

Universidad ORT

Obligatorio 2

Ingeniería de Software en la Práctica

18 de Junio, 2019

Integrantes :

Matías Hernández - 169236

Sebastián Caraballo - 200562

Tabla de Contenido

Introducción	3
Objetivos	3
Gestión del Alcance	4
3.1. Release Plan	4
3.2. Sprint Plan	5
3.3. Desvíos y decisiones sobre cambios en la planificación	7
Alcance del producto	16
4.1. Product Backlog	16
As a user, I want to sign-up with my email, password, name, age and phone number so that I can create my account	16
As a registered user, I want to sign-in with my email and password	17
As a logged in user, I want to sign-out from my account	17
As a user, I want to edit my name, age, phone number and password	18
As a user, I want to take a photo so that I can save it	18
As a user, I want to add local photos so that I can save them at my library	19
As a user, I want to search by tags so that I can search my photos	19
As a user, I want to open a picture so that I can see its details	20
As a user, I want to delete multiple pictures so that these are erased from my library	20
As a user, I want to manage tags of an existing photo	21
As a user, I want to create albums by selecting a tag so that it creates an album with the tag matching photos	21
As a user, I want to see all my albums	22
As a user, I want to delete an existing album	22
As a user, I want to share a photo via Twitter	23
As a user, I want to share an album via Twitter	23
As a user, I want to see a map showing an image location	24
Gestión de la calidad	25
5.1. Proceso de ingeniería	25
5.2. Proceso de Desarrollo	26
5.3. Identificación y cumplimiento de estándares	27
5.3.1. Desarrollo guiado por comportamiento (BDD)	27
5.3.2. Principio INVEST	27
5.4. Aplicación y evidencia de Clean Code	28
5.5. Pruebas de Software	29
5.6 Métricas del código	30
Gestión de los riesgos	31

Gestión de la configuración	33
7.1. Estructura del Repositorio	33
7.2. Registro de Defectos	34
Arquitectura y Diseño	35
8.1. Decisiones de Arquitectura y Diseño del Frontend	35
8.2. Decisiones de Arquitectura y Diseño del Backend	45
8.2.1. BusinessLogic.Tests	46
8.2.2. BusinessLogic	47
8.2.3. BusinessLogic.Interface	48
8.2.4. DataAccess	48
8.2.5. DataAccess.Interface	49
8.2.6. Domain	50
8.2.7. GoogleApi	51
8.2.8. GoogleApi.Interface	51
8.3. Diseño de Experiencia de Usuario (UXD)	52
8.3.1. Estrategia y contenido	52
8.4. Diseño de Interfaz de Usuario (UID)	53
8.4.1. Look and Feel	53
8.4.2. Capacidad de respuesta e Interactividad	54
8.5. Modelo de Tablas de la Base de Datos	55
9. Instrucciones de uso	56
10. Bibliografía	57
10. Anexos	58
10.1. Anexo 1	58
10.2. Anexo 2	58

1. Introducción

El trabajo consta de la construcción de una aplicación móvil, que será posible ejecutarla desde un dispositivo corriendo en Android.

La aplicación, utilizará un backend al cual accederá a través de un servicio web REST. La misma será desarrollada en la plataforma de desarrollo .NET Core.

Se utilizará como metodología de desarrollo del proyecto, la metodología Ágil. La misma será especificada en los siguientes capítulos.

2. Objetivos

1 - Generar un producto para la fecha establecida que respete las características y requisitos definidos en el alcance del mismo.

2 - La aplicación debe poder ser ejecutada en dispositivos Android con versión mínima de SDK 27 (Android 8.1) y como objetivo en dispositivos Android con versión SDK 28 (Android 9.0).

3 - La aplicación debe utilizar las características de Android siguientes :

- Localización GPS
- Interactividad con aplicaciones de terceros (Twitter)
- Interactividad con aplicaciones base de Android (Galería)
- Interactividad con el File System de Android
- Interactividad con Web Services propios (Rest Api del Backend)
- Interactividad con Web Services de terceros (Google Vision Api)
- Utilización de SQLite Android para almacenamiento caché de datos
- Utilización de Google Maps
- Utilización de la aplicación Cámara
- Utilización de componentes de Interfaz Gráfica (Google Design Material)

3. Gestión del Alcance

El alcance del proyecto abarca desde la aceptación del plan de proyecto por parte del cliente, hasta la entrega del producto una vez finalizado.

3.1. Release Plan

Release - Semana 15 de Abril	
Historia	SP
As a user, I want to sign-up with my email, password, name, age and phone number so that I can create my account	5
As a registered user, I want to sign-in with my email and password	3
As a logged in user, I want to sign-out from my account	3
As a user, I want to take a photo so that I can save it	8
As a user, I want to edit my name, age, phone number and password.	5
As a user, I want to add local photos so that I can save them at my library	2
As a user, I want to search by tags so that I can search my photos	3
As a user, I want to open a picture so that I can see its details	5
As a user, I want to delete multiple pictures so that these are erased from my library	3
As a user, I want to manage tags of an existing photo	5
As a user, I want to create albums by selecting a tag so that it creates an album with the tag matching photos	3
As a user, I want to see all my albums	3
As a user, I want to delete an existing album	2
As a user, I want to share a photo via Twitter	5
As a user, I want to share an album via Twitter	5
As a user, I want to see a map showing an image location	5
Puntos:	65

Dado que la velocidad estimada del equipo es **8**, se calcula que el release podrá completarse en **9 sprints**.

3.2. Sprint Plan

Sprint 1 - Semana 15 de Abril	
As a user, I want to sign-up with my email, password, name, age and phone number so that I can create my account	5
As a registered user, I want to sign-in with my email and password	3
Puntos:	8

Sprint 2 - Semana 22 de Abril	
As a logged in user, I want to sign-out from my account	3
As a user, I want to edit my name, age, phone number and password.	5
Puntos:	8

Sprint 3 - Semana 29 de Abril	
As a user, I want to take a photo so that I can save it	8
Puntos:	8

Sprint 4 - Semana 6 de Mayo	
As a user, I want to add local photos so that I can save them at my library	2
As a user, I want to search by tags so that I can search my photos	3
As a user, I want to see all my albums	3
Puntos:	8

Sprint 5 - Semana 13 de Mayo	
As a user, I want to open a picture so that I can see its details	5
As a user, I want to delete multiple pictures so that these are erased from my library	3
Puntos:	8

Sprint 6 - Semana 20 de Mayo	
As a user, I want to manage tags of an existing photo	5
As a user, I want to create albums by selecting a tag so that it creates an album with the tag matching photos	3
Puntos:	8

Sprint 7 - Semana 27 de Mayo	
As a user, I want to delete an existing album	2
As a user, I want to share a photo via Twitter	5
Puntos:	7

Sprint 8 - Semana 3 de Junio	
As a user, I want to share an album via Twitter	5
Puntos:	5

Sprint 9 - Semana 10 de Junio	
As a user, I want to see a map showing an image location	5
Puntos:	5

3.3. Desvíos y decisiones sobre cambios en la planificación

La planificación de los Sprints que se planificó inicialmente, es la que se muestra arriba en la sección anterior.

A medida que fueron pasando los Sprints y fuimos realizando Sprint Reviews, se fueron retocando Sprint a Sprint los Story Points de las cards y también fue aumentando la velocidad del equipo considerablemente.

En un principio, se tuvo que realizar spikes de investigación, ya que no se contaba con toda la información necesaria para poder realizar el trabajo con la correcta arquitectura y funcionalidad.

Luego se verá que nos trancamos 1 Sprint entero por un bug que costó poder solucionarlo y que finalmente se solucionó.

Se va adaptando los sprints a medida que se ve que la velocidad del equipo aumenta, al conocer más la tecnología a aplicar.

El error en la planificación inicial fue asumir que la velocidad iba a ser continua, cuando se fue viendo Sprint a Sprint, que la velocidad del equipo aumentaba considerablemente.

Estos cambios son especificados a continuación, mostrando la planificación de cada Sprint y las decisiones tomadas en el Sprint Review luego de cada Sprint.

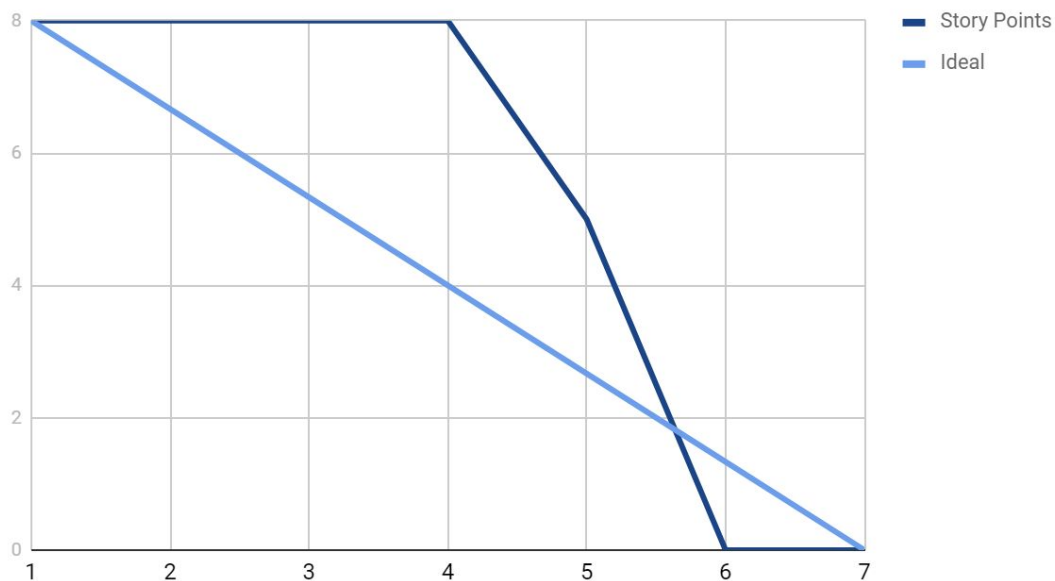
Sprint 1 (semana del 15 de Abril):

Sprint 1 - Semana 15 de Abril	
Desarrollo en Backend de los EndPoints del sprint.	5
Tecnologías y herramientas de desarrollo	3
Puntos:	8

A nivel de backend se terminan los features del sprint exponiendo el Endpoint apropiado.

A nivel de frontend se realiza un spike de investigación de las tecnologías Android para utilizar en el proyecto.

Story Points vs Days



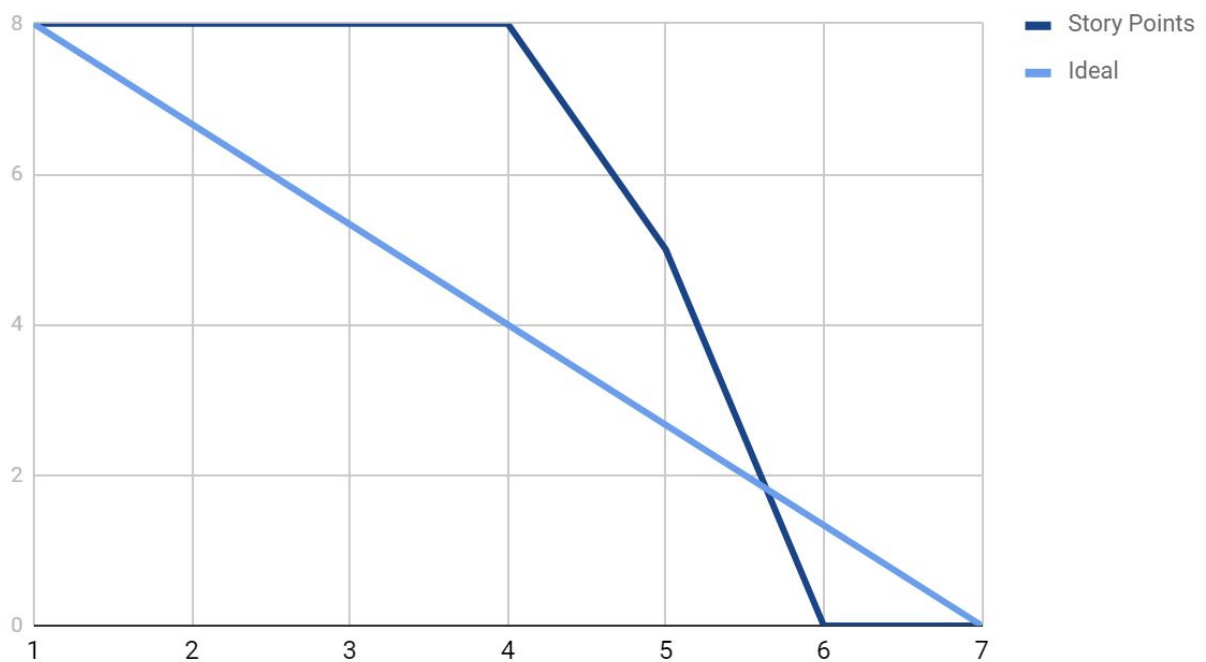
Sprint 2 (semana del 22 de Abril):

Sprint 2 - Semana 22 de Abril	
Desarrollo en Backend de los EndPoints del sprint.	3
Tecnologías y herramientas de desarrollo	5
Puntos:	8

A nivel de backend se terminan los features del sprint exponiendo el Endpoint apropiado.

A nivel de frontend se realizan spikes de investigación y pruebas con prototipos sobre la arquitectura a utilizar en el proyecto Android.

Story Points vs Days



Sprint 3 (semana del 29 de Abril): [6 SP]

Sprint 3 - Semana 29 de Abril	
As a registered user, I want to sign-in with my e-mail and password	3
As a logged in user, I want to sign-out from my account	3
Puntos:	6

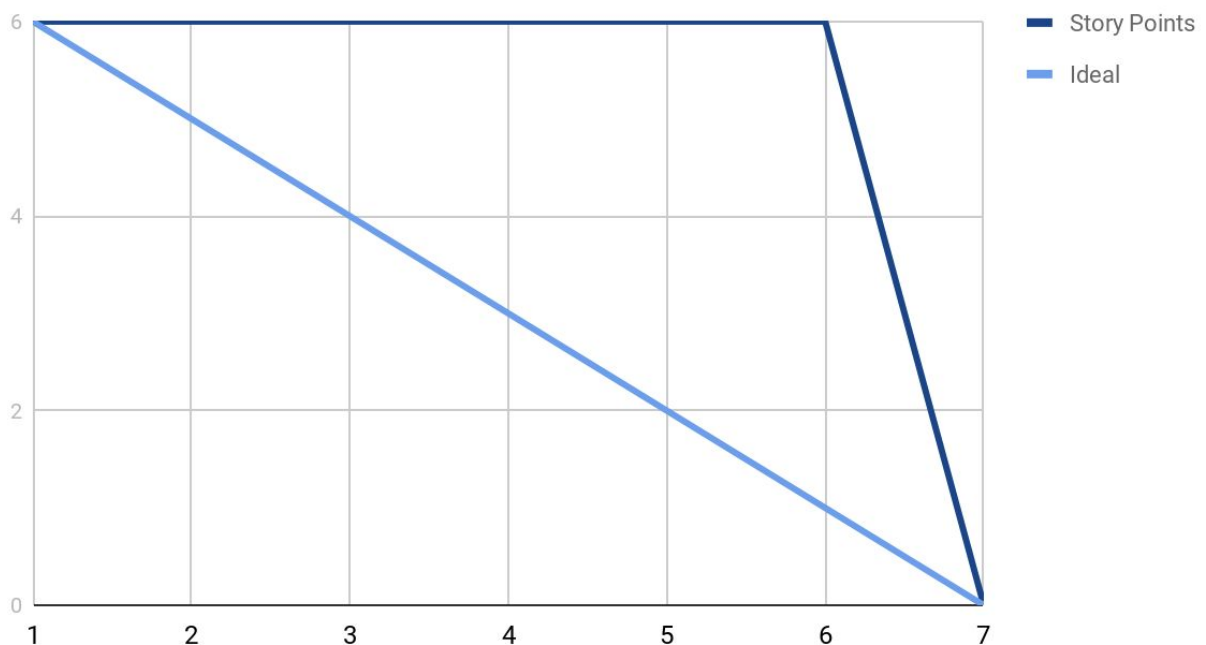
El 5 de mayo se finaliza :

[3 SP] As a registered user, I want to sign-in with my e-mail and password

El 5 de mayo se finaliza :

[3 SP] As a logged in user, I want to sign-out from my account

Story Points vs Days



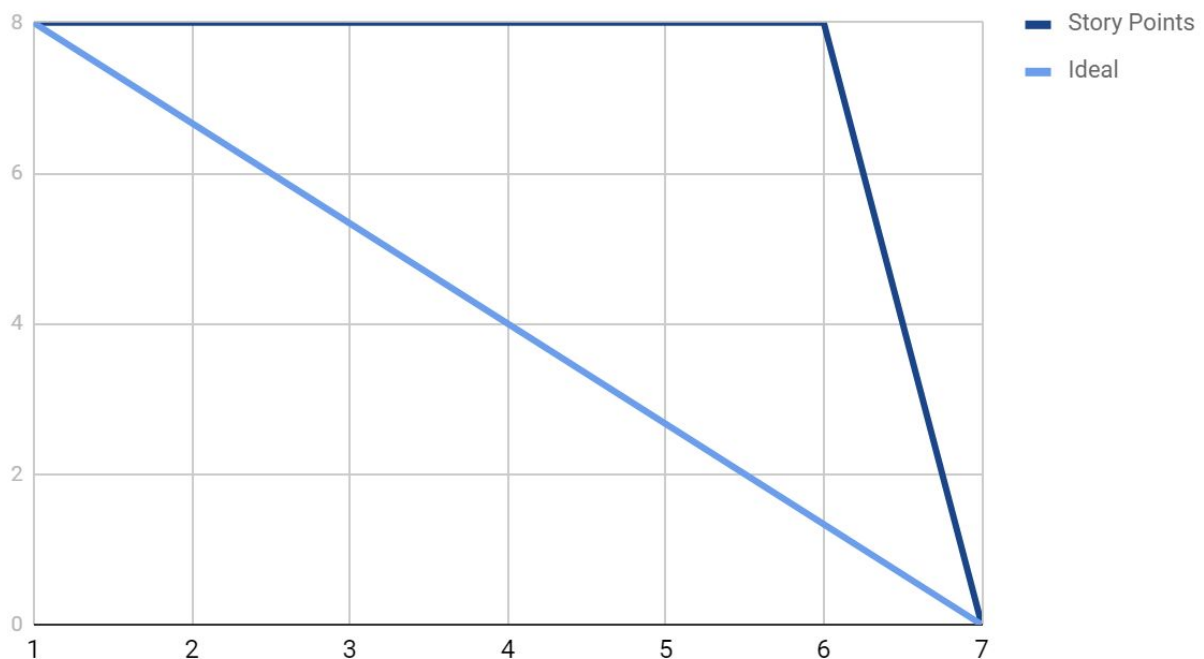
Sprint 4 (semana del 6 de Mayo):

Sprint 4 - Semana 6 de Mayo	
[Bug] Conectar la API desde Android	8
Puntos:	8

El 12 de mayo se finaliza :

[8 SP] [Bug] Conectar la API desde Android

Story Points vs Days



En el Sprint Review se toma la decisión de aumentar la cantidad de SP para el próximo sprint dado que se vé que hemos mejorado el nivel técnico con respecto a la tecnología y estuvimos 1 semana solucionando un Bug, lo cual nos hizo retrasar.

Sprint 5 (semana del 13 de Mayo):

Sprint 5 - Semana 13 de Mayo	
As a user, I want to edit my name, age, phone number and password	5
As a user, I want to delete multiple pictures so that these are erased from my library	8
Puntos:	13

Se consideró la idea de realizar una implementación que dejara seleccionar múltiples fotos y luego se pudiera borrar todas de una sola vez.

Esto consideramos que era innecesario y costoso, por lo que se recurrió a que se pudiera borrar una foto por vez. Repitiendo este mecanismo múltiples veces, se puede borrar múltiples fotos.

Esto hizo que de 8 SP de costo que tendría realizar esta funcionalidad, bajara considerablemente en tiempo y sea posible realizarlo en este Sprint.

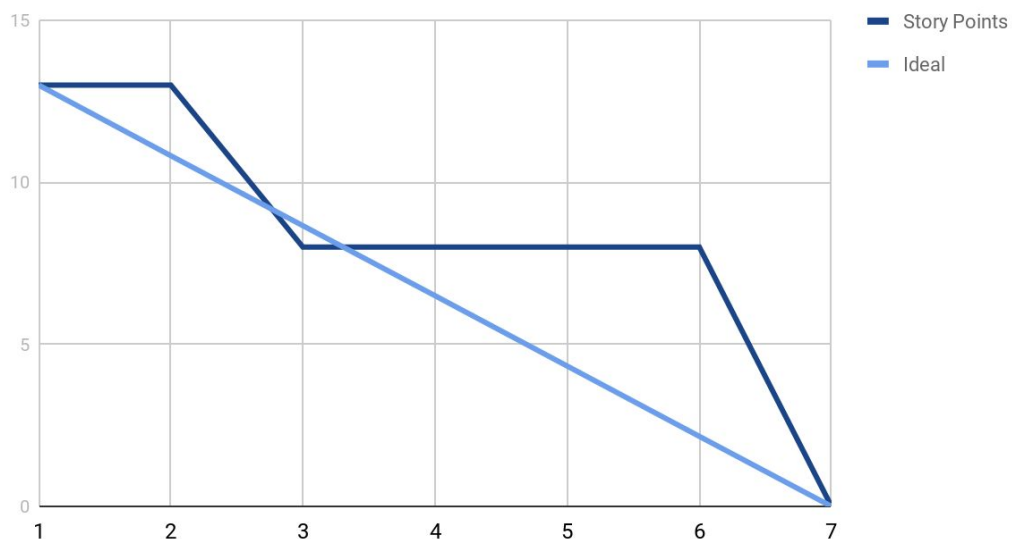
El 15 de mayo se finaliza :

[5 SP] As a user, I want to edit my name, age, phone number and password

El 19 de mayo se finaliza :

[8 SP] As a user, I want to take a photo so that I can save it

Story Points vs Days



En el Sprint Review se considera que se puede mantener este ritmo para el siguiente Sprint, por lo que se decide mantener los SP.

Sprint 6 (semana del 20 de Mayo):

Sprint 6 - Semana 20 de Mayo	
As a user, I want to add local photos so that I can save them at my library	5
As a user, I want to search by tags so that I can search my photos	3
As a user, I want to delete multiple pictures so that these are erased from my library	2
As a user, I want to open a picture so that I can see its details	3
Puntos:	13

El 20 de mayo se finaliza :

[5 SP] As a user, I want to add local photos so that I can save them at my library

El 25 de mayo se finaliza :

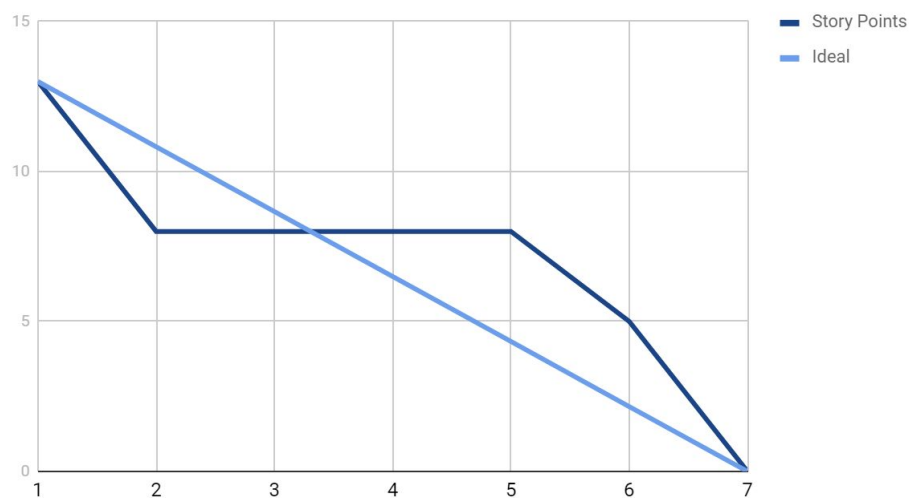
[3 SP] As a user, I want to search by tags so that I can search my photos

El 26 de mayo se finalizan :

[2 SP] As a user, I want to delete multiple pictures so that these are erased from my library

[3 SP] As a user, I want to open a picture so that I can see its details

Story Points vs Days



En el Sprint Review, se toma la decisión de completar los siguientes Story Points en el próximo sprint ya que se ve posible. Muchas funcionalidades siguientes, utilizan código reutilizable (fragmentos) ya desarrollados, por lo que entendemos que los Story Points que habíamos asignado en una primer instancia para las tarjetas, no representa la realidad actual de las mismas.

Ahora sin dudas tendrían una cantidad de Story Points menores y por lo tanto, se les re-asigna Story Points y se ve que es posible incluirlas en el próximo sprint.

Sprint 7 (semana del 27 de Mayo):

Sprint 7 - Semana 27 de Mayo	
As a user, I want to manage tags of an existing photo	3
As a user, I want to create albums by selecting a tag so that it creates an album with the tag matching photos	3
As a user, I want to delete an existing album	2
As a user, I want to see all my albums	3
As a user, I want to share a photo via Twitter	2
[Bug] Add Albums and Photos Many-To-Many relationship	1
As a user, I want to see a map showing an image location	2
[Bug] Check duplicate images after uploading one.	1
Puntos:	17

El 28 de mayo se finaliza :

[3 SP] As a user, I want to manage tags of an existing photo

El 30 de mayo se finaliza :

[3 SP] As a user, I want to create albums by selecting a tag so that it creates an album with the tag matching photos

[2 SP] As a user, I want to delete an existing album

El 31 de mayo se finalizan :

[3 SP] As a user, I want to see all my albums

[2 SP] As a user, I want to share a photo via Twitter

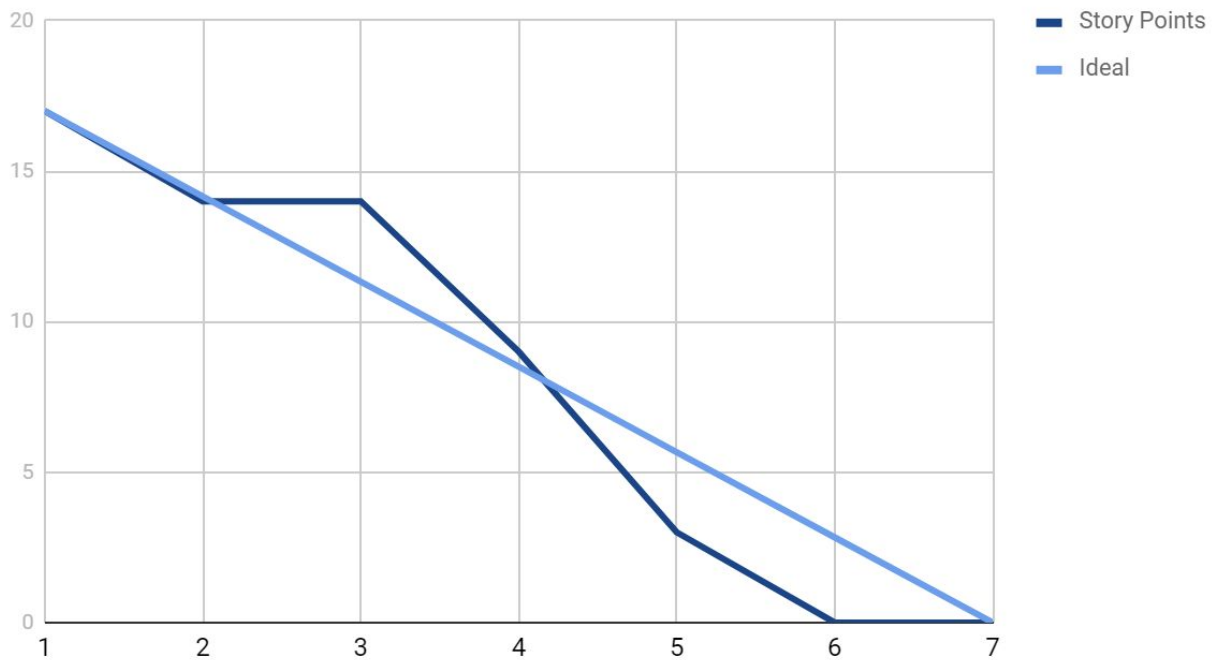
[1 SP] [Bug] Add Albums and Photos Many-To-Many relationship

El 1 de junio se finalizan :

[2 SP] As a user, I want to see a map showing an image location

[1 SP] [Bug] Check duplicate images after uploading one.

Story Points vs Days



Sprint 8 (semana del 3 de Junio):

El programa está finalizado, no se realizan acciones.

Sprint 9 (semana del 10 de Junio):

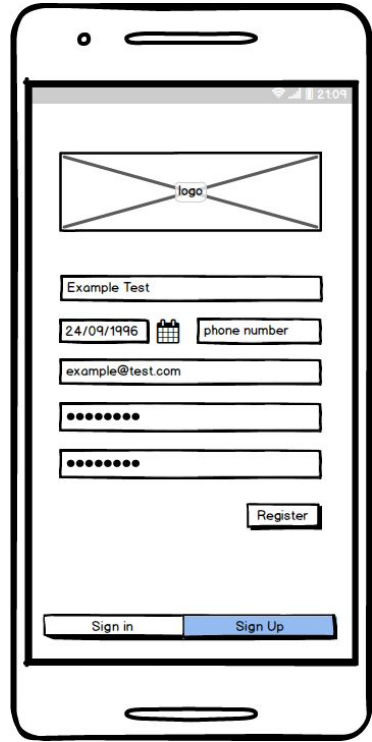
El programa está finalizado, no se realizan acciones.

4. Alcance del producto

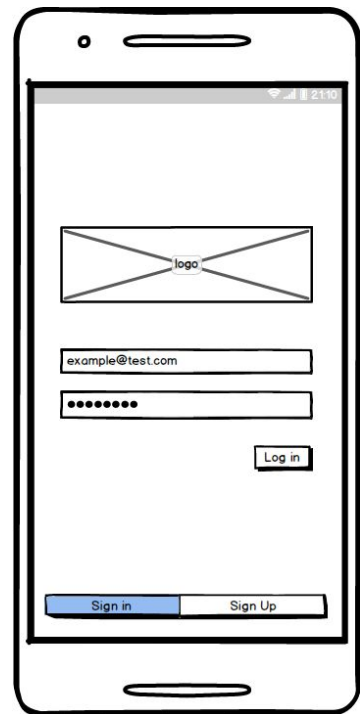
4.1. Product Backlog

Las historias de usuario fueron especificadas utilizando la herramienta Trello, se puede acceder al tablero mediante el siguiente link:

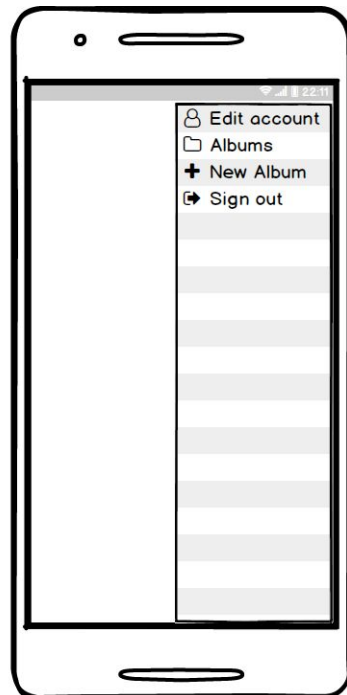
<https://trello.com/invite/b/X5DCrW61/da206b40c3c7a1949c47c9d918f1183a/ingenier%C3%ADa-de-software-en-la-pr%C3%A1ctica-obligatorio-android>

As a user, I want to sign-up with my email, password, name, age and phone number so that I can create my account	
Acceptance Criteria	
Given the user is at the sign-up page	
When the user types an email with a format like user@example.com	
And types an alphanumeric password with at least 8 characters twice	
And types a name	
And types age as a date	
And types a phone number with only numbers	
And taps on the Register button	
Then it creates an account and redirects the user to the log-in screen	
Puntos: 5	

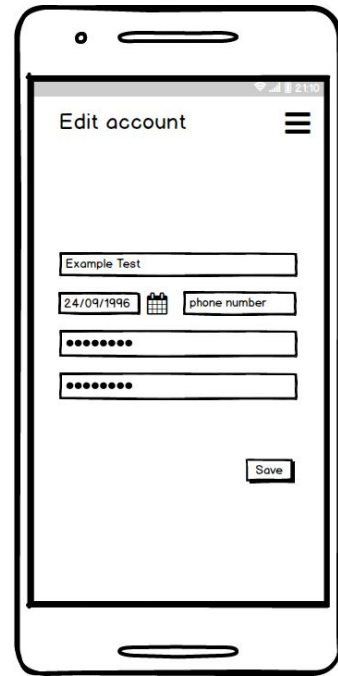
As a registered user, I want to sign-in with my email and password	
Acceptance Criteria	
Given the user is at the sign-in page	
When the user types an email with a format like user@example.com	
And an alphanumeric password with at least 8 characters	
And taps on the Login button	
Then it redirects the user to the homescreen	
Puntos: 3	



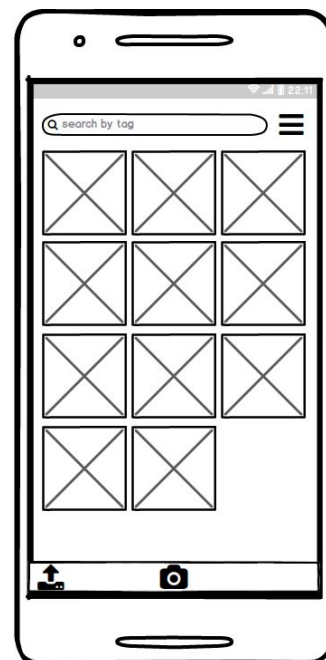
As a logged in user, I want to sign-out from my account	
Acceptance Criteria	
Given the user is logged in	
When the user taps on the Sign out option at the menu	
Then it redirects the user to the log-in screen	
Puntos: 3	



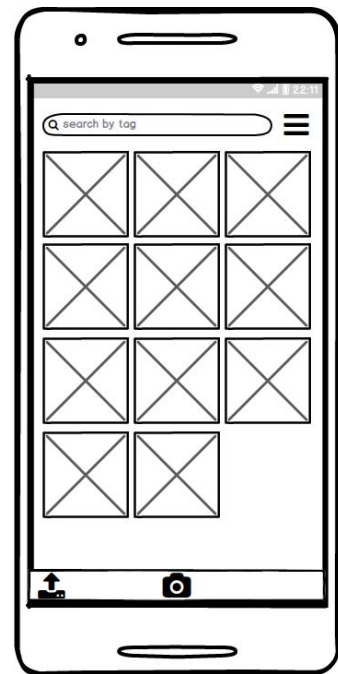
As a user, I want to edit my name, age, phone number and password	
Acceptance Criteria	
Given that the user taps on the Edit account option menu	
When the user types a new alphanumeric password with at least 8 characters twice	
And types a new name	
And types a new age as a date	
And types a new phone number with only numbers	
And taps on the Save button	
Then it redirects the user to the homescreen	
And shows a toaster displaying a success message	
Puntos: 2	



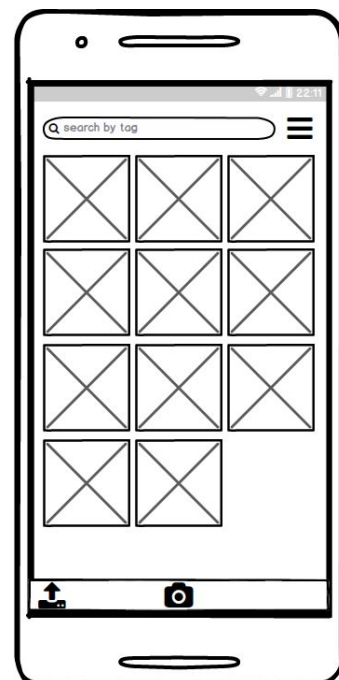
As a user, I want to take a photo so that I can save it	
Acceptance Criteria	
Given that the user is at homescreen	
When the user taps on the take a photo button	
Then the camera roll of the device will open	
Given the user is taking a picture at camera roll	
When the user takes a picture and taps on the upload button	
Then the photo will start uploading and appear at homescreen	
Puntos: 8	

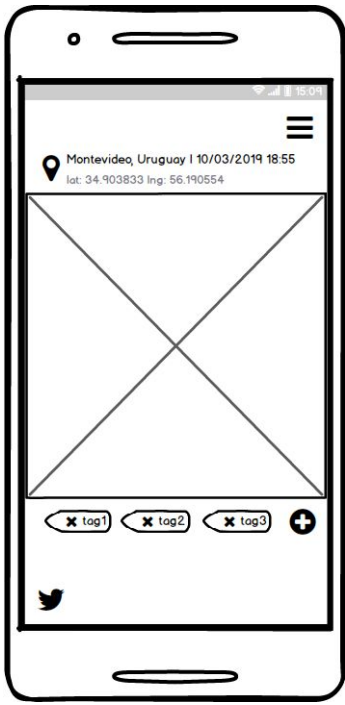


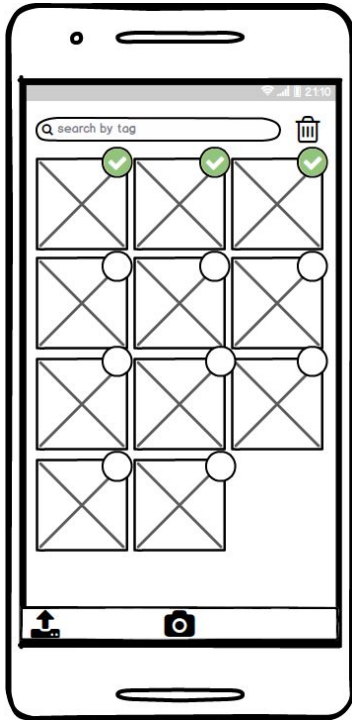
As a user, I want to add local photos so that I can save them at my library	
Given that the user is at homescreen	
When the user taps on the upload photo button	
Then the user will select personal photos	
When the user finishes and taps on the add photos button	
Then the photos will start uploading and appear at homescreen	
Puntos: 5	

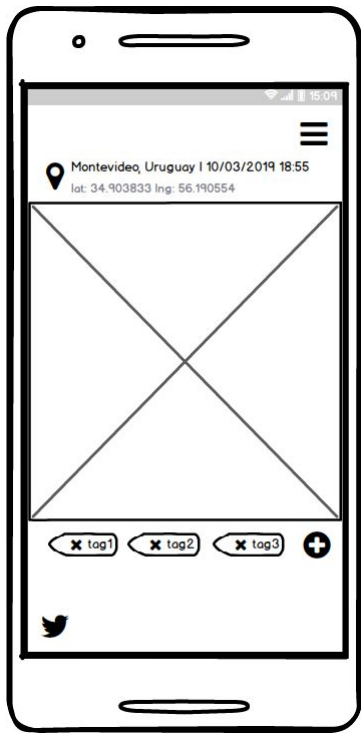


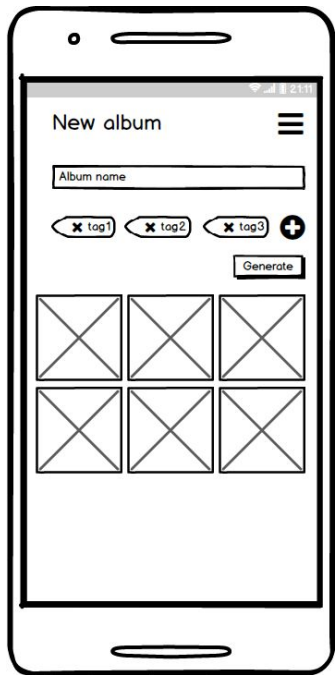
As a user, I want to search by tags so that I can search my photos	
Acceptance Criteria	
Given that the user is at homescreen	
When the user taps on the search bar	
Then the search screen will appear	
When the user types the text and taps on the search button	
Then the pictures grid will be populated with the matching photos by tag	
When the user wants to cancel the search	
Then the user taps on the go back button	
And the user will be redirected to homescreen	
Puntos: 5	

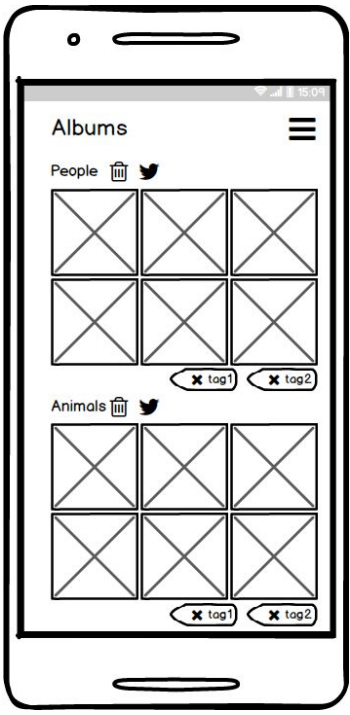


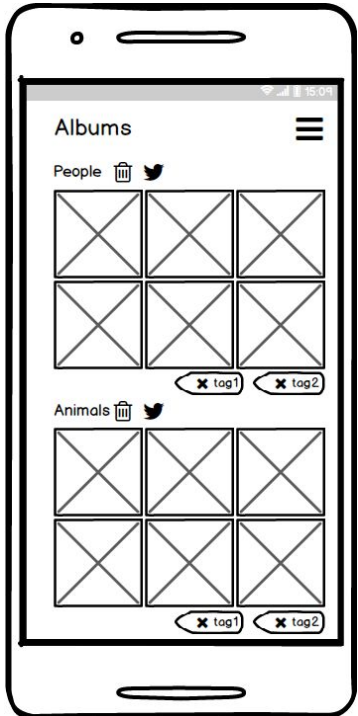
As a user, I want to open a picture so that I can see its details	
Acceptance Criteria	
Given that the user is at homescreen	
When the user taps on a picture	
Then the picture details screen is opened	
And it shows the picture in larger mode and its details (date, location, tags)	
	
Puntos: 5	

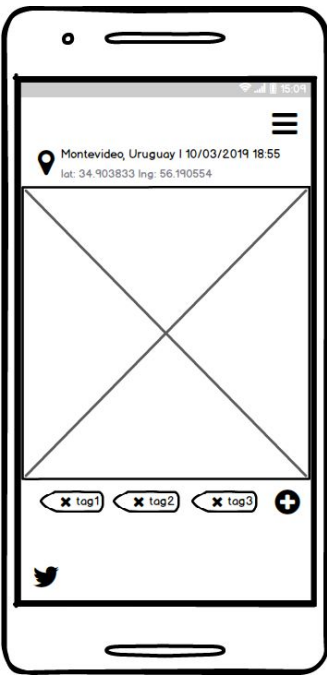
As a user, I want to delete multiple pictures so that these are erased from my library	
Acceptance Criteria	
Given that the user is at homescreen	
When the user selects one or multiple images to be deleted	
Then it appears the delete button	
When the user taps on the delete button	
Then the pictures disappear from the library	
	
Puntos: 3	

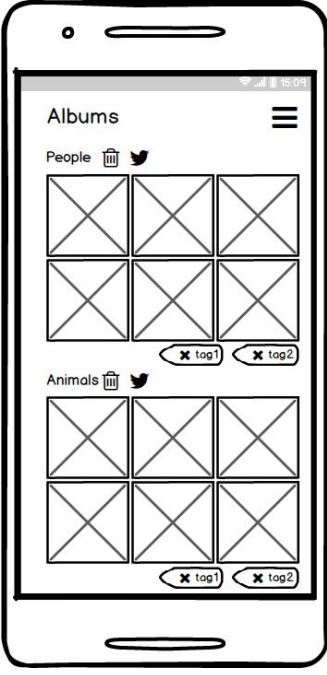
As a user, I want to manage tags of an existing photo	
Acceptance Criteria	
Given that the user is on the picture detail screen	
When the user taps on the add tag button	
And the user types the text	
And taps on the add button	
Then it displays the created label	
When the user taps on a delete button next to a tag	
Then that tag disappears	
	
Puntos: 5	

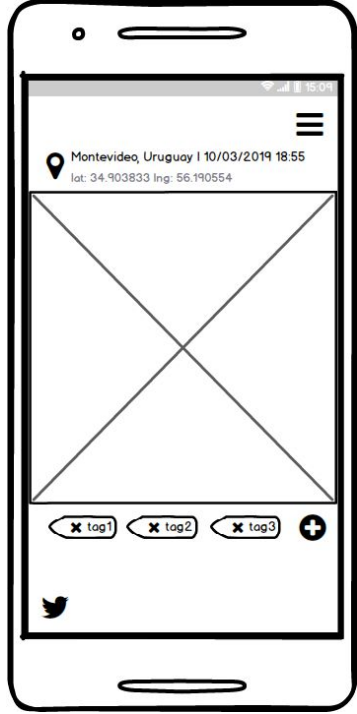
As a user, I want to create albums by selecting a tag so that it creates an album with the tag matching photos	
Acceptance Criteria	
Given that the user taps on the New Album option menu	
When the user selects one or multiple tags	
And the user types an album name	
And the user taps on the Generate Album button	
Then the user will be redirected to the album screen	
	
Puntos: 8	

As a user, I want to see all my albums	
Acceptance Criteria	
Given the user is at homescreen	
When the user taps on the Albums option menu	
Then all the albums appear with their photos and their tags	
	
Puntos: 3	

As a user, I want to delete an existing album	
Acceptance Criteria	
Given that the user taps on the Albums option menu	
When the user taps on the Delete Album option menu	
Then the user will be redirected to the Albums screen	
	
Puntos: 3	

As a user, I want to share a photo via Twitter	
Acceptance Criteria	
Given that the user is on the picture detail screen	
When the user taps on the Twitter share button	
Then the Twitter app will be opened	
And the New Tweet activity will be opened with a tweet with this format: "Here is my photo taken at #{location goes here}" on #{date goes here}: #{link to the image goes here}.	
And the photo app will display the photo from the link	
	
Puntos: 5	

As a user, I want to share an album via Twitter	
Acceptance Criteria	
Given that the user taps on the Albums option menu	
When the user taps on the Twitter share button next to the album	
Then the Twitter app will be opened	
And the New Tweet activity will be opened with a tweet with this format: "Here is my photo album #{album name goes here}: #{link to the album goes here}.	
And the photo app will open displaying the album from the link	
	
Puntos: 5	

As a user, I want to see a map showing an image location	
Acceptance Criteria	
Given that the user is on the picture detail screen	
When the user taps on the location info	
And the coordinates are present	
Then it opens a map displaying the place the picture was taken	
	
Puntos: 5	

5. Gestión de la calidad

5.1. Proceso de ingeniería

Actividad del proceso	Etapas del ciclo PDCA
Release Planning	Plan
Sprint Planning	
Análisis	Do (Sprint)
Codificación	
Diseño	
Diseño de pruebas	
Ejecución de pruebas	
Sprint Review	Check
Sprint Retrospective	Act

Se utilizará el ‘Círculo de Deming’ como proceso de ingeniería a implementar en nuestra metodología Ágil que se optó utilizar en nuestro proyecto.

El mismo, está dividido en 4 etapas :

Plan, donde se encuentra el Release Planning y el Sprint Planning del proyecto (etapas tempranas de cada Sprint y/o Release).

Do, donde se realiza el Análisis, la Codificación, el Diseño de lo que se va a implementar, el Diseño de las pruebas que se van a realizar y la ejecución de las pruebas.

Check, se realiza el Sprint Review donde se ejecuta una demo del release hecho en el Sprint y se ven los resultados de las pruebas.

Act, se realiza la ceremonia Sprint Retrospective donde se evalúa con el método ‘Start, Continue and Stop doing’, donde se ve según lo realizado en dicho sprint, qué cosas se desea empezar a hacer, continuar haciendo o parar de hacer.

5.2. Proceso de Desarrollo

La instanciación del Proceso de Ingeniería PDCA descrito anteriormente, será de la siguiente forma :

Actividad	Resultado	Método	Herramientas	Estándar
Release Planning	Release Backlog	Story Mapping	Trello	User Stories, BDD
Sprint Planning	Product Backlog	Card Conversation Confirmation, User Stories, BDD	Trello	INVEST
	Sprint Backlog	User Stories, BDD	Trello	INVEST
	Sprint Burndown Chart	N/A	Google Docs	SCRUM Standard
Análisis	Gestión de Riesgos	Matriz de Riesgos	Google Docs	N/A
Diseño	Especificación de Diseño del Software (Cap. 8)	UML	Astah	UML 2.0
Codificación	Código fuente y Ejecutable	Object Oriented Programming	Visual Studio Community 2017, Android Studio, Microsoft SQL Server Express 2017	Clean code, SOLID, GRASP
Diseño de pruebas	Casos de prueba	Prueba de caja negra, Pruebas de exploración	Google Docs	IEEE 829
Ejecución de pruebas	Resultados de la prueba	N/A	Visual Studio Community 2017, Android Studio	N/A
Sprint Review	Evaluación del Sprint	Demo	Android Mobile	N/A
Sprint Retrospective	Cambios en el proceso	Start, Continue and Stop doing	Google Docs	N/A

5.3. Identificación y cumplimiento de estándares

5.3.1. Desarrollo guiado por comportamiento (BDD)

Desarrollo guiado por el comportamiento (behavior-driven development), es una metodología de desarrollo de software que esencialmente afirma que por cada unidad de software, un desarrollador de software debe:

- primero definir un conjunto de pruebas para la unidad;
- después implementar la unidad;
- finalmente verificar que la implementación de la unidad haga exitosa las pruebas.

BDD señala el comportamiento en términos de las historias de usuario. Cada historia de usuario debe seguir la siguiente estructura de seguimiento:

Título: La historia debe tener un título claro y explícito.

Narrativa

Una pequeña sección de introducción que especifique

- quién (cuál negocio o rol del proyecto) es el conductor o principal parte interesada de la historia (el actor que recibe beneficios de la historia)
- qué efecto se quiere en la historia, señalado por la parte interesada
- qué remuneración tendrá la parte interesada a partir de este efecto

Criterios de aceptación o escenarios

Es una descripción de cada caso específico de la narrativa. Tal escenario tiene la siguiente estructura:

- Comienza al especificar la condición inicial, la cual es asumida como verdadera al inicio del escenario. Esto puede consistir en una causa o varias.
- Después declara cuál evento causa el inicio del escenario.
- Finalmente, declara el resultado esperado, en una o más cláusulas.

Dicho estándar fue utilizado para especificar las historias de usuario tal como se puede apreciar en la sección [Product Backlog](#).

5.3.2. Principio INVEST

INVEST es un principio utilizado en nuestro proyecto para escribir las historias de usuario de tal forma que respeten las características de calidad. A continuación se describe el acrónimo:

- Independiente (Independent) - la historia no debe depender de otras historias.
- Negociable (Negotiable) - las historias no son contratos explícitos y deben dar espacio a la discusión.
- Valuable (Valuable) - una historia debe entregar valor a los stakeholders.
- Estimable (Estimateable) - siempre se debe poder estimar el tamaño de una historia.
- Pequeña (Small) - las historias deben ser pequeñas para mantener el nivel de exactitud.
- Testeable (Testable) - la descripción de la historia debe proveer la información necesaria para hacer el desarrollo de pruebas posible.

5.4. Aplicación y evidencia de Clean Code

A continuación, se enumeran los principios más importantes de Clean Code, extraídos del libro de Robert C. Martin.

- Comentarios
 - Comentario obsoleto
 - Un comentario anticuado, irrelevante e incorrecto es obsoleto. Por eso que no se codifica utilizando comentarios.
- Entorno
 - Las pruebas requieren más de un paso
 - Debería poder ejecutar todas las pruebas de unidad con un solo comando.
 - Efectivamente, se utiliza un único comando para correr las pruebas en el proyecto de pruebas MSTest BusinessLogic.Tests de la solución.
- Funciones
 - Demasiados argumentos
 - Las funciones deben tener un número reducido de argumentos. Lo mejor es que no tengan, seguido de uno, dos y tres argumentos.
 - A modo de ejemplo en [Anexo 1](#), notar como el máximo de argumentos es dos.
 - Argumentos de indicador
 - Los argumentos booleanos declaran abiertamente que la función hace más de una cosa.
 - Notar que en el código se cumple con esta condición ya que no se utilizan argumentos booleanos.
 - Intención desconocida
 - Queremos que el código sea lo más expresivo posible. Expresiones extensas, notación húngara y números mágicos distorsionan la intención del autor.
 - En nuestro obligatorio notar como los nombres de las funciones junto con sus argumentos expresan claramente la intención. (Ejemplo, SignIn en [Anexo 1](#))
 - Usar variables explicativas
 - Una de las técnicas más completas para que un programa sea legible consiste en dividir los cálculos en valores intermedios almacenados en variables con nombres descriptivos.
 - Encapsular condicionales
 - Extraiga funciones que expliquen el cometido de la condicional (Ver ejemplo en [Anexo 2](#))
 - Evitar condicionales negativas
 - Debe expresar las condiciones como positivas.
 - Las funciones solo deben hacer una cosa
 - Si una función solo realiza los pasos situados un nivel por debajo del nombre de la función, entonces hace una cosa.

5.5. Pruebas de Software

📁 Obligatorio (30 pruebas)	
▲ ✓ BusinessLogic.Tests (30)	2 s
▲ ✓ BusinessLogic.Tests (30)	2 s
▲ ✓ AlbumLogicTests (5)	2 s
✓ CreateAlbum	2 s
✓ CreateAlbumWithExistentName	91 ms
✓ CreateAlbumWithNoName	22 ms
✓ CreateAlbumWithNoTags	91 ms
✓ DeleteAlbum	104 ms
▲ ✓ PhotoLogicTests (4)	74 ms
✓ AddTag	4 ms
✓ DeletePhoto	42 ms
✓ DeleteTag	20 ms
✓ SearchByTags	6 ms
▲ ✓ SessionLogicTests (11)	32 ms
✓ SignIn	3 ms
✓ SignInWithInvalidCredentials	2 ms
✓ SignInWithInvalidEmail	3 ms
✓ SignUp	2 ms
✓ SignUpWithInvalidDate	1 ms
✓ SignUpWithInvalidEmail	4 ms
✓ SignUpWithInvalidName	5 ms
✓ SignUpWithInvalidPasswordConfirmation	1 ms
✓ SignUpWithInvalidPasswordWithoutNumbers	3 ms
✓ SignUpWithInvalidPhone	1 ms
✓ SignUpWithInvalidShortPassword	1 ms
▲ ✓ UserLogicTests (10)	61 ms
✓ GetAllAlbums	34 ms
✓ GetAllPhotos	8 ms
✓ Update	3 ms
✓ UpdateWithInvalidDate	1 ms
✓ UpdateWithInvalidName	2 ms
✓ UpdateWithInvalidPasswordConfirmation	1 ms
✓ UpdateWithInvalidPasswordWithoutNumbers	1 ms
✓ UpdateWithInvalidPhone	1 ms
✓ UpdateWithInvalidShortPassword	1 ms
✓ UploadPhoto	3 ms

Es de vital importancia las pruebas de la lógica de negocio en la aplicación. Para esto, se creó el proyecto **BusinessLogic.Tests**. En la figura se muestran las pruebas codificadas, y notar las distintas validaciones realizadas. Las pruebas fueron escritas utilizando TDD (Test Driven Development), es decir, primero se escribe la prueba que falla (**RED**), luego se codifica el código mínimo y necesario para que la prueba pase (**GREEN**) y por último se refactoriza el código existente para eliminar code smells (**REFACTOR**). TDD asegura una completa cobertura del código, ya que al codificar el mínimo necesario para que las pruebas pasen, no se agrega código a la solución inutilizado. [4]

5.6 Métricas del código

Métricas de la API obtenidas a partir de la herramienta de análisis de Visual Studio. Notar que el índice de mantenimiento es alto. En parte esto se debe a la utilización de Test Driven Development.

Jerarquía	Índice de mantenimiento	Complejidad ciclomática	Profundidad de herencia	Acoplamiento de clases	Líneas de código
BusinessLogic (Debug)	74	35	1	33	0
BusinessLogic	74	35	1	33	0
AlbumLogic	76	11	1	21	68
PhotoLogic	76	7	1	16	52
SessionLogic	76	8	1	8	42
UserLogic	70	9	1	18	87
BusinessLogic.Interface (Debug)	99	19	2	9	55
BusinessLogic.Tests (Debug)	74	59	1	52	714
DataAccess (Debug)	65	88	2	184	1.159
DataAccess	78	66	2	113	0
AlbumRepository	73	9	2	35	61
AppContext	72	4	2	39	26
BaseRepository<T>	82	21	1	14	109
ContextFactory	77	8	1	13	47
ContextType	100	1	1	0	4
PhotoRepository	76	9	2	24	57
UserRepository	72	14	2	50	99
DataAccess.Migrations	55	22	2	75	1.159
DataAccess.Interface (Debug)	98	10	2	4	31
Domain (Debug)	96	105	2	27	312
Domain	96	99	2	27	276
Album	90	9	1	9	28
AlbumPhoto	100	8	1	3	8
InvalidCredentials	98	2	2	1	8
InvalidDate	98	2	2	1	8
InvalidEmail	98	2	2	1	8
InvalidName	98	2	2	1	8
InvalidPasswordConfirmation	98	2	2	1	8
InvalidPasswordFormat	98	2	2	1	8
InvalidPhone	98	2	2	1	8
Photo	98	23	1	7	21
Tag	91	11	1	4	29
User	90	34	1	16	94
Domain.Exceptions	98	6	2	1	36
GoogleAPI (Debug)	78	13	1	42	138
GoogleAPI	75	7	1	34	65
GoogleServices	75	7	1	34	61
GoogleAPI.Properties	81	6	1	10	73
GoogleAPIInterface (Debug)	100	4	0	1	11
GoogleAPIInterface	100	4	0	1	11
IGoogleServices	100	4	0	1	7
WebApi (Debug)	83	118	3	117	622

6. Gestión de los riesgos

Los riesgos del proyecto, serán gestionados a través de una matriz de riesgos. La matriz de riesgos identificará los riesgos más significativos que afectan cada sprint del proyecto.

Riesgo	Especificación	I	PO	OT	M	Plan de Respuesta	Plan de Contingencia
Bug	Condición : Posible descubrimiento de un Bug.	3	0,6	1	1,8	Aceptar	Se deja varios Sprints sin abarcar la totalidad de los Story Points que podría, con la finalidad de ingresar Bugs en caso de ser necesario.
	Consecuencia : Problemas para cumplir el alcance del proyecto en el tiempo indicado.						
	Contexto : La fecha de la entrega es fija.						
Enfermedad	Condición : Posible enfermedad de un integrante del equipo.	3	0,4	1	1,2	Aceptar	El resto del equipo, tomará el lugar del integrante faltante, y lo reemplazará en sus tareas.
	Consecuencia : Falta de un integrante del proyecto durante un período por culpa de una enfermedad.						
	Contexto : La fecha de entrega es fija.						
Accidentes técnicos	Condición : Posible rotura de equipamiento de desarrollo.	4	0,2	1	0,8	Mitigar : Se previene la posible pérdida de datos, mediante la utilización de un repositorio GIT a través de GitHub.	Realizar horas extras de trabajo, para recuperar el tiempo perdido por la rotura del equipamiento.
	Consecuencia : Pérdida de datos, posible retraso en el calendario y en los costos.						
	Contexto : La fecha de entrega es fija.						
Falla del servicio GitHub	Condición : Posible caída del servicio de GitHub.	2	0,1	1	0,2	Mitigar : El desarrollador mantiene una copia local del repositorio en su equipo.	Utilización de otro servidor de GIT.
	Consecuencia : Problemas de coordinación entre el equipo, posible retraso en el sprint.						
	Contexto : La fecha de entrega es fija.						

Notas :

La tabla de riesgos debe estar ordenada primero por Ocurrencia en el Tiempo y luego por Magnitud.

Sigladas :

I - Impacto

PO - Probabilidad de Ocurrencia

OT - Ocurrencia en el Tiempo

M - Magnitud ($I \times PO$)

Escala de Impacto :

0 - Ninguno.

1 - Marginal.

2 - Poco importante.

3 - Importante (puede retrasar el proyecto).

4 - Crítica (puede detener el proyecto).

5 - Catastrófica (fracaso del proyecto).

Escala de Probabilidad de Ocurrencia :

0.0 - No probable.

0.2 - Poco probable.

0.4 - Probable.

0.6 - Muy probable.

0.8 - Altamente probable.

1.0 - Se convierte en problema.

Escala de Ocurrencia en el Tiempo :

1 - Inmediato.

2 - Mediano plazo (siguientes fases).

3 - Largo plazo (al final del proyecto).

7. Gestión de la configuración

7.1. Estructura del Repositorio

API: <https://github.com/sebastiancarballo/oblingsoftpr-api>

Android: <https://github.com/sebastiancarballo/oblingsoftpr-android>

Para el obligatorio se decidió trabajar sobre dos repositorios: uno para la API desarrollada en .NET Core, y otro repositorio para la aplicación de Android. A su vez, y debido a que se está utilizando una metodología ágil, es que se decidió realizar el desarrollo de las funcionalidades con **features branches** y **pull requests** para mantener un correcto registro de la evolución del código.

The screenshot shows a GitHub Pull Request (PR) interface. At the top, there are tabs for Code, Issues (0), Pull requests (1), Projects (0), Wiki, Insights, and Settings. The PR title is "As a registered user, I want to sign-in with my e-mail and password #2". Below the title, it says "sebastiancaraba... wants to merge 1 commit into develop from feature/signin". There are buttons for "Open" and "Edit". Below the PR title, there are statistics: Conversation (0), Commits (1), Checks (0), and Files changed (14). On the right side, there are sections for Reviewers (matiashrndz), Assignees (No one—assign yourself), Labels (None yet), Projects (None yet), and Milestone (No milestone). At the bottom, there is a "Notifications" section with an "Unsubscribe" button. The main content area shows a comment from sebastiancaraba... 15 days ago, which includes a Trello board reference and a description. The description lists two items: "As a registered user, I want to sign-in with my e-mail and password." and "Update Postman collection." Below the comment, there is a timeline of events: sebastiancarballo requested a review from matiashrndz 15 days ago, and sebastiancarballo force-pushed the feature/signin branch from a37c95e to 66f02ee 3 days ago.

A modo de ejemplo, para la [feature del inicio de sesión](#), en el repositorio de la API, se trabajó sobre la feature branch **feature/signin**. Luego, se realizó un **pull request** con el mero objetivo de agregar una descripción con la referencia hacia la historia de usuario especificada con la herramienta Trello en donde se describen todos los detalles de la funcionalidad.

7.2. Registro de Defectos

Para el registro de defectos, se utiliza la herramienta Trello (<https://trello.com>). Esta herramienta permite crear distintas listas para priorizar las tarjetas de un sprint. Para el caso de los defectos, se utiliza una nueva lista Bugs. Por lo tanto, si surge un defecto durante el sprint o en el review, se puede crear una nueva tarjeta explicando el defecto y la misma se coloca en la lista Bugs y se le asigna la etiqueta **Bug**. Luego, y dependiendo de la cantidad de defectos encontrados, es que se priorizan para su solución.

Trello permite asignar miembros a las tarjetas (como la presentada abajo), para mantener un flujo ordenado de trabajo.

App crashes when signing up with an invalid email format ✕
en la lista **Bugs**

ETIQUETAS
Bug +

Descripción Editar

Related card: [\(5\) As a user, I want to sign-up with my email, password, name, age and phone number so that I can create my account](#)

Añadir comentario

Escriba un comentario...

Guardar

Actividad Ocultar detalles

Sebastián Caraballo ha añadido esta tarjeta a Bugs
hace unos segundos

AÑADIR A LA TARJETA

- Miembros**
- Etiquetas**
- Checklist**
- Vencimiento**
- Adjunto**

POWER-UPS

- Google Drive**
- [Conseguir más Power-Ups](#)

ACCIONES

- Mover**
- Copiar**
- Seguir**
- Archivar**
- Compartir**

8. Arquitectura y Diseño

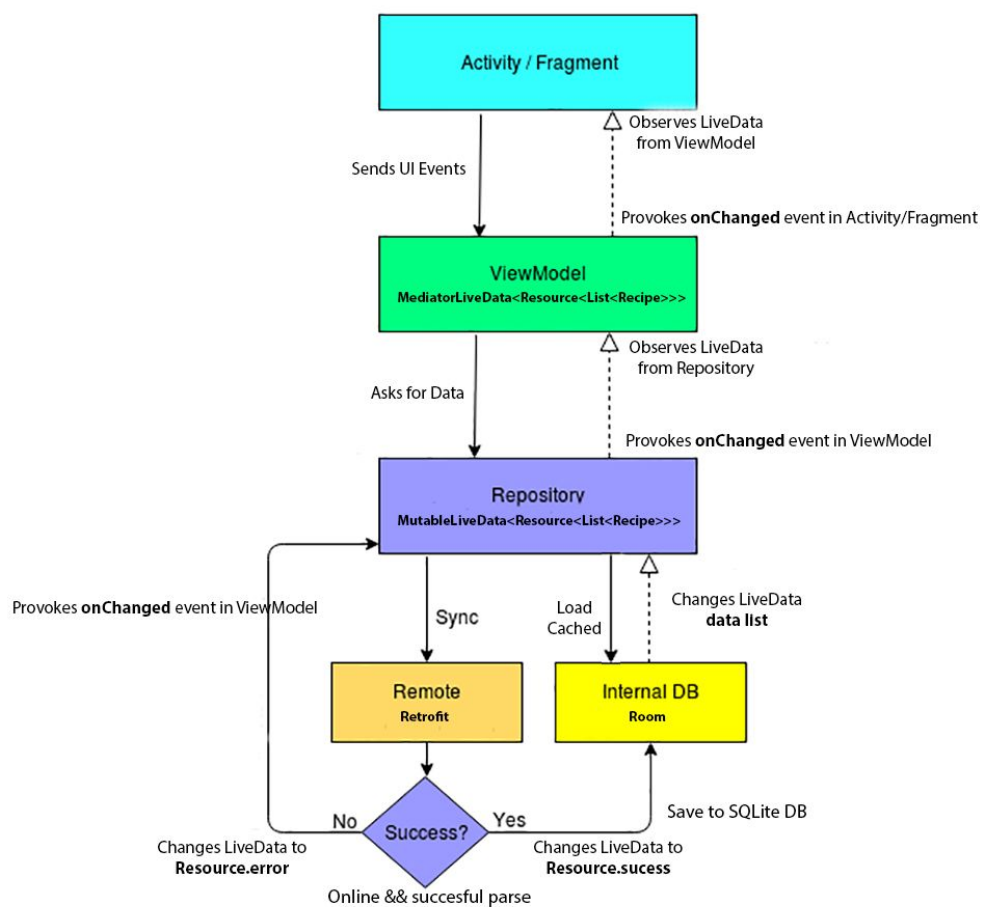
Se especificarán las decisiones de la arquitectura y diseño propuesto, con la justificación y diagramas correspondientes.

8.1. Decisiones de Arquitectura y Diseño del Frontend

Se realizó una aplicación Android para el Frontend de nuestro producto.

Se utilizó para la realización de la aplicación Android, el patrón de Arquitectura MVVM (Model-View-ViewModel).

Diagrama de actividad representando el patrón de arquitectura MVVM :



Explicación general del patrón

Una característica que se puede notar en este patrón es que cada componente, sólo depende del componente que está un nivel más abajo que él.

La única excepción a esta característica es el Repositorio, ya que el mismo depende de dos componentes que están a un nivel más abajo que él, esto es para tener la posibilidad de manejar una memoria Caché interna en la aplicación. La misma se guardará utilizando Room, en una base de datos SQLite.

Sentimos que esta característica es muy importante, ya que nuestra aplicación estará trabajando con carga de datos importantes (imágenes), utilizar para esos casos, la memoria caché del dispositivo, es fundamental para brindar una experiencia de usuario más fluida. Cuando la aplicación se encuentre inactiva, actualizará los datos en segundo plano, para intentar mantener lo más posible, dicha experiencia fluida.

Como biblioteca para acceder al Web Service Rest del Back End utilizaremos Retrofit 2.

Una pieza esencial para que funcione esta arquitectura descrita anteriormente, es la utilización de las clases `LiveData` y `MutableLiveData` (Subclase de `LiveData`). Gracias a estas clases proporcionadas por Google, podremos trabajar con el patrón Observer el cual notifica a la capa superior, de un cambio que haya ocurrido en su objeto observado de la capa inferior. Una vez dicho cambio llega al Activity / Fragment, se realiza la acción apropiada en la UI.

Cada objeto ViewModel, proporcionará los datos para un componente de UI específico (Activity o Fragment), el cual contendrá la lógica de negocio e interactúa con el repositorio adecuado. En esta capa no se utilizará ni se tendrá ningún conocimiento sobre los componentes de la UI, de manera que no se vea afectado por los cambios de configuración.

Implementación del patrón en nuestra aplicación

- View : Contiene todos los Activity y Fragments necesarios para brindar la experiencia de usuario apropiada.
 - MainActivity, Activity principal de la aplicación.
 - SignupActivity, la cual se encarga de mostrar el menú de Signup.
 - SignupActivity, la cual se encarga de mostrar el menú de Signin.
 - SimpleLifecycleActivity, las actividades que implementen de esta clase, podrán ser propietarios de un Lifecycle.
 - PhotoListFragment, la cual se encarga de mostrar la lista de photos, y permite sacar fotos y cargar fotos del móvil.
 - PhotoDetailFragment, la cual se encarga de mostrar el detalle de las fotos incluyendo el mapa donde aparece dónde fue tomada la foto, los tags entre otros datos.
 - NewAlbumFragment, la cual se encarga de mostrar las configuraciones para poder crear un nuevo álbum al ingresar a dicha funcionalidad.
 - HomeActivity, la cual se encarga de mostrar el home, donde aparecen los menú de opciones para acceder a las diferentes funcionalidades y maneja navegabilidad.
 - AlbumsFragment, la cual se encarga de mostrar los álbumes y las funcionalidades que brindan los álbumes.
 - AlbumPhotoListFragment, la cual se encarga de mostrar las fotos que tienen los álbumes dentro.
- View.Adapter : Contiene adaptadores para utilizar RecyclerView para los thumbnails de las fotos, los tags y los álbumes.
 - AlbumsRecyclerViewAdapter, la cual se encarga de adaptar los álbumes para que sean mostrados en el recyclerView.
 - TagsRecyclerViewAdapter, la cual se encarga de adaptar los tags para que sean mostrados en el recyclerView.
 - ThumbnailRecyclerViewAdapter, la cual se encarga de adaptar los thumbnails de las fotos para que sean mostrados en el recyclerView.
- ViewModel : Contiene todos los ViewModels necesarios para controlar las Views.
 - ViewModelFactory, que es una factory para crear ViewModels inyectando el repositorio correspondiente.
 - SignupViewModel, que se encarga de la lógica de SignupActivity y de llamar a UserRepository para datos.
 - AlbumViewModel, que se encarga de la lógica correspondiente a los álbumes.
 - EditAccountViewModel, que se encarga de la lógica correspondiente a editar un usuario.
 - PhotoViewModel, que se encarga de la lógica correspondiente a las fotos.
 - SigninViewModel, que se encarga de la lógica correspondiente a ingresar con el usuario al sistema.
 - SignoutViewModel, que se encarga de la lógica correspondiente a las salidas de los usuarios del sistema.

- **Repository** : Contiene todos los repositorios necesarios para manejar los datos de la aplicación.
 - **UserRepository**, que se encarga de tener las funciones necesarias para manejar usuarios. El repositorio se encarga de hacer la lógica comentada anteriormente de utilizar la memoria Caché, de cargar datos en la base de datos y de realizar llamadas a la WebApi.
 - **AlbumRepository**, que se encarga de tener las funciones necesarias para manejar los álbumes, almacenando los resultados en memoria caché y sabiendo cuándo se debe llamar a la Api para pedir los datos.
 - **SessionRepository**, que se encarga de tener las funciones necesarias para manejar los ingreso, registros y salidas de los usuarios en el sistema.
 - **PhotoRepository**, que se encarga de tener las funciones necesarias para manejar las fotos, almacenando los resultados en memoria caché y sabiendo cuándo se debe llamar a la Api para pedir los datos.

- **LocalDataSource** : Contiene todas las clases encargadas de los datos locales en la aplicación. Incluyendo la clase que contiene la implementación de la base de datos y las clases DAO encargadas del acceso a dichos datos.
 - **AppDatabase**, que extiende de **RoomDatabase** y tiene las tablas de la aplicación.
 - **UserDao**, se encarga de hacer las queries a la base de datos, relacionadas con los usuarios del sistema.
 - **PhotoDao**, se encarga de hacer las queries a la base de datos, relacionadas con las fotos del sistema.
 - **SessionDao**, se encarga de hacer las queries a la base de datos, relacionadas con las sesiones de usuarios del sistema.
 - **TagDao**, se encarga de hacer las queries a la base de datos, relacionadas con los tags de las fotos.

- **LocalDataSource.Model** : Contiene todos los modelos que se deben mapear a la base de datos como tabla.
 - **User**, que contiene todas las propiedades de un usuario.
 - **Photo**, que contiene todas las propiedades de una foto.
 - **Session**, que contiene todas las propiedades de una sesión.
 - **Tag**, que contiene todas las propiedades de un tag.
 - **Album**, que contiene todas las propiedades de un álbum.

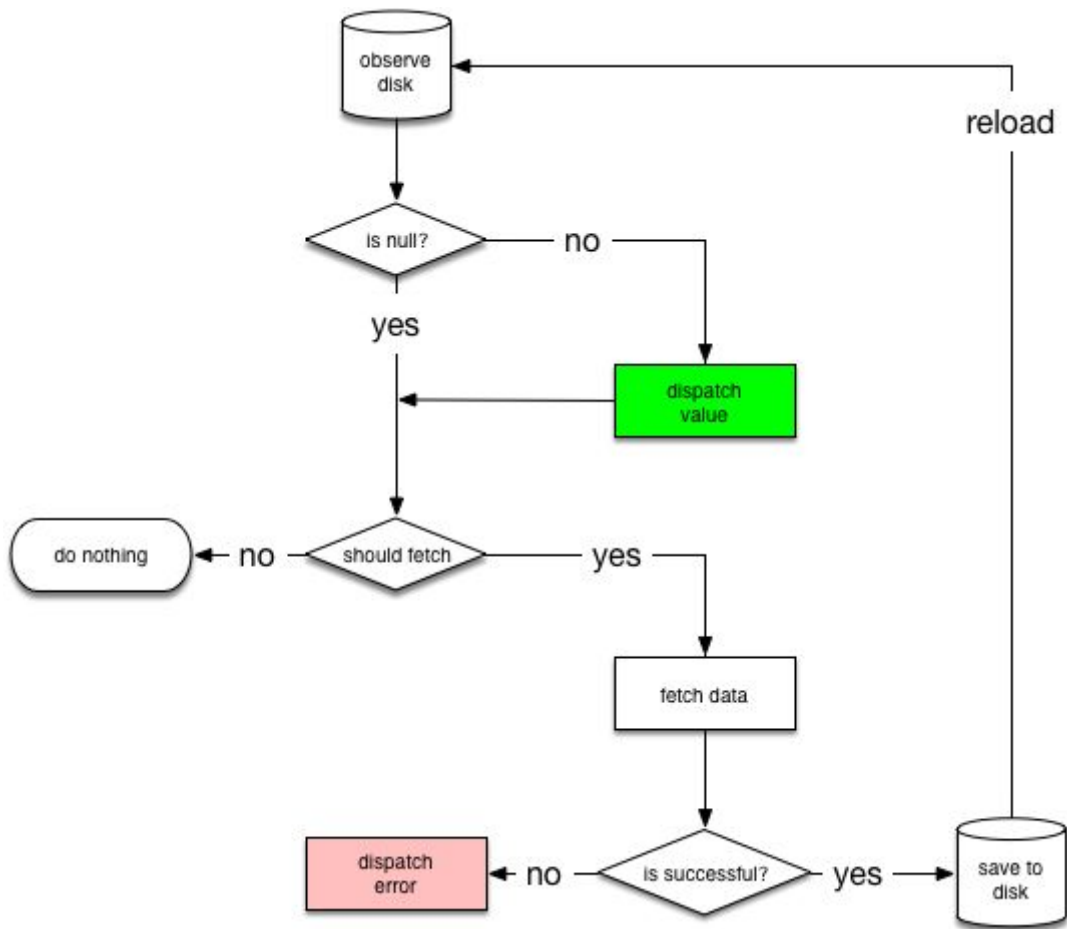
- **RemoteDataSource** : Contiene todas las clases encargadas de los datos conseguidos a través de la WebApi del backend y las clases necesarias para realizar, obtener e interpretar las respuestas de la WebApi.
 - **ApiResponse**, clase utilizada para analizar la respuesta de la WebApi. Recibe un **Response** (clase provista por Retrofit 2 con el contenido de la respuesta de la WebApi) y procesa la misma para convertirla en un **ApiResponse** con cuerpo, código y mensaje de error.
 - **AuthInterceptor**, clase que sirve para interceptar los Requests que se le hacen a la WebApi y determinar si es necesario o no que se utilice un token para el Request. Tiene

también la funcionalidad para ingresar el token en un field del header llamado `AUTHORIZATION`. Posee una anotación en particular para que se pueda marcar un Request como `NO_AUTHORIZATION`, en ese caso, no se ingresa token.

- `LiveDataCallAdapter`, es un Adaptador Retrofit que convierte la `ApiResponse` en una `LiveData<ApiResponse>`, esto favorece a la hora de observar esta respuesta desde una capa superior utilizando el patrón Observer.
 - `LiveDataCallAdapterFactory`, es la Factory que se encarga de crear la clase `LiveDataCallAdapter` con un tipo de dato en el cuerpo específico.
 - `NetworkBoundResource`, es una clase proporcionada por Google para proveer recursos conseguidos desde SQLite y desde la WebApi.
 - `Resource`, clase genérica que se utiliza para que guarde el dato y el estado de carga que tiene el mismo (esto es importante cuando se desea conocer si un recurso se está cargando, si dió un error o si fue exitoso.)
 - `Status`, enum que contiene los tres estados que puede tener un `Resource`, los cuales son : `SUCCESS`, `ERROR` y `LOADING`.
 - `Urls`, contiene las urls necesarias para poder llegar a cada EndPoint de nuestra WebApi.
- `RemoteDataSource.Services` : Contiene las interfaces correspondientes a cada Servicio que contará nuestra aplicación. Aquí se ejecutarán los request a la WebApi.
 - `UserService`, cuenta con los request apropiados para llamar a la WebApi del backend sobre servicios de usuario.
 - `PhotoService`, cuenta con los request apropiados para llamar a la WebApi del backend sobre servicios de fotos.
 - `SessionService`, cuenta con los request apropiados para llamar a la WebApi del backend sobre servicios de sesión..
 - `AlbumService`, cuenta con los request apropiados para llamar a la WebApi del backend sobre servicios de álbumes.
 - `RemoteDataSource.Model` : Contiene las clases DTO que se utilizan para transportar los datos para que sean enviados a la WebApi a través del servicio apropiado.
 - `SignupRequest`, contiene los datos necesarios para enviar el request de signup a la WebApi.
 - `AddTagRequest`, contiene los datos necesarios para enviar el request de agregar un tag a la WebApi.
 - `GetPhotoRequest`, contiene los datos necesarios para enviar el request de conseguir una foto a la WebApi.
 - `GetTagsRequest`, contiene los datos necesarios para enviar el request de conseguir los tags a la WebApi.
 - `PhotoAddRequest`, contiene los datos necesarios para enviar el request de agregar una foto a la WebApi.
 - `SearchByTagsRequest`, contiene los datos necesarios para enviar el request de buscar por tags a la WebApi.
 - `SigninRequest`, contiene los datos necesarios para enviar el request de signin a la WebApi.

- **UserUpdateRequest**, contiene los datos necesarios para enviar el request de actualizar un usuario a la WebApi.
- **RemoteDataSource.Deserializer** : Contiene deserializadores de los modelos del sistema que necesiten uno.
 - **UserDeserializer**, es un deserializador de un usuario, el cual es utilizado para poder parsear correctamente la fecha.
- **Injection** : Contiene los módulos necesarios para poder aplicar inyección de dependencia y establecer los principales componentes del sistema.
 - **AppComponent**, es un componente Dagger y utiliza a los Módulos **AppModule** y **UtilsModule** para inyectar dependencias de los mismos en otras clases.
 - **AppModule**, es un Módulo Dagger que provee el contexto de la aplicación para que sea inyectado en otra clase.
 - **MyApplication**, es la clase base de la aplicación, extiende de **Application** y posee el contexto y el **AppComponent** de la aplicación.
 - **UtilsModule**, es un Módulo Dagger que provee los métodos necesarios para inyectar dependencias en las clases de nuestra aplicación. Contiene métodos como **provideUserDao**, **provideDatabase**, **provideRetrofit**, entre otros.
- **Utilities** : Contiene las clases con utilidades que sean necesarias para nuestra aplicación.
 - **AbsentLiveData**, contiene una implementación de **LiveData** con el valor nulo.
 - **ConnectionChecker**, contiene un método que chequea la conexión a internet y retorna un booleano afirmando que hay o no hay conexión a internet.
 - **AppExecutors**, se encarga de ejecutar tareas utilizando los recursos de la aplicación.
 - **RecyclerViewUtility**, contiene una función que calcula el número de columnas que debe tener el **RecyclerView**.

Diagrama de actividad de NetworkBoundResource



Bibliotecas externas utilizadas por la aplicación

- Default

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation 'androidx.appcompat:appcompat:1.0.0-beta01'
implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
implementation 'androidx.legacy:legacy-support-v4:1.0.0'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'androidx.test:runner:1.1.0-alpha4'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.0-alpha4'
```

- ViewModel y LiveData

La clase LiveData se utilizará para retener datos observables. Otros componentes en la aplicación pueden supervisar cambios en este tipo de objetos, sin crear dependencias explícitas entre ellos.

```
def lifecycle_version = '2.0.0-beta01'
implementation "androidx.lifecycle:lifecycle-extensions:$lifecycle_version"
implementation "androidx.lifecycle:lifecycle-runtime:$lifecycle_version"
annotationProcessor "androidx.lifecycle:lifecycle-compiler:$lifecycle_version"
implementation "androidx.lifecycle:lifecycle-common-java8:$lifecycle_version"
implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0-rc01'
```

- Butterknife

Se utilizará para realizar bindings a campos y métodos para las Views.

```
def butterknife_version = '10.0.0'
implementation "com.jakewharton:butterknife:$butterknife_version"
annotationProcessor "com.jakewharton:butterknife-compiler:$butterknife_version"
```

- Dagger

Se utilizará Dagger para realizar las inyecciones de dependencia.

```
def dagger_version = '2.13'
implementation "com.google.dagger:dagger:$dagger_version"
implementation "com.google.dagger:dagger-android-support:$dagger_version"
annotationProcessor "com.google.dagger:dagger-compiler:$dagger_version"
compileOnly "javax.annotation:jsr250-api:1.0"
implementation "javax.inject:javax.inject:1"
```

- Android Material Design

Biblioteca que utilizaremos para el diseño visual de nuestra aplicación.

```
def androidmaterial_version = '1.0.0'
implementation "androidx.annotation:annotation:$androidmaterial_version-beta01"
implementation "com.google.android.material:material:$androidmaterial_version"
implementation "androidx.fragment:fragment:$androidmaterial_version"
implementation "com.android.support:support-annotations:28.0.0"
implementation "com.wdullaer:materialdatetimepicker:3.2.2"
```

- Retrofit

Se utilizará para hacer las llamadas a la WebApi.

```
def retrofit_version = '2.3.0'
implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
implementation "com.squareup.retrofit2:converter-gson:$retrofit_version"
implementation "com.squareup.retrofit2:converter-scalars:$retrofit_version"
implementation "com.itkacher.okhttpprofiler:okhttpprofiler:1.0.5"
```

- RxJava

Alternativamente (o complementariamente) a los LiveData, se puede utilizar esta biblioteca para observar datos sin depender directamente de la fuente de dónde sale el dato.

```
implementation 'io.reactivex.rxjava2:rxandroid:2.0.1'
implementation "io.reactivex.rxjava2:rxjava:2.1.8"
implementation 'com.jakewharton.retrofit:retrofit2-rxjava2-adapter:1.0.0'
```

- Timber

Biblioteca que se utiliza para realizar logs de la aplicación que proporciona mejoras en las utilidades básicas de los logs nativos.

```
implementation 'com.jakewharton.timber:timber:4.7.1'
```

- ROOM

```
def room_version = '2.0.0'
implementation "androidx.room:room-runtime:$room_version"
annotationProcessor "androidx.room:room-compiler:$room_version"
```

- Picasso

```
def picasso_version = '2.71828'  
implementation "com.squareup.picasso:picasso:$picasso_version"  
implementation 'androidx.recyclerview:recyclerview:1.0.0'
```

- Twitter Kit

```
def twitter_version = '3.1.1'  
implementation "com.twitter.sdk.android:tweet-composer:$twitter_version"
```

- Google Maps

```
implementation "com.google.android.gms:play-services-maps:16.0.0"
```

8.2. Decisiones de Arquitectura y Diseño del Backend

La solución se diseñó separando las principales responsabilidades en distintos proyectos.

Con el objetivo de generar código mantenible y extensible, es que se crearon proyectos de implementación e interfaces. La descripción actual se realizó en base al progreso existente de código al momento de ser escrito este documento.

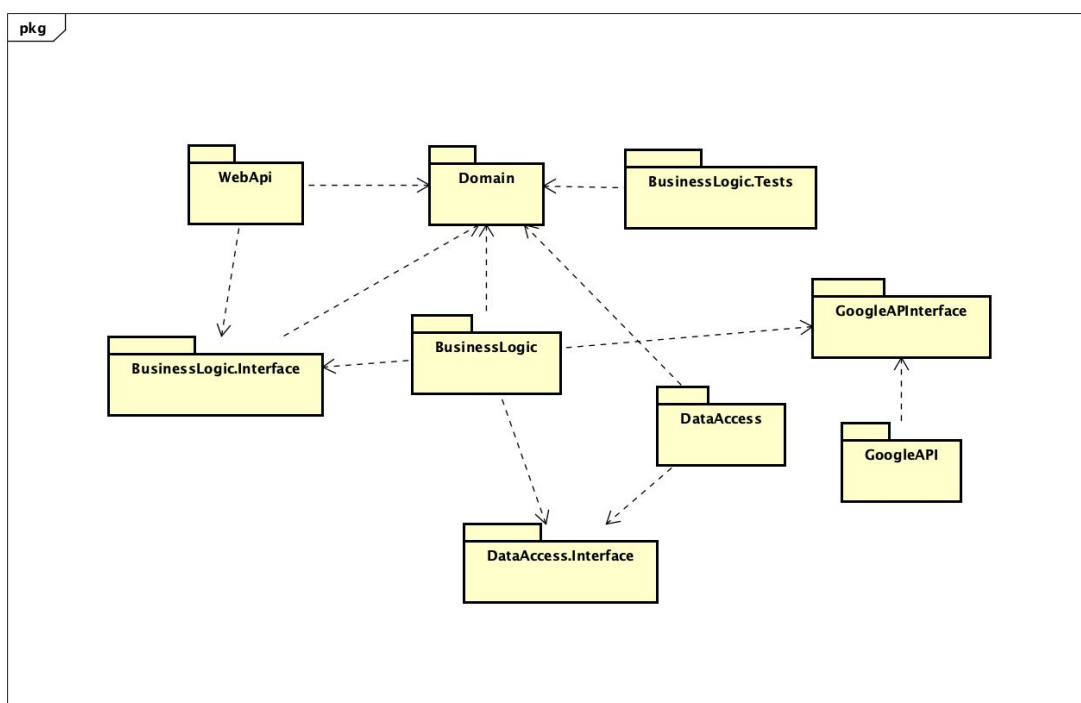
BusinessLogic.Tests contiene todas las pruebas de la lógica de negocio. **BusinessLogic.Interface** contiene todas las interfaces que utiliza el cliente que consumirá los servicios especificados en ella (en este caso, la WebAPI). **BusinessLogic** contiene la implementación de todos los métodos especificados en las interfaces.

DataAccess.Interface contiene las interfaces con los servicios que deben ser provistos para poder manejar el acceso a la base de datos. **DataAccess** contiene la implementación de la interfaz, tanto para utilizar una base de datos a SQL Server, como tanto en memoria (para facilitar la ejecución de las pruebas).

Domain es un paquete creado para alojar las clases que representan las entidades del problema de negocio. Dichas clases fueron separadas en su propio paquete ya que son utilizadas por varios paquetes.

WebApi es el proyecto donde se encuentra definida la API REST que será consumida por la aplicación Android. **WebApi** utiliza **BusinessLogic.Interface** para ejecutar todas las funcionalidades.

Se agregó los paquetes **GoogleAPI.Interface** y **GoogleAPI** para poder comunicarse con la API de Google Storage y Google Vision la cual usamos para poder realizar etiquetamiento de las fotos y alojar las fotos en el cloud.



8.2.1. BusinessLogic.Tests

AlbumLogicTests
Clase

Campos

- AlbumLogic
- context
- GoogleServices
- SessionLogic
- UserLogic

Métodos

- Cleanup
- CreateAlbum
- CreateAlbumWithExistentName
- CreateAlbumWithNoName
- CreateAlbumWithNoTags
- DeleteAlbum
- Initialize
- InitializeMemory
- InitializeSQL
- Photo
- User

SessionLogicTests
Clase

Campos

- context
- SessionLogic
- UserLogic

Métodos

- Cleanup
- Initialize
- InitializeMemory
- InitializeSQL
- SignIn
- SignInWithInvalidCredentials
- SignInWithInvalidEmail
- SignUp
- SignUpWithInvalidDate
- SignUpWithInvalidEmail
- SignUpWithInvalidName
- SignUpWithInvalidPasswordConfirmation
- SignUpWithInvalidPasswordWithoutNumbers
- SignUpWithInvalidPhone
- SignUpWithInvalidShortPassword
- User

PhotoLogicTests
Clase

Campos

- context
- GoogleServices
- PhotoLogic
- SessionLogic
- UserLogic

Métodos

- AddTag
- Cleanup
- DeletePhoto
- DeleteTag
- Initialize
- InitializeMemory
- InitializeSQL
- Photo
- SearchByTags
- User

UserLogicTests
Clase

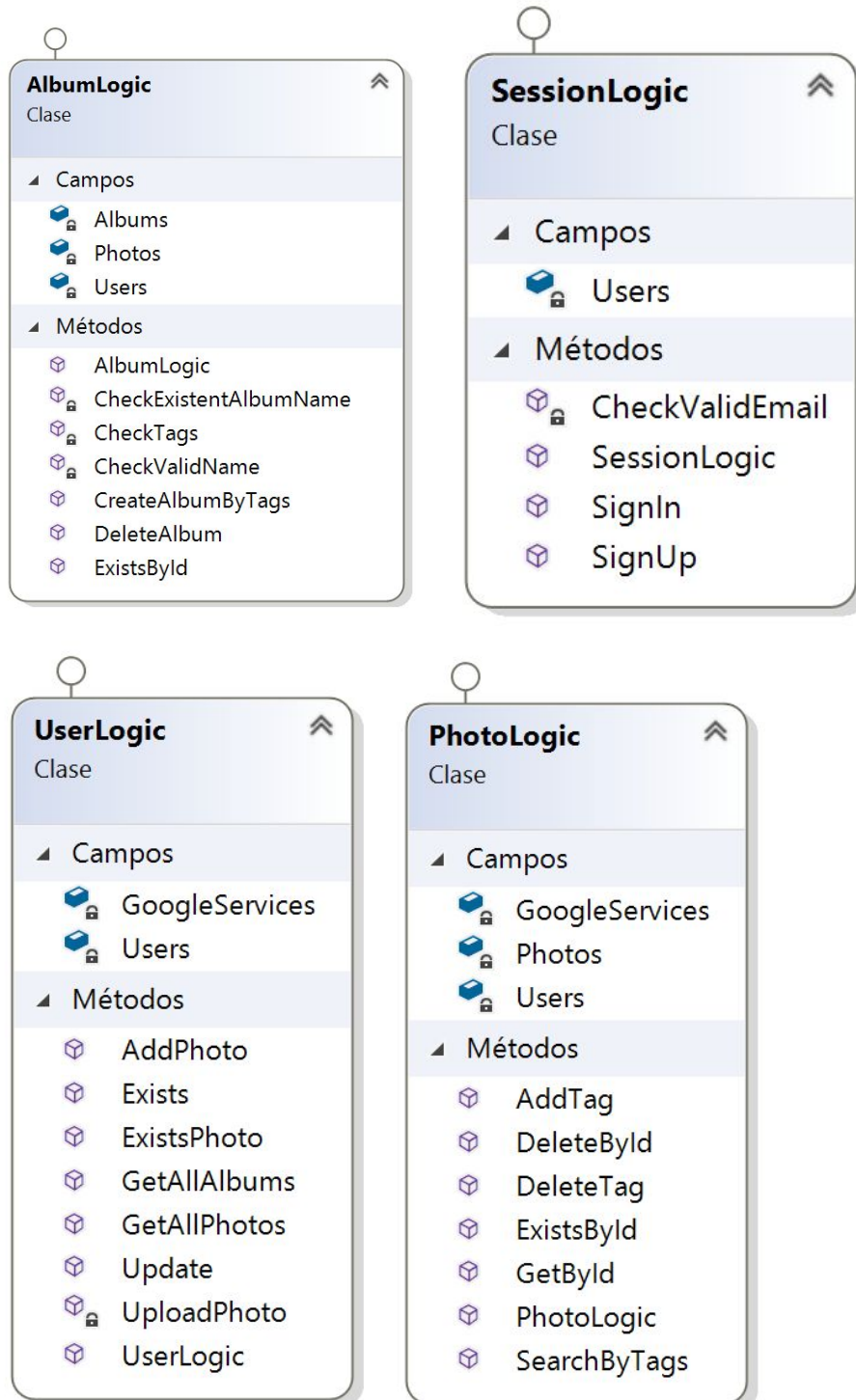
Campos

- AlbumLogic
- context
- GoogleServices
- SessionLogic
- UserLogic

Métodos

- Cleanup
- GetAllAlbums
- GetAllPhotos
- Initialize
- InitializeMemory
- InitializeSQL
- Photo
- Update
- UpdateWithInvalidDate
- UpdateWithInvalidName
- UpdateWithInvalidPasswordConfirmation
- UpdateWithInvalidPasswordWithoutNumbers
- UpdateWithInvalidPhone
- UpdateWithInvalidShortPassword
- UploadPhoto
- User

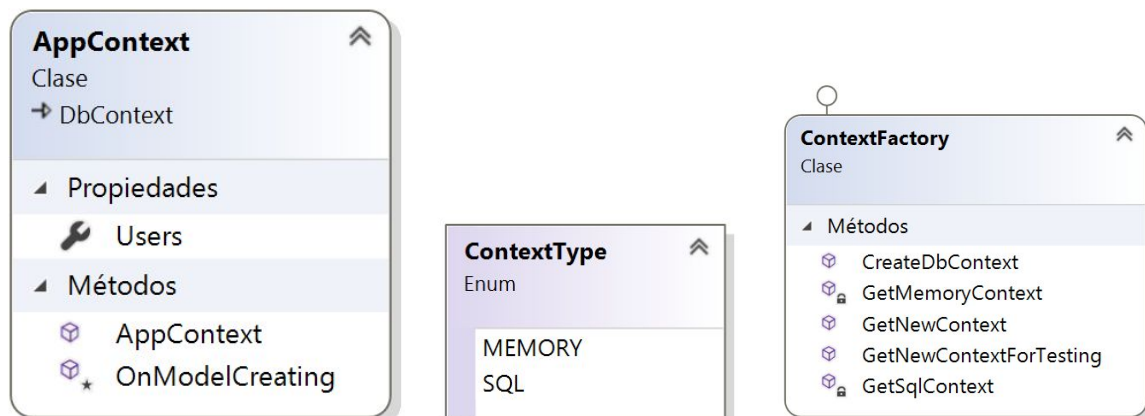
8.2.2. BusinessLogic

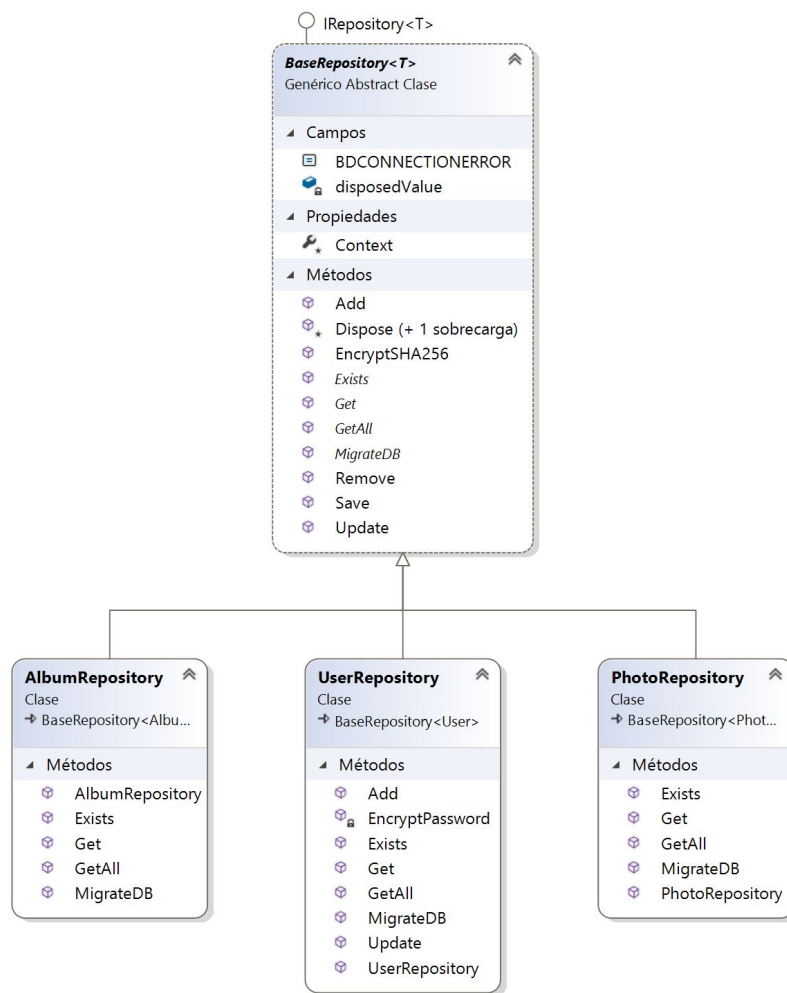


8.2.3. BusinessLogic.Interface

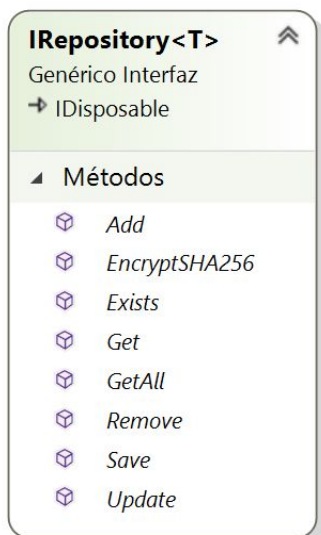


8.2.4. DataAccess

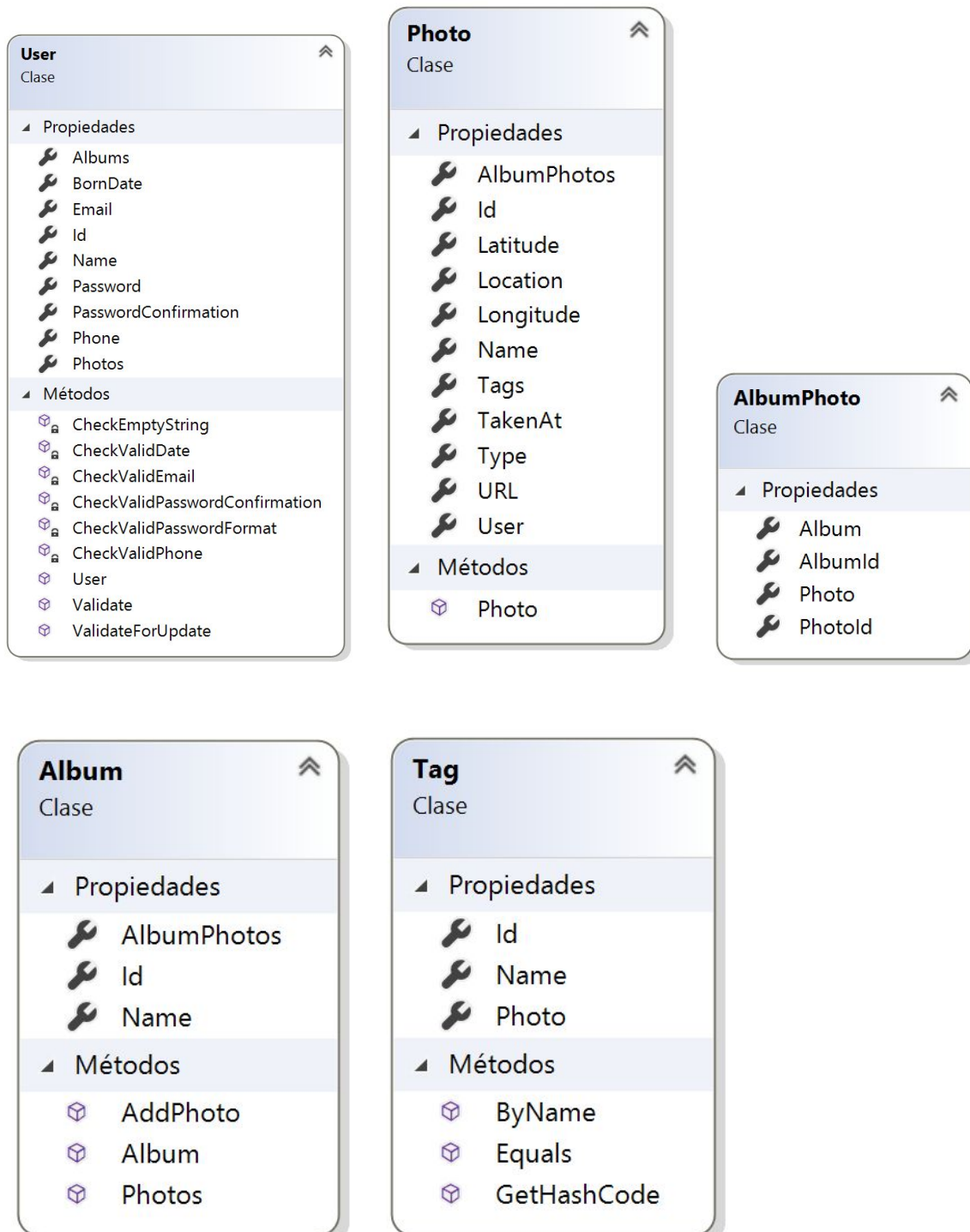




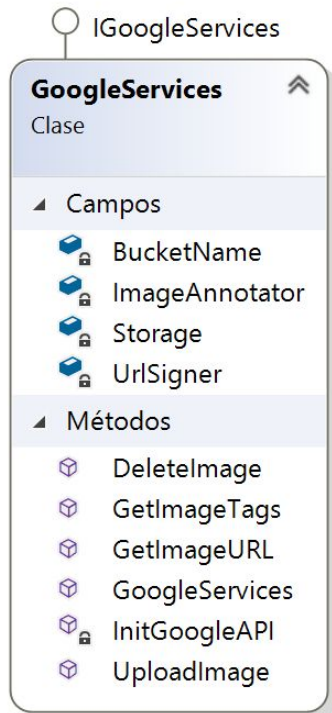
8.2.5. DataAccess.Interface



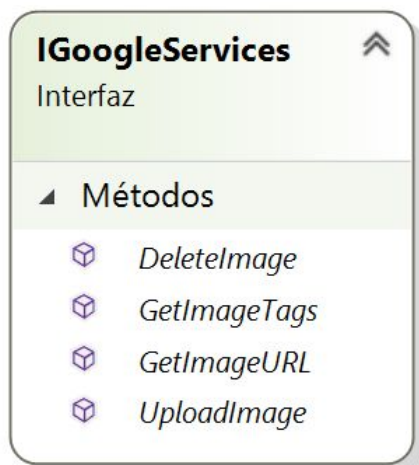
8.2.6. Domain



8.2.7. GoogleApi



8.2.8. GoogleApi.Interface



8.3. Diseño de Experiencia de Usuario (UXD)

Se especificará el proceso para mejorar la satisfacción del cliente, el compromiso de mejora de la usabilidad y la facilidad de uso del producto.

8.3.1. Estrategia y contenido

Análisis de la competencia

La competencia utiliza diseños y navegaciones muy similares. Los ejemplos serían Google Photos, Instagram y Photos. Vemos que se ha estandarizado dichos diseños y los clientes se han acostumbrado a los posicionamientos de los botones que se utilizan en dichas aplicaciones. Nuestra intención es brindar una experiencia de uso similar generando como valor agregado funcionalidades que nos destaquen de la competencia.

Análisis del cliente

El usuario que usará esta aplicación, va a estar familiarizado al uso de alguna aplicación líder del mercado como Google Photos, Instagram o Photos. Nuestra intención, es que para dichos usuarios, la transición a nuestra aplicación le sea fácil de adoptar y acostumbrarse. Por lo tanto, se optó por realizar un diseño, utilizando como base la ideal de diseño de dichas aplicaciones.

Sentimos que gran parte de nuestros usuarios, tendrán la intención de compartir sus álbumes y fotos con otras personas y a través de las redes sociales (Twitter), por lo tanto, sentimos fundamental poder brindarle a dichos usuarios las funcionalidades respectivas.

Se utilizará como estrategia la rápida adaptabilidad del cliente a nuestra aplicación y el rápido entendimiento que nuestra aplicación le genera un valor agregado y complementario a otras aplicaciones líderes en el mercado que él utiliza.

Desarrollo del contenido y estructura

El desarrollo del contenido estará sujeto a una estructura jerárquica de funcionalidades las cuales irán desde las funcionalidades macros, a funcionalidades micros que pertenezcan a sus subniveles.

No se realizará acceso a todas las funcionalidades desde el menú principal preferentemente para mantener una Interfaz de Usuario limpia y fácil de entender en todo momento.

A medida que el usuario vaya navegando y entrando a diferentes secciones, se le habilitarán las funcionalidades micros pertenecientes a dichas secciones.

8.4. Diseño de Interfaz de Usuario (UID)

Se especificará la investigación y el proceso de cómo interactúa nuestro producto con el usuario final.

8.4.1. Look and Feel

Investigación sobre el cliente

Creemos que es fundamental que el cliente se encuentre rápidamente familiarizado con la Interfaz de Usuario de nuestro sistema. El cliente está acostumbrado a un posicionamiento estándar de botones y menús, utilización de íconos y navegabilidad que ayudarán al usuario a sumergirse rápidamente en nuestra aplicación, ya que la misma contará con el uso de estos estándares de diseño.

Investigación sobre el diseño

La librería Material Design de Google, la cual es reconocida a nivel mundial como un estándar de diseño en el mercado, se decidió utilizar la misma para que nuestra aplicación cuente con los estándares de diseño contemporáneos. La misma proporciona componentes predefinidos, fáciles de adaptar y utilizar, con los formatos de medida adecuados.

Al ser una aplicación complementaria para el usuario, la cual le generará un valor agregado a otras aplicaciones similares que él ya utilice, se quiere que su adaptabilidad sea rápida, por lo que se asume como fundamental la utilización de íconos preestablecidos a nivel de mercado para identificar cada componente importante del diseño. Por ejemplo el uso del clásico botón de etiquetas, el ícono de localización en el mapa, el ícono de cámara para realizar una foto, entre otros (se especifican a total claridad en los prototipos de Interfaz de Usuario).

Implementación del diseño

La implementación será a nivel de la aplicación de Android, la cual usará los componentes de Material Design de Google versión 1.0.0.

Storyline

La aplicación está diseñada para que sea fácil de navegar e intuitiva. Se realizaron bocetos de Interfaz de Usuario en los cuales se puede ver que fácilmente se puede acceder a cada funcionalidad.

8.4.2. Capacidad de respuesta e Interactividad

Prototipación de la UI

Se realizaron prototipación de UI para cada User Story con la finalidad de validar las mismas con usuarios finales y dentro del equipo de desarrollo. La lista de los prototipos se pueden encontrar en el capítulo de Gestión del Alcance.

Interactividad y Animación

Se maximiza la interactividad con el usuario, se permite realizar operaciones de selección, borrado, inserción y modificación de objetos de manera fácil e interactiva. Se evita lo más posible, la utilización de menús para cada una de las operaciones, intentando que la cantidad de inputs que deba hacer un usuario sea la mínima posible.

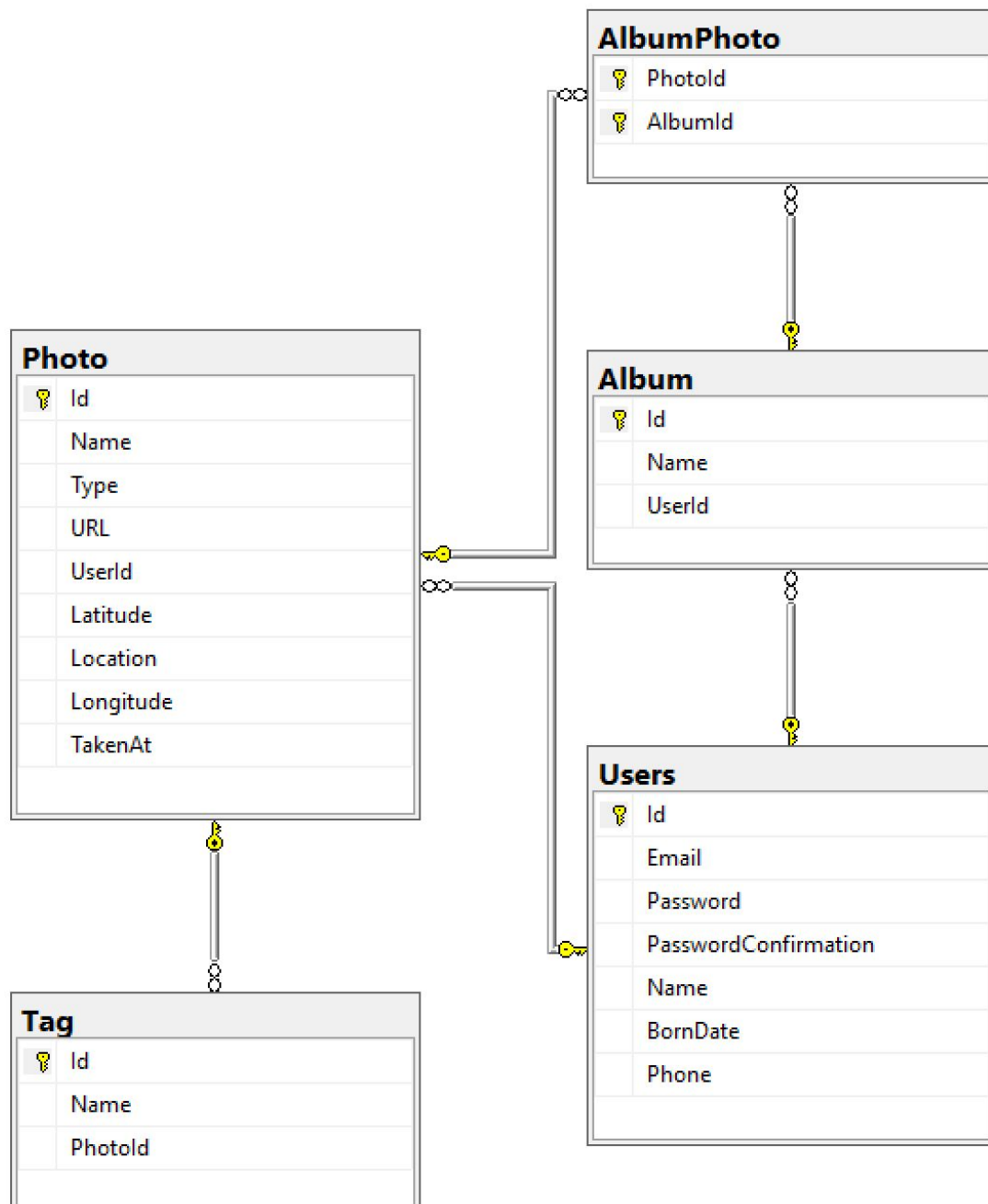
Se realizarán animaciones en los componentes utilizados en la Interfaz de Usuario. Los mismos serán provistos por la biblioteca de Material Design de Google.

Adaptación a los tamaños de las pantallas de los dispositivos

Se utilizan constraints y tamaños porcentuales, intentando evitar tamaños fijos para que la Interfaz de Usuario sea adaptable a los diferentes tamaños de pantalla del mercado.

Debido al tipo de aplicación que estamos desarrollando, no se permitirá la utilización de la aplicación de forma horizontal, ya que no sentimos que brinde una buena experiencia de usuario.

8.5. Modelo de Tablas de la Base de Datos



9. Instrucciones de uso

Configuración :

Se debe cambiar el ip (por defecto es 192.168.0.111:5000), al ip de la pc que tenga el backend. Esto se puede realizar desde el menú de propiedades del proyecto de WebApi en VisualStudio.

Se debe cambiar en el proyecto de Android, el ip del backend desde `com.app.RemoteDataSource.Uris`, campo `BASE_URL`.

Iniciar el backend corriendo el proyecto Obligatorio.

Iniciar la aplicación de Android.

Información operativa :

Eliminar una foto, álbum o etiqueta :

Para eliminar, se debe mantener presionado sobre la foto/album/etiqueta correspondiente y aparecerá un mensaje 'Deleted!' que confirma que se eliminó correctamente dicho elemento.

Buscar por tags :

Para buscar por tags, se debe ingresar con el siguiente formato: Tag1, Tag2, Tag3

Es importante conservar la mayúscula en el primer carácter, el resto en minúscula.

Si se desea buscar por más de 1 tag a la vez, se puede poner comas entre medio de ellos y el sistema buscará fotos que tengan un tag o el otro.

El mecanismo de búsqueda funciona como la unión de las fotos que tengan dichos tags, no la intersección.

Otros :

El resto de las funcionalidades, cumplen con el estándar habitual de las aplicaciones y consideramos que no es necesario especificarlas.

10. Bibliografía

[1] Desarrollo guiado por comportamiento

https://es.wikipedia.org/wiki/Desarrollo_guiado_por_comportamiento

[2] INVEST

[https://en.wikipedia.org/wiki/INVEST_\(mnemonic\)](https://en.wikipedia.org/wiki/INVEST_(mnemonic))

[3] Material Design

<https://material.io/design/>

<https://material.io/develop/android/>

[4] Test Driven Development

https://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas

[5] Android Developers

<https://developer.android.com/>

10. Anexos

10.1. Anexo 1

```
namespace BusinessLogic.Interface
{
    public interface ISessionLogic
    {
        User SignUp(User user);
        User SignIn(string email, string password);
        bool IsUserLoggedIn(string email);
    }
}
```

10.2. Anexo 2

```
public User SignIn(string email, string password)
{
    CheckValidEmail(email);
    bool exists = Users.GetAll().Exists(u => u.Email.Equals(email) &&
u.Password.Equals(password));

    if (exists)
    {
        return Users.GetAll().Find(u => u.Email.Equals(email) &&
u.Password.Equals(password));
    }
    else
    {
        throw new InvalidCredentials();
    }
}
```