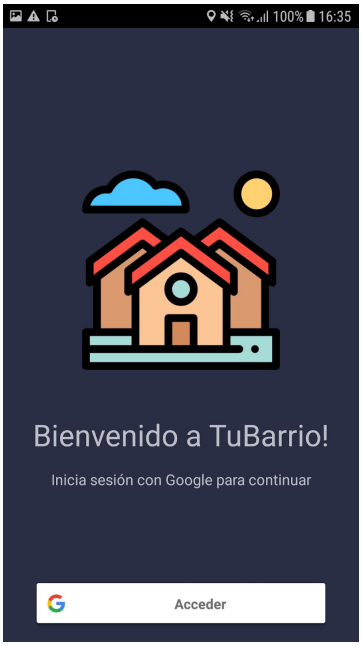
Pensando en retroceso, y aportando una mirada crítica, algo que falta en términos de UX de la aplicación es una sección de ayuda, en donde el usuario pueda ver instrucciones básicas donde pueda ver ejemplos del funcionamiento de la aplicación.

A continuación se incluyen algunas capturas de pantalla que evidencian lo dicho anteriormente:









9. Datos de prueba

A continuación, se pasará a especificar los datos de prueba incluidos en el archivo de base de datos con datos precargados. En el repositorio se podrán encontrar archivos de base de datos: uno vacío y otro con los datos que se indicarán a continuación. Cada archivo se podrá encontrar en dos formatos: .bak y .sql

Users

Id	Email	Nombre	Aellido	Telefono
1	moonideasuy@gmail.com	Damiani	Nicolas	091277008

Publications

Id	Titulo	Description	Longitude	Latitude
1	Perdi mi perro	El otro dia perdi mi perro. Si alguien lo ve, porfavor avise. desde ya muchas gracias	-56.119054555893	-34.8938749126813
3	El Gran Parque Central	Primer estadio mundialista	-56.1593617871404	-34.8858371573658

Comments

Id	Text	Creator_Email
1	Creo que lo vi en Cuareim	moonideasuy@gmail.com
2	Si lo veo te aviso!	moonideasuy@gmail.com

10. Manual de instalación

Pasos a seguir para la instalación del backend de la aplicación:

- Asegurarse de que IIS está instalado en la computadora en la cual se va a realizar el deploy. Para ello seguir los siguientes pasos:
    - Ir a Panel de control\Todos los elementos de Panel de control\ Programas y características.
    - Seleccionar “Activar o desactivar caracteristicas de Windows”.
    - Marcar la carpeta Internet Information Services y todo su interior.
    - Asegurarse de que el servidor SQL Server esté iniciado.
- Abrir el archivo web.config del proyecto, que se encuentra en la carpeta Web.Api dentro de TuBarrioServer, y modificar el connection string escribiendo el nombre que tiene el servidor SQL Server en la computadora donde se está haciendo el deploy.
- Copiar la carpeta Web.Api y pegarla en la siguiente ruta: C:\inetpub\wwwroot.
- Abrir el Administrador de Internet Information Services (IIS). Para llegar a esto buscar en Windows “inetmgr”.
- Agregar un nuevo sitio. Para ello seguir los siguientes pasos:
    - Click derecho sobre sitios y seleccionar Agregar nuevo sitio.
    - Escribir un nombre, en ruta físico seleccionar la carpeta Web.Api que acabamos de copiar en wwwroot y escribir el puerto 8080 o alguno que no esté en uso.
- Abrir SQL Management Studio. En la instancia de la base de datos buscar la pestaña Servidor/Inicios de Sesión.
- Click derecho sobre Inicio de Sesión y seleccionar Nuevo Inicio de Sesión.
- Como nombre de inicio de sesión agregar: IIS APPPOOL\Nombre_de_nuestro_sitio_en_IIS.
- Seleccionar Roles del servidor y marcar todos los roles. Luego dar aceptar

Pasos para correr la aplicación de Android:

- Crear un emulador o tener un dispositivo con Android.
- Buscar la apk de nombre TuBarrio.apk en la carpeta del proyecto.
- Hacer drag and drop al emulador para que se instale.

Asumiendo que la aplicación se correrá en una sola computadora, la dirección IP del release queda seteada (en el LoginActivity) para correr en un emulador con el servidor deployado en la misma computadora.