

Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software en la Práctica Obligatorio 2

Diego Franggi
Federico Martínez
Juan Andrés Nervi

Docentes: Luis Barrague & Valentin Moscone

[Link al repositorio](#)

15 de junio de 2021

Declaraciones de autoría

Nosotros, Diego Franggi, Federico Martínez y Juan Andrés Nervi, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos la materia ingeniería de software en la practica
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Índice general

1. Alcance y Descripción del Proyecto	5
1.1. Análisis del problema	5
1.2. Product backlog	6
1.2.1. Método de Estimación	6
1.2.2. Acceder a la app	7
1.2.3. Generar contraseña segura	8
1.2.4. Guardar contraseña	8
1.2.5. Modificar contraseña guardada	9
1.2.6. Borrar contraseña	10
1.2.7. Analizar el estado de una cuenta de correo	10
1.2.8. Notificaciones push con el monitoreo de correo	11
1.2.9. Guardar notas seguras	11
1.2.10. Modificar notas seguras	12
1.2.11. Borrar notas seguras	13
1.2.12. Escaneo de contraseña mediante código QR	14
1.2.13. Exportar cuentas a un .zip	14
1.3. Historias que cambiaron	15
1.3.1. Exportar contraseñas a un .xlsx	15
1.3.2. Envío de contraseña a página web mediante código QR	16
1.3.3. Definición de hecho	16
2. Gestión del Proyecto	17
2.1. Planificación de Releases y Sprints	17
2.2. Evolución del proceso de desarrollo	18
2.2.1. Sprint 1	19
2.2.2. Sprint 2	19
2.2.3. Sprint 3	19
2.2.4. Sprint 4	19
2.2.5. Sprint 5	20
2.2.6. Sprint 6	20
2.2.7. Sprint 7	20
2.2.8. Sprint 8	21
2.2.9. Burndown Chart	21
2.3. Gestión de Riesgos	22
2.3.1. R01: Inexperiencia del equipo	22
2.3.2. R02: Enfermedad de un integrante	23

2.3.3.	R03: Corte de luz y/o servicio de Internet por tiempos prolongados	23
2.4.	Comentarios del proceso de desarrollo	23
2.5.	Gestión de la Configuración	24
2.6.	Aseguramiento de la Calidad	25
2.6.1.	Proceso y Producto	25
3.	Implementación de la Solución: Backend	27
3.1.	Dominio	29
3.2.	ILeaksApi & LeaksApiDummy	31
3.3.	SessionLogic	31
3.4.	Business Logic	31
3.5.	BusinessLogicInterface	34
3.6.	DataAccess	34
3.7.	IspWebApi	36
3.7.1.	WebApiServiceFactory	36
3.7.2.	IspWebApi.Models	36
3.7.3.	IspWebApi.Filters	36
3.7.4.	IspWebApi.Controllers	37
3.8.	Errores conocidos	40
4.	Implementación de la Solución: Frontend	41
4.1.	Paquetes de solución	41
4.1.1.	Organización de paquetes	42
4.2.	Arquitectura general	42
4.3.	Arquitectura UI	46
4.4.	Diseño de vistas	46
5.	Visualización de la Solución	49
5.1.	Primeros mockups	49
5.2.	Primera solución real	54
5.2.1.	Ingreso y registro	54
5.2.2.	Pantalla principal	55
5.2.3.	Contraseñas	56
5.2.4.	Notas	57
5.2.5.	Scan de qr	58
5.2.6.	Exportación de contraseñas	59
5.2.7.	Análisis de seguridad	60
6.	Tecnologías utilizadas	61
6.1.	Backend	61
6.2.	Frontend	61
6.3.	Proceso	62

7. Guía de instalación	63
7.1. Requisitos API	63
7.2. Modificando elementos del runtime	63
7.3. Corriendo la api por primera vez	64
7.4. Corriendo la aplicación desde el emulador	65
7.5. Notas finales	65
8. Anexo	66
Bibliografía	70

1. Alcance y Descripción del Proyecto

1.1. Análisis del problema

Hoy por hoy nos encontramos en un punto clave en la historia, la pandemia trajo de la mano miles de cambios en el mundo cibernético que están aquí para quedarse y no todos implican mejoras al sistema. Con las empresas recurriendo a migrar su modo de operación al mundo virtual, los ataques cibernéticos han aumentado considerablemente y el impacto de estos golpes puede afectar al ciudadano promedio de forma inesperada.

Quizás no podamos escudarnos de que nuestros datos sean liberados a los foros más oscuros de la web, pero si podemos proteger nuestra seguridad a lo largo de nuestra presencia en la web manteniendo una estricta política de contraseñas y accesos que pueda ser manejada y monitoreada por algún elemento externo.

Las contraseñas son robadas y liberadas todo el tiempo, hace unos meses Facebook recibió una filtración que dejó un total de 509 millones de contraseñas expuestas al mundo. Los usuarios normales estamos vulnerables a ataques de phishing que quizás nos engañen a liberar nuestra contraseña sin siquiera darnos cuenta y por nuestra naturaleza humana nuestra debilidad a la hora de elegir nuestras credenciales para autenticarse en muchos servicios deja mucho que desear al ser repetidas o fácilmente vulnerables.

Aprovechando este contexto de inseguridad optamos por la realización de un gestor de contraseñas, capaz de generar y almacenar contraseñas altamente seguras que eviten nuestra vulnerabilidad a los llamados ataques de “credential stuffing”. El sistema será altamente flexible al permitir el almacenado de notas seguras y la exportación de la lista de contraseñas a un archivo Zip asegurado con contraseña.

Además, presentará un sistema que permita monitorear la Dark Web en busca de alguna de nuestras credenciales en el caso de ser vulneradas, con ayuda de la api del popular sitio HaveIBeenPwned¹. Sumado a esto la funcionalidad insignia de

¹Simulado por una aplicacion creada por nosotros

nuestra aplicación es la que permite la obtencion de una contraseña mediante un código QR en la web para permitir el traslado rápido de contraseñas generadas en la web a nuestro celular. Todo presentando seguridad avanzada para cifrar aquellas contraseñas almacenadas y protegerlas tras una contraseña u el uso de la huella digital.

1.2. Product backlog

1.2.1. Método de Estimación

La técnica de estimación utilizada fue **planning poker**. Definimos como unidad de estimación las story points, las cartas configuradas seguían la secuencia de Fibonacci con los comodines de infinito (siendo una estimación imposible que requiera que separemos la user story en otras más pequeñas). Para la realización de la estimación se fue leyendo carta por carta del backlog y efectuando los pasos propios de la técnica hasta llegar a un acuerdo.

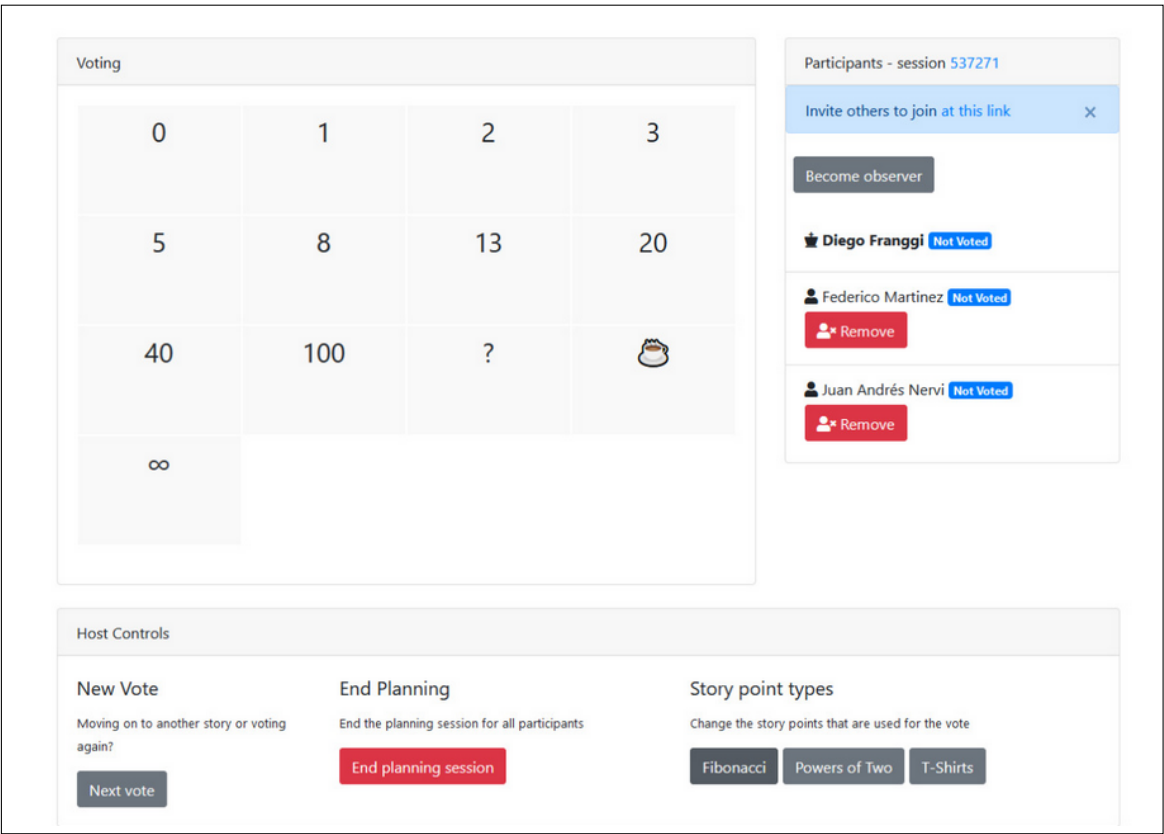


Figura 1.1: Votación en planning poker

1.2.2. Acceder a la app

Prioridad

Alta

Estimado

8 SP

User story

Como usuario no autenticado Quiero poder acceder a las listas seguras con la huella digital o con contraseña Para poder ahorrar tiempo

Criterios de aceptación

- DADO un usuario sin autenticar CUANDO desea entrar a la app y selecciona ingresar con huella ENTONCES se le ofrece una opción de entrar con la huella dactilar
- DADO un usuario sin autenticar CUANDO ingresa su huella ENTONCES se verifica que sea realmente su huella
- DADO un usuario autenticándose CUANDO su huella es incorrecta ENTONCES se le avisara al usuario que le quedan 2 intentos antes de bloquear la app temporalmente por 5 minutos
- DADO un usuario sin autenticar CUANDO desea entrar a la app y selecciona ingresar con contraseña ENTONCES se le ofrece una opción de entrar con contraseña
- DADO un usuario autenticándose CUANDO su contraseña es incorrecta ENTONCES se le avisa al usuario que le quedan 2 intentos antes de bloquear la app temporalmente por 5 minutos
- DADO un usuario sin autenticar CUANDO ingresa su contraseña ENTONCES se verifica que sea realmente su contraseña

1.2.3. Generar contraseña segura

Prioridad

Alta

Estimado

2 SP

User story

Como usuario Quiero poder generar una contraseña segura Para que sea difícil de romper en caso de una filtración de una base de datos y no tener contraseñas repetidas

Criterios de aceptación

- DADA un usuario accediendo a la función CUANDO oprimo un botón ENTONCES se genera un string de texto de largo mayor a 10, que contenga caracteres aleatorios y no sea una palabra de diccionario, además de algún símbolo, numero y carácter especial

1.2.4. Guardar contraseña

Prioridad

Alta

Estimado

2 SP

User story

Como usuario Quiero poder guardar una contraseña Para tenerla segura en mi teléfono

Criterios de aceptación

- DADO un usuario autenticado que guarda su contraseña en la app CUANDO va a ver sus contraseñas ENTONCES la contraseña que guardó se ve en la lista
- DADA una contraseña guardada CUANDO se intenta acceder a la misma del filesystem ENTONCES se encuentra la contraseña cifrada
- DADA una contraseña guardada CUANDO intento acceder a la contraseña guardada ENTONCES debo autenticarme antes de ver la lista

1.2.5. Modificar contraseña guardada

Prioridad

Alta

Estimado

1 SP

User story

Como usuario Quiero poder modificar mi contraseña Para cambiar mi clave en caso de ser necesario

Criterios de aceptación

- DADO un usuario autorizado que modifica su contraseña en la app CUANDO va a ver sus contraseña modificada ENTONCES la contraseña que modificó y guardó se ve en la lista.
- DADO un usuario que modifica su contraseña en la app CUANDO modifica una contraseña ENTONCES su contraseña queda modificada en toda la aplicación.
- DADO un usuario que modifica su contraseña en la app CUANDO va a realizar la modificación debe estar autenticado y autorizado
- DADO un usuario que modifica su contraseña en la app CUANDO modifica una contraseña ENTONCES la contraseña que modificó es efectivamente la que quiso modificar.

1.2.6. Borrar contraseña

Prioridad

Alta

Estimado

1 SP

User story

Como usuario autenticado Quiero poder borrar una contraseña guardada Para removerla de la lista de contraseñas

Criterios de aceptación

- DADO un usuario autenticado que borra un usuario en la app CUANDO va a ver contraseñas ENTONCES la contraseña que borro se eliminó de la lista
- DADO un usuario que quiera borrar una contraseña en particular en la app CUANDO es seleccionada dicha contraseña y arrastrada en direccion horizontal ENTONCES se elimina la contraseña de la lista

1.2.7. Analizar el estado de una cuenta de correo

Prioridad

Alta

Estimado

5 SP

User story

Como usuario autenticado Quiero poder verificar el estado de una dirección de correo Para saber si fue comprometida en alguna filtración de bases de datos

Criterios de aceptación

- DADO un usuario de la app CUANDO este selecciona el botón 'Verificar seguridad de la cuenta' ENTONCES la app se conecta con la API de HaveIBeenPowned y verifica si la cuenta fue filtrada
- DADO un usuario seleccionando al verificar la seguridad de cuenta CUANDO se encuentra una filtración ENTONCES se alerta al usuario

1.2.8. Notificaciones push con el monitoreo de correo

Prioridad

Alta

Estimado

5 SP

User story

Como usuario de la aplicación Quiero saber si mi correo fue comprometido Para saber si he de tomar acciones sobre mis contraseñas

Criterios de aceptación

- DADA una dirección de correo del usuario CUANDO la misma aparece en una filtración conocida ENTONCES se envía una notificación al usuario

1.2.9. Guardar notas seguras

Prioridad

Media

Estimado

2 SP

User story

Como usuario autenticado Quiero poder guardar una nota segura Para poder tener notas privadas en un lugar seguro

Criterios de aceptación

- DADO un usuario autenticado CUANDO guarda una nota ENTONCES esta queda guardada en el sistema y solo es visible por ese usuario
- DADO un usuario que quiere hacer una nota CUANDO guarda una nota repetida (mismo título) ENTONCES el sistema avisa que la nota ya fue creada.
- DADA una nota guardada CUANDO un usuario autenticado distinto al creador quiere entrar ENTONCES es inaccesible por el usuario no creador.
- DADO un usuario no autenticado CUANDO intenta acceder a una nota segura ENTONCES se le avisa que no esta autenticado

1.2.10. Modificar notas seguras

Prioridad

Media

Estimado

1 SP

User story

Como usuario autenticado Quiero poder modificar mis notas Para corregir errores o modificar mis notas ya creadas.

Criterios de aceptación

- DADO una nota segura CUANDO es modificada ENTONCES se modifica la nota que se quería modificar y queda guardada en el sistema.
- DADO una nota segura CUANDO la nota no es modificada ENTONCES la nota se guarda igualmente sin cambios.
- DADO una nota segura que ya fue modificada CUANDO la nota se vuelve a modificar ENTONCES se modifica la nota que se quería modificar y queda guardada en el sistema

1.2.11. Borrar notas seguras

Prioridad

Media

Estimado

1 SP

User story

Como usuario autenticado Quiero poder borrar una nota segura ya creada Para poder eliminar las notas que no deseo guardar más.

Criterios de aceptación

- DADO un usuario autenticado borrando una nota segura CUANDO oprime borrar la nota ENTONCES se elimina del sistema
- DADO una nota segura borrada CUANDO quiero acceder a ella ENTONCES el sistema no me deja porque ya fue eliminada.
- DADO un usuario CUANDO quiere crear una nota segura igual a una que fue eliminada anteriormente ENTONCES el sistema permite crear una nota igual

Criterios de aceptación

- DADO un usuario de la app CUANDO este selecciona la opción 'Exportar contraseñas' ENTONCES la app exporta todas las contraseñas de la lista y protege el archivo con la contraseña de acceso a la app

1.2.12. Escaneo de contraseña mediante código QR

Prioridad

Media

Estimado

8 SP

User story

Como usuario autenticado Quiero poder escanear un código QR Para tomar del código QR la información y convertirlo en la contraseña de un sitio específico que el usuario deberá aportar con su nombre.

Criterios de aceptación

- DADO un usuario autenticado CUANDO escaneo un código QR ENTONCES puedo tomar la información y convertirla en la contraseña del sitio a agregar.
- DADA una contraseña escaneada por el código QR CUANDO se guarda la contraseña ENTONCES no puedo verla en texto plano
- DADA una contraseña escaneada por el código QR CUANDO se guarda la contraseña ENTONCES se almacena en la base y no puedo verla en texto plano.

1.2.13. Exportar cuentas a un .zip

Prioridad

Baja

Estimado

13 SP

User story

Como usuario Quiero poder exportar mis cuentas Para poder acceder a ellas desde .zip conteniendo mis cuentas con su correspondiente contraseña

Criterios de aceptación

- DADO un usuario de la app CUANDO este selecciona la opción *Exportar contraseñas* ENTONCES la app exporta todas las cuentas de la lista conteniendo una fila por cada cuenta con los siguientes datos: e-mail, sitio y contraseña. El archivo es protegido por contraseña.

1.3. Historias que cambiaron

A medida que se avanzó en el proceso de desarrollo las siguientes historias fueron modificadas por las presentadas arriba, sin embargo se dejan como registro historico de que se presentaron en la primer entrega.

1.3.1. Exportar contraseñas a un .xlsx

Prioridad

Baja

Estimado

13 SP

User story

Como usuario Quiero poder exportar mis contraseñas Para poder acceder a ellas desde un Excel

Criterios de aceptación

- DADO un usuario de la app CUANDO este selecciona la opción *Exportar contraseñas* ENTONCES la app exporta todas las contraseñas de la lista y protege el archivo con la contraseña de acceso a la app

1.3.2. Envío de contraseña a página web mediante código QR

Prioridad

Media

Estimado

20 SP

User story

Como usuario autenticado Quiero poder acceder a una página web donde enviar una contraseña en un navegador web Para acceder a la misma de forma rápida en cualquier ordenador

Criterios de aceptación

- DADO un usuario autenticado CUANDO accedo a un sitio específico de la aplicación ENTONCES puedo enviar mi contraseña vía QR a otro dispositivo
- DADA una contraseña enviada mediante el QR CUANDO recibe la contraseña ENTONCES no puedo verla en texto plano
- DADA una contraseña recibida CUANDO la copio y pego en otro lado ENTONCES es la misma contraseña que envié desde la aplicación

1.3.3. Definición de hecho

Una user story esta completa únicamente al completarse todos sus criterios de aceptación.

2. Gestión del Proyecto

Para llevar a cabo el proyecto, se decidió utilizar como metodología ágil al framework SCRUM. Este framework de gestión de proyectos se caracteriza por hacer énfasis en el trabajo en equipo, mediante la repetición de iteraciones o sprints de un largo determinado por el SCRUM Master para alcanzar un objetivo accesible a corto plazo. Al finalizar una iteración, se obtiene un entregable que va a estar compuesto de las User Stories terminadas. Un conjunto de iteraciones componen un release y al finalizar el mismo se espera tener con un MVP (Mínimo Producto Viable).

Este framework establece ciertas reuniones a lo largo de los sprints. Las mismas son: Sprint Planning Meeting, Daily Planning Meeting, Sprint Review y por último Sprint Retrospective. En las mismas se hablan de definir un objetivo para el sprint, a evaluar el progreso del equipo, a recibir feedback de cómo se está llevando a cabo el desarrollo del proyecto, y a mejorar el proceso y evolución del equipo respectivamente.

Párrafo aparte, merece la pena mencionar que al estar en tiempos de pandemia, serán realizadas tres reuniones semanales mediante vídeo llamada con motivo de aclarar dudas y realizar un seguimiento de los objetivos alcanzados por el equipo, aparte de las reuniones pre establecidas por la metodología SCRUM utilizada.

2.1. Planificación de Releases y Sprints

Para llevar a cabo la planificación de los releases del proyecto, se estableció realizar el enfoque Date-Driven. Dado que la fecha para la entrega del mismo está definida para el 16/06/2021 y a partir de la fecha de entrega del primer documento (21 de Abril de 2021), se tendrá un total de 56 días (8 semanas) para la realización de la documentación y el proyecto en su totalidad. Se considera entonces el próximo plan de releases:

1. Primer Release (1.0): pautado para la fecha 19/05/2021 (4 semanas de duración). Para esta fecha se esperaba tener una versión preliminar del producto

final. Se esperaba poder entregar un MVP de forma que la aplicación tenga las funcionalidades mínimas categorizadas con prioridad alta. Este release habría estado constituido por 4 sprints de una semana de duración cada uno. Por decisión del equipo que será comentada a continuación, se desestimó dicho release.

2. Segundo Release (2.0): pautado para la fecha 16/06/2021 (4 semanas de duración). Para esta fecha se esperaba tener el proyecto culminado. Se espera proceder con las tarjetas de prioridad media y baja y realizar la documentación del mismo. Este release será constituido por 4 sprints de una semana de duración cada uno.

En cuanto a los sprints del proyecto, se decidió utilizar el enfoque de velocity-driven. Basado en las estimaciones, el total de story points para nuestro proyecto es de 49 SP, por lo que se espera poder completar cerca de 6 SP por sprint. De todas formas se analizará en la sprint review del primer sprint la velocidad acordada para ver si se estimó correctamente. En caso de que la velocidad del equipo sea menor, habrá que dedicarle más horas por parte del equipo para hacer frente al retraso y poder cumplir con los plazos. De no ser así, se continuará con la velocidad establecida.

2.2. Evolución del proceso de desarrollo

Si bien en nuestra primera planificación habíamos decidido primero realizar un release con las historias de usuario que sean de prioridad Alta, posteriormente el equipo tomó la decisión de avanzar sobre todo en la solución del Backend para tener una base sólida donde construir con posterioridad el FrontEnd. Eso no quiere decir que no se haya avanzado en paralelo respecto al FrontEnd desde el principio.

Si bien nos decidimos por un release solo, tratamos de mantener los sprints planificados en cuanto al contenido y cantidad. Se mantuvieron los 8 sprints acordados. Los 4 sprints relacionados al release 1.0 fueron reubicados en el release 2.0.

A medida que fuimos avanzando e interiorizando con las librerías de Android nos dimos cuenta que la tarjeta de envío de contraseña a página web mediante código QR iba a consumir más tiempo de lo planificado debido a la complejidad de la implementación y al manejo del emulador del propio Android Studio. Fue por eso que para no eliminar la card, se decidió implementar la lectura de un QR pero limitando su funcionalidad a la lectura y posterior almacenamiento en la base de forma encriptada.

2.2.1. Sprint 1

Semana del 26/4

- Acceder a la App: 8SP

Se comienza a implementar el dominio de la solución, la creación de la base de datos y el comienzo del inicio de sesión en FrontEnd.

2.2.2. Sprint 2

Semana del 3/5

- Guardar Notas Seguras: 2SP
- Exportar contraseñas a un .zip: 13SP

Se termina de implementar el dominio de la solución. Se comienza a implementar la API y la lógica de algunos componentes y el comprimido de las cuentas personales con su respectiva contraseña. Posteriormente se envía en base64 el archivo .zip protegido por contraseña, que en este caso y a modo de una primera versión, la contraseña será el mail del usuario que usó cuando se registró que se almacenará en el teléfono. Anteriormente la card del export de contraseñas se iba a realizar a un .xlsx pero se mejoró la misma comprimiendo un .txt con la información y protegiéndolo en un .zip. En el apartado de FrontEnd se diseño la HomeActivity.

2.2.3. Sprint 3

Semana del 10/5

- Borrado de notas seguras: 1SP

Se termina de implementar el inicio de sesión mediante el mail y se da a elegir al usuario si desea logearse mediante su password o si desea ingresar su huella dactilar. Se comienza a integrar la solución con el servicio Web de Firebase Messaging Cloud.

2.2.4. Sprint 4

Semana del 17/5

- Analizar el estado de una cuenta de correo: 5SP

Se realiza un refactor del Login, inicio de sesión y la HomeActivity de Kotlin a Android, se decide implementar el resto del FrontEnd en Java debido a que el conocimiento sobre dicho lenguaje es mayor. Se definen los endpoints y se implementa el analizador de cuentas filtradas.

2.2.5. Sprint 5

Semana del 24/5

- Guardar contraseña 2SP
- Borrar contraseña 1SP
- Modificar contraseña guardada 1SP

Se identifican algunos bugs, se realiza un refactor y se resuelven los mismos. Se comienza a implementar la API de Leaks simulando la API "Have Been Pwned". No estaba en los planes implementar la API leaks, pero dado a que la API 'Have Been Pwned' es paga fue necesario adaptarnos a ese contratiempo. Por último, se implemento la lógica de dicha API y algunas user stories de baja demanda de horas.

2.2.6. Sprint 6

Semana del 31/5

- Modificar notas seguras: 1SP
- Generar contraseña segura: 2SP

Se implementa las notas en FrontEnd y se generan contraseñas seguras randomicas. También se implementó botones de back en la aplicación y se agregó la lógica de cerrar sesión.

2.2.7. Sprint 7

Semana del 7/6

- Notificaciones push con el monitorio de correo: 5SP

- Escaneo de contraseña mediante QR: 8SP

Se implementa la lógica del envío de notificaciones push mediante el token de firebase único para cada dispositivo almacenado al momento de realizar el registro del cliente, las notificaciones se visualizan tanto en primer plano como en segundo plano. También se realizó la variante de la card inicial explicada anteriormente para el escaneo de una contraseña mediante un código QR y el guardado en la base.

2.2.8. Sprint 8

Semana del 14/6

Se realiza un refactor general de la aplicación tanto en Backend como en FrontEnd y se realiza una prueba general de la misma, nuevamente se identifican nuevos bugs y solucionan. Se utiliza este último sprint para documentar de igual manera.

2.2.9. Burndown Chart

La tabla Burndown Chart mide el progreso del proyecto a medida que pasa el tiempo. Dejándonos saber cuanto trabajo y cuanto tiempo nos queda por realizar, trata de pronosticar cuando finalizará el mismo permitiendo monitorear el estado.

La burndown chart se compone de dos elementos principalmente, una línea de trabajo ideal o línea de trabajo restante que es la línea que conecta el punto inicial del proyecto con el punto final que debería ser la intersección con el eje de las x mostrando que no queda más trabajo por hacer; y por otro lado la línea del trabajo actual remanente, que marca los desvíos de la línea ideal.

Para realizar la misma se fijó el calendario de trabajo correspondiente a los 8 sprints anteriormente descritos y el estimado de story points a cubrir una vez finalizado y el trabajo actual.

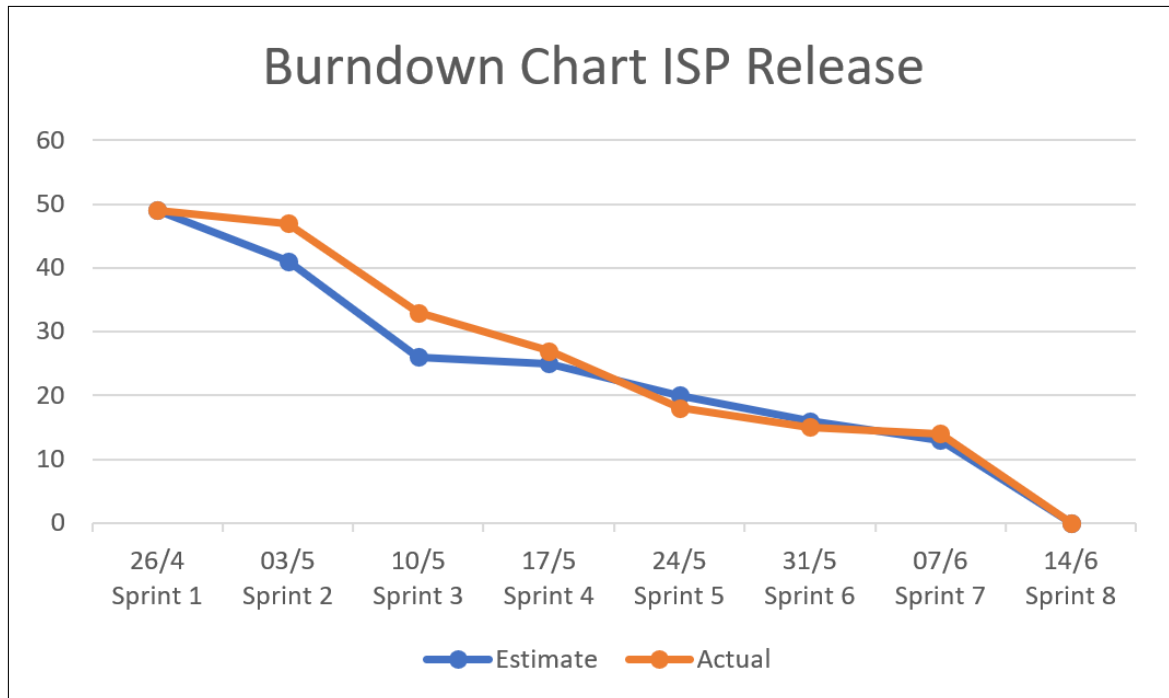


Figura 2.1: Burndown Chart Release 1.0

Como se puede apreciar en el gráfico, si observamos el intervalo comprendido entre el Sprint 6 y el Sprint 7 (línea anaranjada) nos estancamos de cierto modo, dado que presentamos problemas con card de la captura del QR y también presentamos ciertas dificultades con el token de firebase y las notificaciones en primer plano.

2.3. Gestión de Riesgos

Los posibles riesgos que pudimos identificar que podrían jugar en contra en nuestro proyecto podrían ser los siguientes:

2.3.1. R01: Inexperiencia del equipo

Descripción: Dado a que ninguno de los integrantes del equipo tiene experiencia desarrollando en Kotlin para el desarrollo de software mobile (Android), existe la posibilidad de que la curva de aprendizaje sea demasiado grande y afecte al desarrollo del proyecto. Se puede dar si las estimaciones de trabajo planteadas se encuentran muy por debajo del trabajo real.

- Probabilidad de ocurrencia: Posible (0,5 sobre 1)
- Nivel de Riesgo: Medio

2.3.2. R02: Enfermedad de un integrante

Descripción: En la presente situación sanitaria que atraviesa el país diremos que si uno de nuestros integrantes del equipo se enfermase de Covid-19 u otra enfermedad que impida a la persona de trabajar en el proyecto, recaería el trabajo en los demás integrantes o en su defecto se reflejaría en una baja en la velocidad del equipo.

- Probabilidad de ocurrencia: Improbable (0,3 sobre 1)
- Nivel de Riesgo: Alto

2.3.3. R03: Corte de luz y/o servicio de Internet por tiempos prolongados

Descripción: Existe la posibilidad de que algún integrante del equipo se vea afectado de manera prolongada en sus actividades por falta de energía eléctrica o servicio de Internet en su domicilio. Dada la demanda de personas en los hogares con problemas de conectividad, podría repercutir en la demora de la reconexión al servicio en cuestión generando atrasos en el equipo.

- Probabilidad de ocurrencia: Improbable (0,1 sobre 1)
- Nivel de Riesgo: Medio

2.4. Comentarios del proceso de desarrollo

Con respecto de los riesgos, efectivamente ocurrió el riesgo identificado RO1: *Inexperiencia del Equipo* debido a que se cumplió la premisa de que la falta de experiencia en Kotlin podría retrasarnos, para mitigar cierto riesgo optamos por desarrollar en Java donde el equipo se sentía más seguro.

También podemos observar que la línea de trabajo ideal tendría que tender a ser lo más *lineal* posible, valga la redundancia, sin caídas bruscas como se planificó en el Sprint 2, se trató de abarcar demasiadas story points que el equipo no pudo cumplir. Con el correr de los siguientes sprints creemos que en general se efectuó un buen trabajo y se logró el objetivo adaptándonos a las circunstancias y a los problemas.

2.5. Gestión de la Configuración

En cuanto a el control de cambios y de versiones, para poder tener un registro se utilizó un repositorio en Github. En este repositorio se elaboró todo el código referente al proyecto (backend y desarrollo en android). Esto permitió que todos los miembros del equipo trabajen en simultáneo en el proyecto, así de esta manera mejorando el flujo y acelerando el desarrollo.

En Github sera utilizado el flujo de trabajo Git-Flow, el cual nos proporciona ventajas a la hora de la gestión de las funcionalidades, y promueve la entrega e integración continua que se da en el desarrollo con metodologías ágiles, de una manera más organizada.

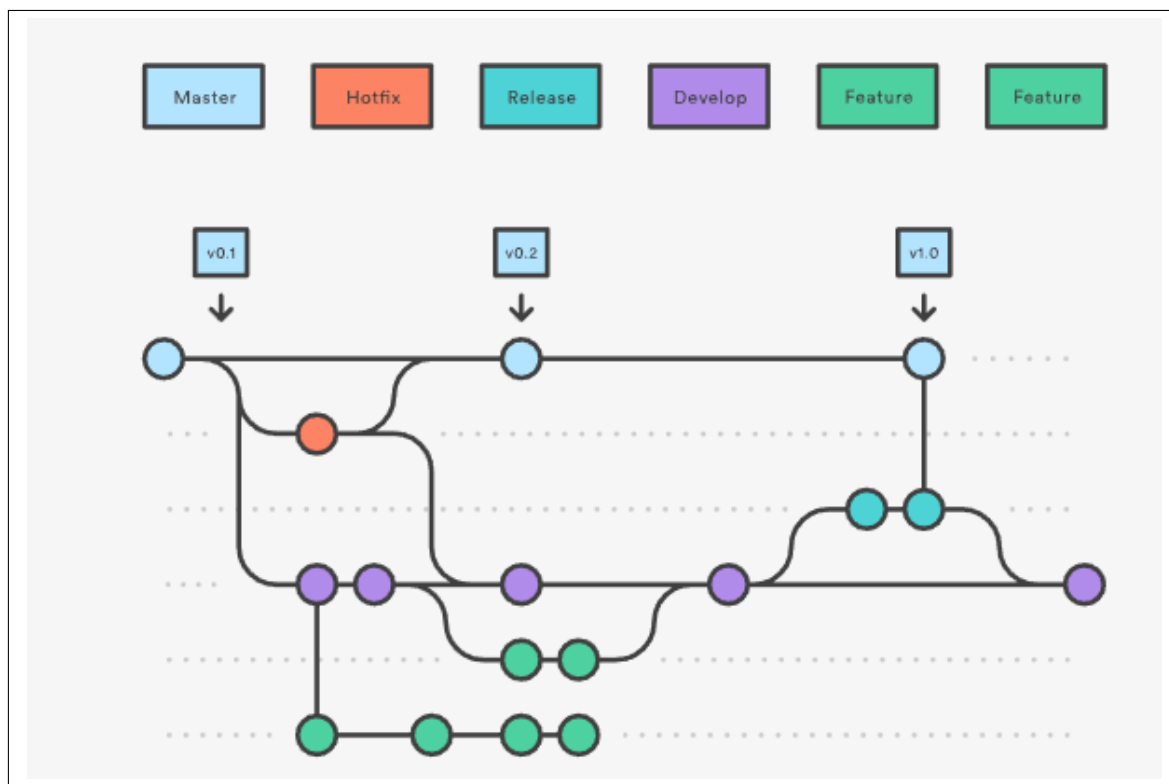


Figura 2.2: Imagen Gitflow

Dentro del repositorio se encuentran las carpetas de Aplicación (con el código del backend compilado en release listo para ejecutar y el APK del android), BaseCode (con el código del front y backend), Database (con los dos .bak solicitados) y Postman Test (con algunos casos de prueba para la webapi).

2.6. Aseguramiento de la Calidad

2.6.1. Proceso y Producto

Metodología de Desarrollo

Para el desarrollo de la aplicación, se decidió utilizar la metodología BDD (Behavior Driven Development), esta misma consiste en definir los requisitos del sistema como historias de usuario con escenarios específicos, estos mismo serán utilizados como casos de prueba para comprobar que las funcionalidades han sido incorporadas exitosamente (ejemplos de esto se pueden observar en el Product Backlog).

Esta metodología ofrece una manera sencilla para corroborar cuando una funcionalidad se encuentra finalizada (se deben cumplir todos los escenarios planteados en las historias de usuario para que esto sea posible). Además de esto, también delimita explícitamente el alcance esperado para cada requisito, esto ayuda a evitar trabajar en funcionalidades extra que pueden surgir en el momento de realización de la aplicación.

Identificación y Uso de Estándares

En cuanto a la codificación del código de la aplicación se seguirán los principios de Clean Code [?], tales como realizar clases y métodos lo más cortos posibles (utilizando el método de dividir para conquistar en el que los problemas se separan con tal de generar un código más limpio y legible), utilizar nombres mnemotécnicos para variables y escribir la menor cantidad de comentarios siendo estos descriptibles y relevantes (se debe generar un código que sea legible y no se necesite indicar que está sucediendo) entre varias prácticas.

Se intentará aplicar TDD para probar el funcionamiento del código del backend, algo que recomienda Robert Martin en su libro *Clean Code: A Handbook of Agile Software Craftsmanship* [2], entre otros consejos que se encuentran en el mismo. Al manejar las recomendaciones de clean code podremos desarrollar un código que sea legible, ampliable y fácil de mantener a futuro.

A la hora de diseñar la solución se hará uso de los patrones de diseño GRASP, y de la aplicación de los principios SOLID. Utilizando estos podremos aportar a nuestro plan de generar un código mantenible, reusable y escalable.

Al momento de realizar la interfaz de usuario se respetarán las heurísticas de Nilsen para evitar problemas de usabilidad y brindar la mejor experiencia de usuario posible.

Para verificar el cumplimiento acorde del uso de las metodologías optamos por

apoyarnos en el siguiente [checklist](#) previo a realizar un merge a la rama develop.

3. Implementación de la Solución: Backend

A continuación, se comienza a describir la implementación del backend de la solución. El mismo es compuesto por una WebApi encargada de facilitar operaciones relacionadas a la exportación de contraseñas en formato ZIP, así como el monitoreo de correos en busca de filtraciones conocidas en la *Dark Web*.

Para realizar un estudio comprensible haremos una recorrida a través de las distintas capas en la que estructuramos nuestra solución. Comenzamos con una capa de acceso web donde encontramos la mencionada API en conjunto a sus paquetes internos como lo son los modelos, controllers y filtros (sobre los que se detallará más adelante), capas de interfaces medias que funcionan de limite entre las capas web y lógicas, para luego llegar a la capa de acceso a datos.

Se destaca el funcionamiento de una *dummy api para leaks* dentro de nuestro sistema. Con el propósito de ser una prueba de concepto esta versión de la aplicación no utiliza sistemas reales para el monitoreo de correos filtrados. Como sustituto el sistema implementa un mecanismo de agregado de cuentas filtradas llamado **DummyLeaksApi** el cual una vez suministrado un esquema de filtrado simula el comportamiento de una api real como lo puede ser HaveIBeenPwned.

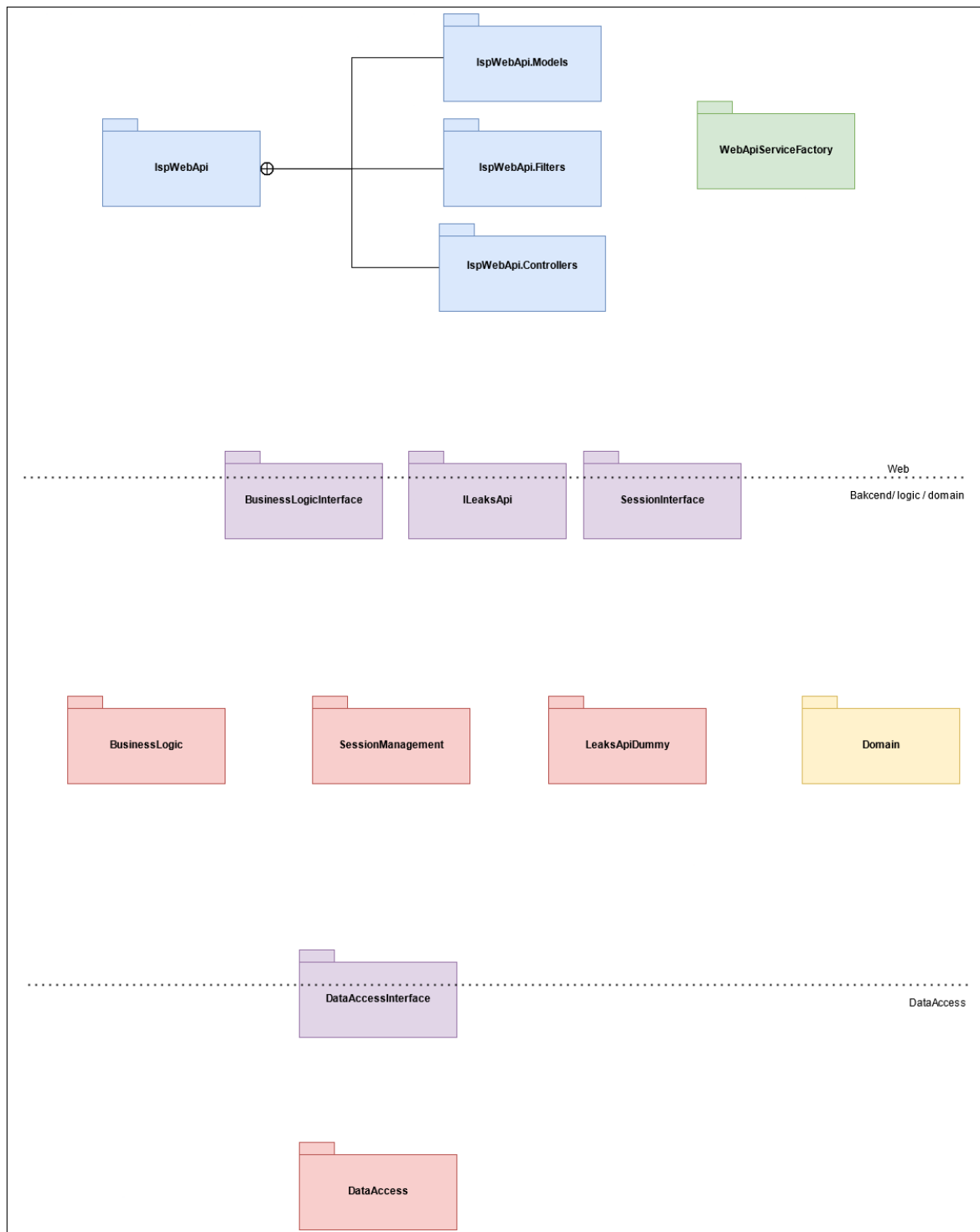


Figura 3.1: Diagrama de descomposición en namespaces y en capas

Luego de tener una visión de la estructura en capas de la solución podemos pasar a ver como se utilizan mutuamente los paquetes que componen la aplicación. Vemos como los módulos superiores no dependen directamente de los inferiores sino de paquetes con abstracciones bien definidas, cumpliendo el principio de inversión de dependencias. Se busca que las capas solamente se comuniquen con las (interfaces)

de capas inferiores a ellas, con la excepción del dominio de la solución.

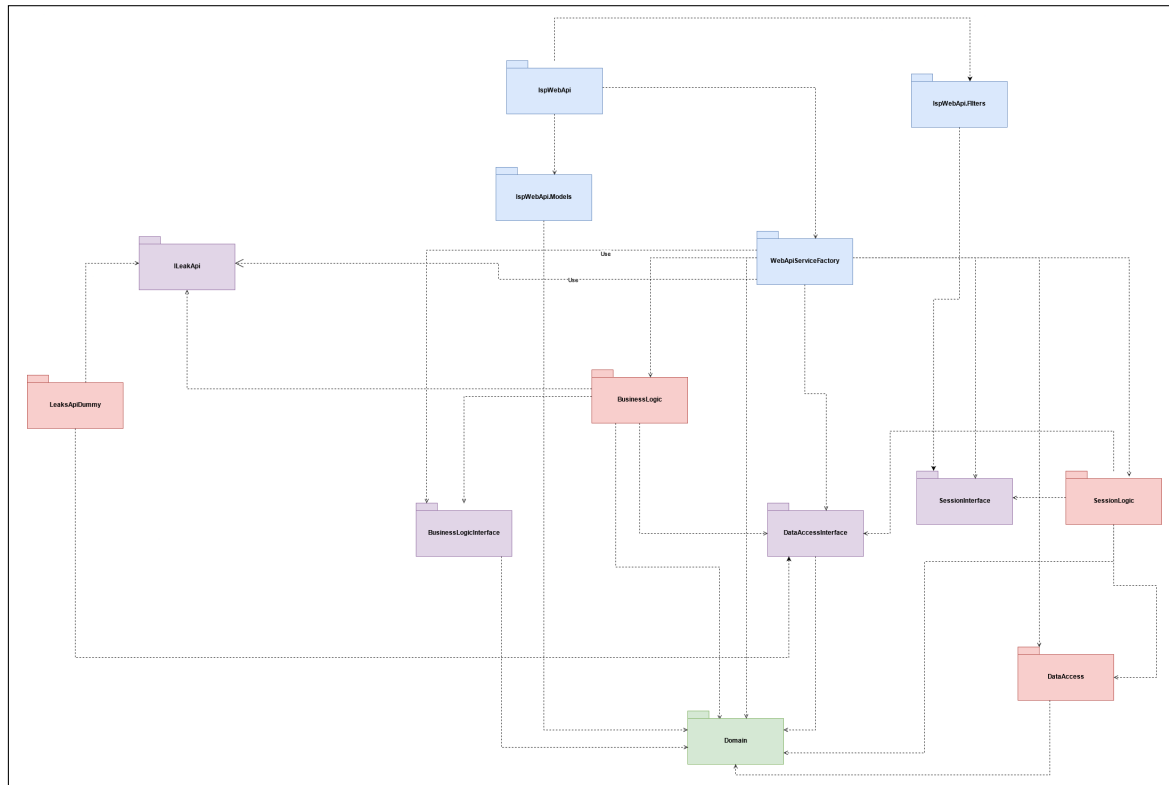


Figura 3.2: Diagrama UML de paquetes

3.1. Dominio

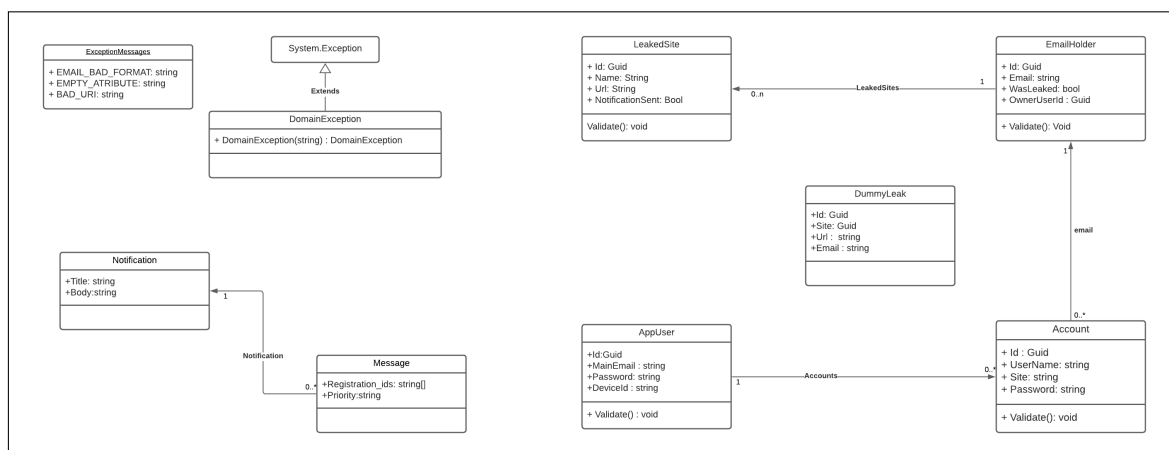


Figura 3.3: Diagrama de clases para el dominio

Comenzamos el análisis de la solución con uno de los paquetes más importantes para la misma, el domino. La solución se construyó siguiendo alguna de las ideas

planteadas por Martin Fowler sobre el dominio y su rol en la validación¹, algo así como un *DomainDrivenDevelopment*. Tras estudiar las ideas planteadas decidimos adoptar esa filosofía, manteniendo todas las validaciones que puedan ser realizadas por una entidad dentro de si misma, al final del día ella es su propia *information expert*.

Las validaciones que optamos usar caen en **excepciones propias** para el dominio, estas han de ser capturadas por la capa quien las solicite, por ejemplo la lógica, y manejadas acorde. Una excepción de dominio previo a un registro en la base de datos trae consigo la información de porque no es valido el modelo enviado. Esto reduce enormemente la carga de la lógica de negocios y el impacto de un cambio por parte del dominio.

Descripción de clases

- **AppUser:** se trata de los usuarios de la aplicación, poseen su colección de cuentas así como información que permite a los usuarios autenticarse en el sitio.
- **Account:** cada cuenta representa la información para autenticarse en un sitio particular.
- **EmailHolder:** a modo de facilitar la búsqueda y análisis de los correos para las filtraciones se abstraen a un contenedor por separado de su cuenta o cuentas. Cada emailholder posee un atributo que indica si fue filtrado previamente y una lista de los sitios donde se filtró.
- **LeakedSite y DummyLeak:** en la siguiente sección se entrará en detalle sobre el funcionamiento de los dummy leaks pero se diferencian de los leaked sites ya que estos últimos se planean conservar en futuras iteraciones del producto. Los dummy leaks existen como una forma de simular el registro de un leaked site real que luego se eliminaran de la base de datos.
- **DomainException y ExceptionMessages:** como mencionado previamente el dominio posee sus propias excepciones. Las mismas están empaquetadas junto al mismo y poseen una clase estática donde se almacenan los mensajes apropiados, a modo de facilitar un posible cambio en estos.
- **Message y Notification:** son utilizados como elementos de envío de notificaciones push.

¹<https://martinfowler.com/eaCatalog/domainModel.html>

3.2. ILeaksApi & LeaksApiDummy

A modo de desacoplar la mencionada API para filtraciones conocidas se implementa una interfaz capaz de agregar filtraciones y verificar que estas fueron filtradas para una lista de cuentas. La idea es que al momento de implementar una conexión a una API externa capaz de manejar estos sistemas de una forma real (como lo hace *HaveIBeenPwned*), el impacto en el proyecto sea mínimo, tan solo habría que realizar la incorporación a la aplicación de una clase que herede de ILeaksApi.

3.3. SessionLogic

Se coloco fuera de la lógica de negocios ya que se trata de una implementación sumamente genérica; quizás el día de mañana se nos requiera hacer otra api con un sistema de login y esta implementación nos pueda venir muy bien. Los cambios no son difíciles, y la dependencia al acceso a datos es rápidamente modificable.

En este paquete se encuentra el sistema que permite el login para usuarios registrados. Se mantiene un diccionario en memoria donde se almacenan tokens únicos por cada usuario que hace un login exitoso, además de un mecanismo de obtención de un usuario dado un token particular.

3.4. Business Logic

El paquete de lógica de negocios es el encargado de soportar todas aquellas operaciones requeridas por la especificación de requerimientos (las user stories), brindando las herramientas para operar y realizar las funcionalidades especificadas en el documento. Todas utilizan inyección de dependencias para poder soportar un cambio de acceso a datos. Todas heredan de su propia interfaz contenida en el paquete BusinessLogicInterface.

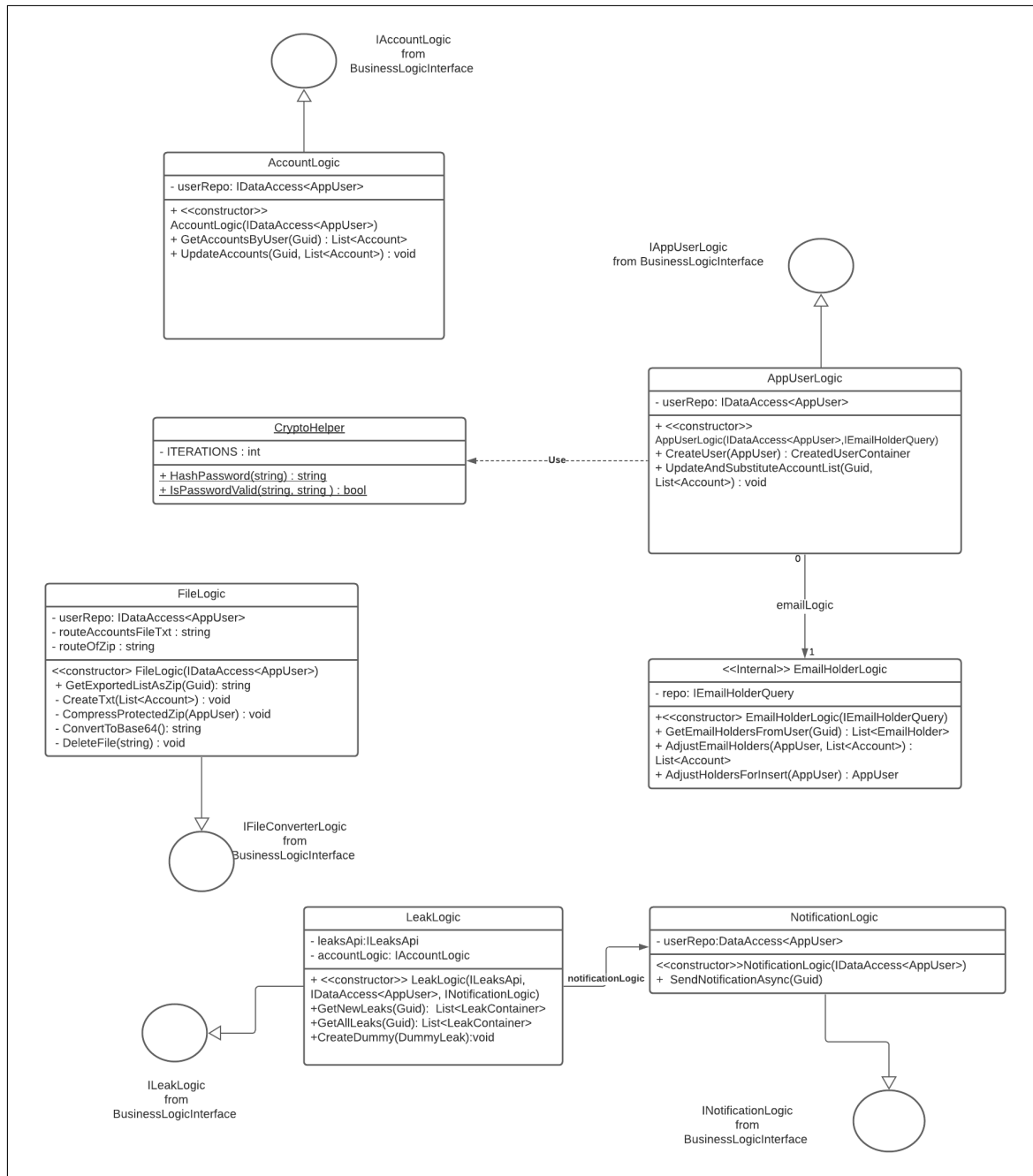


Figura 3.4: Diagrama de clases para la lógica de negocios

Al ampliar el paquete dentro del explorador verán que fue ordenado por carpetas, estas carpetas no representan una distinción de paquetes internos pero si una distinción lógica. Dentro de la carpeta de **logic** encontramos las principales clases de logicas que serán utilizadas posteriormente por la WebApi como una forma de acceder a las funcionalidades y al acceso a datos.

- **AppUserLogic:** esta clase es la responsable de administrar las modificaciones relacionadas con los usuarios. Permite el alta de un usuario sin repetir su correo

y también la actualización de las listas de usuarios. Esta clase hace uso de la utilidad de encriptado sobre la que se hablará mas adelante.

- **AccountLogic:** se trata de una clase especializada en las cuentas de cada usuario. Permite la modificación puntual para alguna de ellas o la obtención de las listas de un usuario.
- **EmailHolderLogic:** esta clase interna exclusiva al paquete de BusinessLogic se encarga de realizar el ajuste de los contenedores de los correos del usuario. Por ejemplo para el momento de insertar una cuenta si ambas comparten correo ambas deben referenciar al mismo contenedor. Esto permite el filtrado rápido de cuentas expuestas a futuro.
- **FileLogic:** es la encargada de la conversión de las cuentas de un usuario a un archivo en formato zip. Esta clase hace uso de la **dll** colocada en el mismo proyecto de .NET para realizar la conversión. El archivo es encodeado en base64 y encriptado tomando como contraseña el correo del usuario.
- **LeakLogic:** esta logica es la encargada de analizar los contenedores de correos de los usuarios y notificarles cuales fueron expuestos, donde y darles la posibilidad de elegir al solicitante de si desea obtener solamente las filtraciones no notificadas previamente o si desea el historico de las mismas.
- **NotificationLogic:** clase encargada del envio de notificaciones al correo del usuario.

Sobre CryptoHelper: Para el guardado de las contraseñas se hace ayuda de una funcion de hash apoyada con salting propio de cada cuenta². La clase cryptohelper proporciona un mecanismo de verificación de si la contraseña es correcta o no.

A destacar es que esta función no es aplicada por sobre las contraseñas de cada cuenta como debería ser aplicada en la versión final. El porque de esto es debido a que las cuentas del usuario han de ser exportadas directamente desde la webapi por lo acordado con el cliente (*el profesor*) y no podría ser retornada a su forma no-encriptada dada la naturaleza de las funciones hash. Para solucionar esto teníamos dos alternativas:

1. Utilizar AES o similar para tener un algoritmo que brinde la posibilidad de desencriptado utilizando información conocida (dígase la id del usuario). Esto requeriría que las cuentas se continúen enviando en texto plano.
2. No encriptar las contraseñas ya que el lugar correcto de hacer esta exportación es directamente sobre el celular.

²recomendamos el siguiente vídeo: How NOT to Store Passwords! - Computerphile

Optamos tomar la segunda ruta y documentar aquí la fuerte sugerencia de que no es seguro enviar las cuentas en texto plano para luego ser encriptadas en la WebApi. Con la existencia de los sniffers de trafico que podrían comprometer a cualquier usuario de nuestra app es mucho mejor realizar la encriptación del lado movil y dejar la WebApi como un futuro backup de información encriptada con algún mecanismo como *pbkdf2* basado en la contraseña y el correo del usuario.

3.5. BusinessLogicInterface

Se trata del paquete que contiene las interfaces para las clases mencionadas previamente. Además en ella se contienen los contenedores de formato de retorno de usuario creado y de filtraciones. Las excepciones de la lógica de negocio están ubicadas en este paquete. A modo de evitar una dependencia de un paquete de excepciones innecesario, las excepciones se contienen junto a las interfaces de quienes las requieran.

3.6. DataAccess

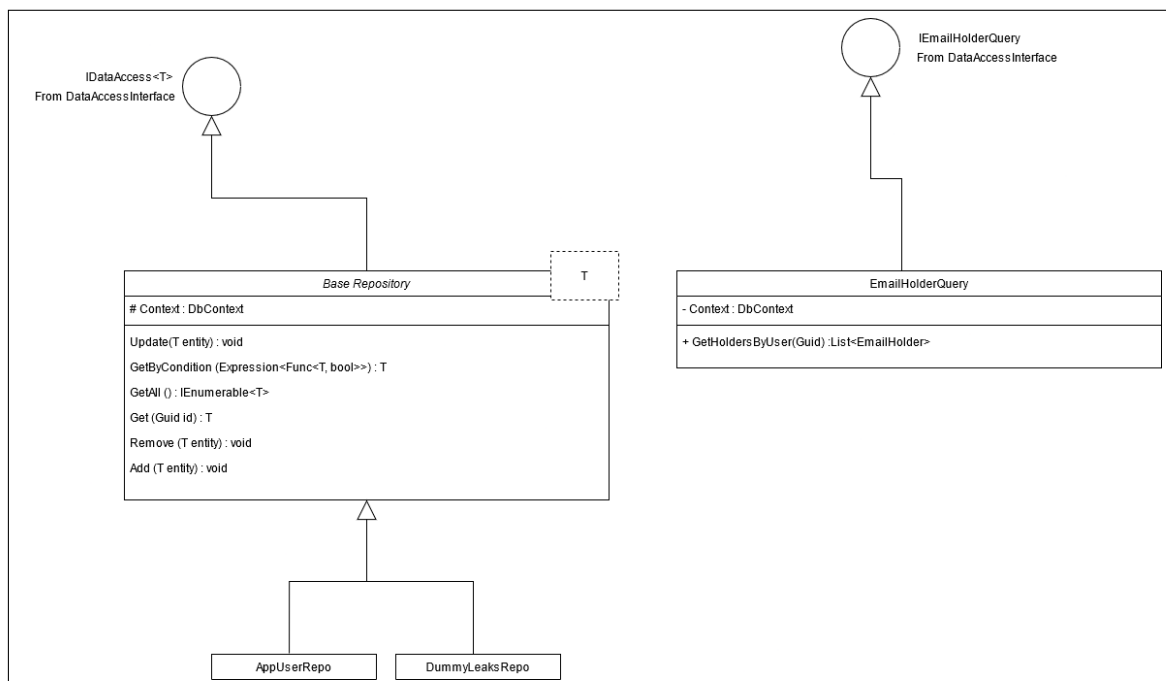


Figura 3.5: Diagrama de clases para el acceso a datos

Vemos aquí como se implemento un patrón repositorio abstracto para facilitar el uso de los mismos, junto a esto se logro ahorrar una alta cantidad de código repetido. De aparecer el día de mañana una nueva entidad implementar un repositorio nuevo sería sumamente sencillo, bastaría con hacerlo heredar de nuestro repositorio

base y sobrescribir aquellas operaciones que necesitemos. Referente a la sobre escritura de ciertos métodos los mas comunes a ser sobrescritos son aquellos referentes a la obtención de entidades; esto se debe a que el contexto utilizado responde a la metodología de trabajo *explicit loading*, donde le hemos de indicar que entidades traer de la base de datos al solicitar una entidad. En tanto al motor de base de datos usado y su uso tenemos la versión de sql de microsoft, SQL Server Express, a la cual accedemos mediante el ORM Entity Framework Core. Dentro de la carpeta Migrations en data access pueden encontrarse los distintos cambios que fue viviendo nuestra base de datos a lo largo del proyecto.

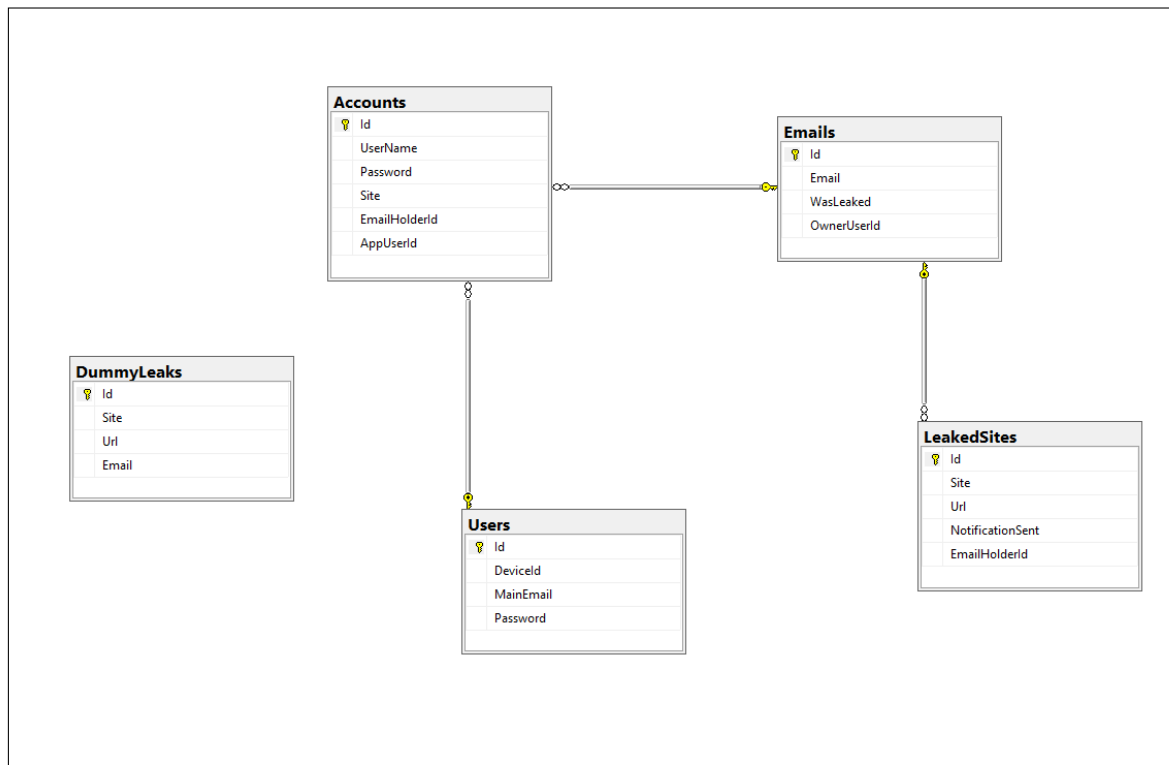


Figura 3.6: MER de la base de datos

El string de conexión a nuestra base de datos es rápidamente modificable desde la misma webapi como sera documentado mas adelante, pero se destaca que basta con cambiar la siguiente linea en *appsettings.json* por el nuevo string de conexión deseado.

```

"ConnectionStrings": {
  "Db": "Server=.\SQLEXPRESS;Database=Db;
  Trusted_Connection=True;MultipleActiveResultSets=True"
}
  
```

3.7. IspWebApi

Finalmente llegamos a la capa web, esta es la que expone nuestros recursos al mundo, al ser una api representa el patrón fachada para nuestra aplicación. Busca cuidar el uso de mensajes de error apropiado y maneja las excepciones lanzadas por capas inferiores acorde a ellas. Nuestro paquete WebApi esta compuesto por Models, Filters y Controllers. Los modelos conocen la implementación del domino y estos pueden convertirse en las entidades apropiadas, ya sea del dominio o datos de interfaz. Una petición llega a la webapi y esta viaja desde la misma a la lógica, de aquí al acceso a datos y finalmente llegaran al servidor SQL.

3.7.1. WebApiServiceFactory

Este es un paquete externo pero este es el mejor lugar para hablar del mismo. Para evitar que la webapi dependa virtualmente de todos los paquetes del sistema para configurar su startup se hace este paquete para manejar de una forma más elegante todo lo relacionado al scope de los servicios para la inyección de dependencia. Es una capa de abstracción entre la webapi y nuestra lógica y módulos inferiores.

Su única clase es llamada ScopedFactory y brinda todo lo necesario para inyectar dependencias entre los servicios y sus interfaces. Luego en IspWebApi se hace uso de esta para facilitar la inyección cumpliendo con lo dicho arriba.

3.7.2. IspWebApi.Models

Aquí radican los modelos que se manejan en la webapi. Se encuentran organizados por carpetas dependiendo de si es un modelo entrante o saliente. La existencia de modelos facilita enormemente la omisión de parámetros enormes y es este innerpackage que conoce al dominio, no la web api en si, eliminando una dependencia volátil innecesaria.

3.7.3. IspWebApi.Filters

Presentamos dos filtros en este paquete: el de sesión y el de excepciones. Por falta de tiempo no se pudo mejorar este ultimo para que sea el encargado de atajar todos los errores y construir una respuesta http basado en algún código presente en la excepción. Esa mejora quedara para otra implementación de una api pero hubiese reducido enormemente el tamaño de los controladores que ya no deberían utilizar try catch y la prolijidad de la misma. Pero de momento este innerpackage contiene los filtros mencionados. Para no confundir se destaca que el filtro de excepciones

solamente ataja aquellas **no capturadas** por la webapi, retornando un mensaje de error que informa al usuario del problema.

3.7.4. IspWebApi.Controllers

Buscando tener la menor lógica posible y haciendo uso del llamado a una o varias interfaces de lógica para trabajar, los controladores son quienes administran nuestros endpoints que son puntos de acceso al sistema que manejan los criterios REST.

Construimos nuestra API tratando de seguir el estilo arquitectónico REST. Para el manejo del obligatorio planteamos los siguientes *endpoints*, todos bajo una única dirección de un */api/recurso*. Los recursos con su *endpoint* se mostrarán a continuación:

- *api/exportedLists*
- *api/leaks/new*
- *api/leaks/all*
- *api/leaks/dummy*
- *api/sessions*
- *api/users*
- *api/users/accounts*

Previo a mostrar como podremos interactuar con nuestros recursos comenzaremos por analizar a los mismos bajo los ojos de REST. Un **recurso** es un elemento que exponemos en nuestro sistema, la manipulación que permitamos al usuario realizar con estos mismos determinara la usabilidad de nuestra API.

Como criterios adicionales para el estilo arquitectónico respetamos algunas restricciones:

1. **Prurales sobre singulares:** como se aprecia todos los recursos se exponen en plural. Es discutido en REST cuál es la mejor lectura de un recurso, si singular o plural pero como la principal aplicación de algunos *endpoints* es el de obtener todos los recursos optamos por el uso de plurales, y al optar por plural para un recurso es importante hacerlo en todos a modo de hacer nuestra api lo más intuitiva posible.
2. **Tamaño de la uri:** luego quedara evidenciado junto al *swagger* pero nuestra api es diseñada de forma tal que no nos pasemos de tres niveles. Siguiendo esta restricción evitamos la complejidad innecesaria para nuestra api.

3. **Nouns are good; verbs are bad:** esa es la frase que hemos de recordar al diseñar nuestros *endpoints* para recursos. No existe ningún endpoint donde se utilice un verbo en nuestra api dejando entonces la manipulación de los recursos solo mediante los verbos http que aceptemos recibir para un determinado recurso.
4. **Nombres concretos antes que abstracciones:** esto es un criterio que nos surgió de forma natural, al no existir herencias marcadas no hay ningún recurso que se solape con otro, pero, es importante tener en cuenta esto para el caso que se agregue algún recurso que pueda solaparse, por ejemplo algún tipo de rol similar a un usuario solo que sea administrador.
5. **Uri específica a la api:** para que un desarrollador no se confunda nuestra api con una url web de igual nombre se agrega la palabra api/ previo a cada recurso.

Para la generación automática de documentación utilizamos la herramienta **Swagger**. La herramienta permite visualizar que verbos http acepta cada recurso expuesto en nuestra api y además brinda la posibilidad de interactuar con la api misma. Para acceder a la misma basta con correr la api y entrar a (dirección donde se levante la api)/swagger

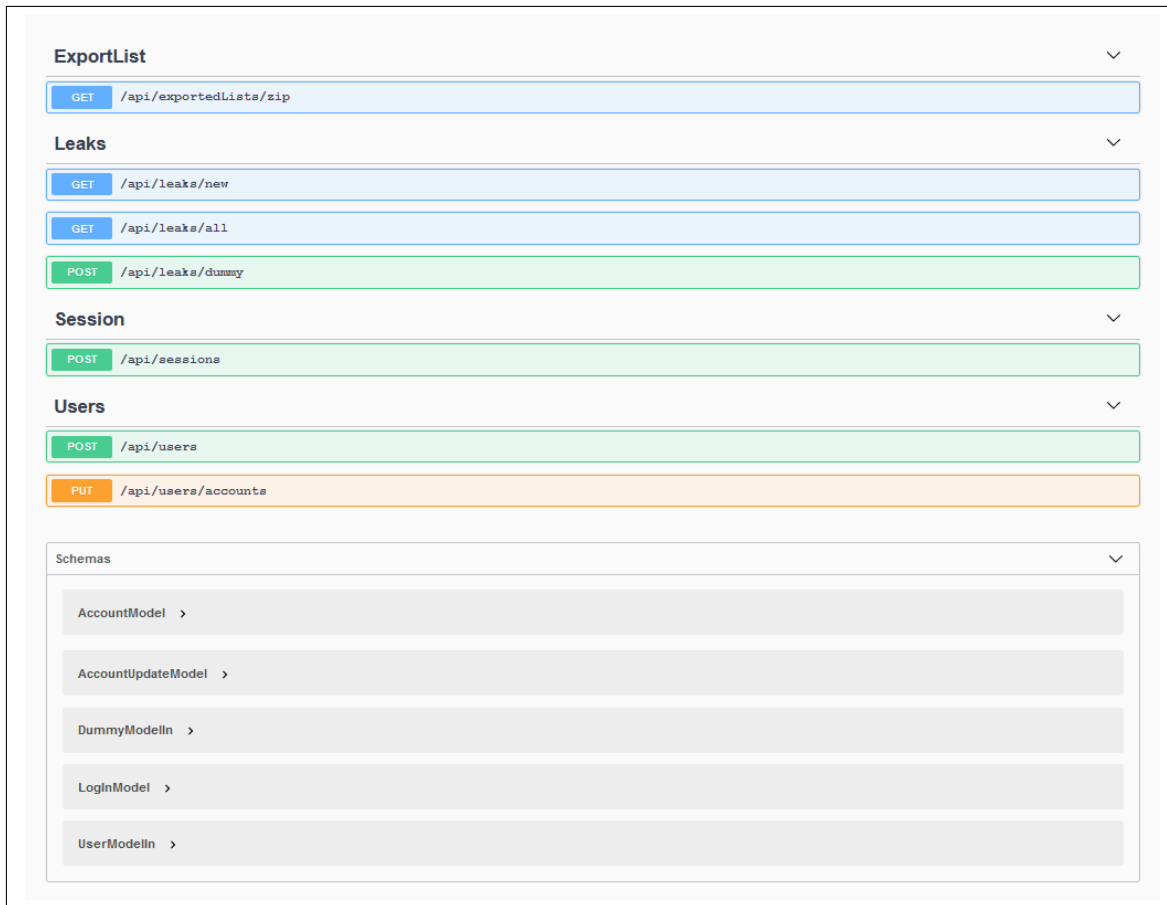


Figura 3.7: Vista en swagger de todos los endpoints

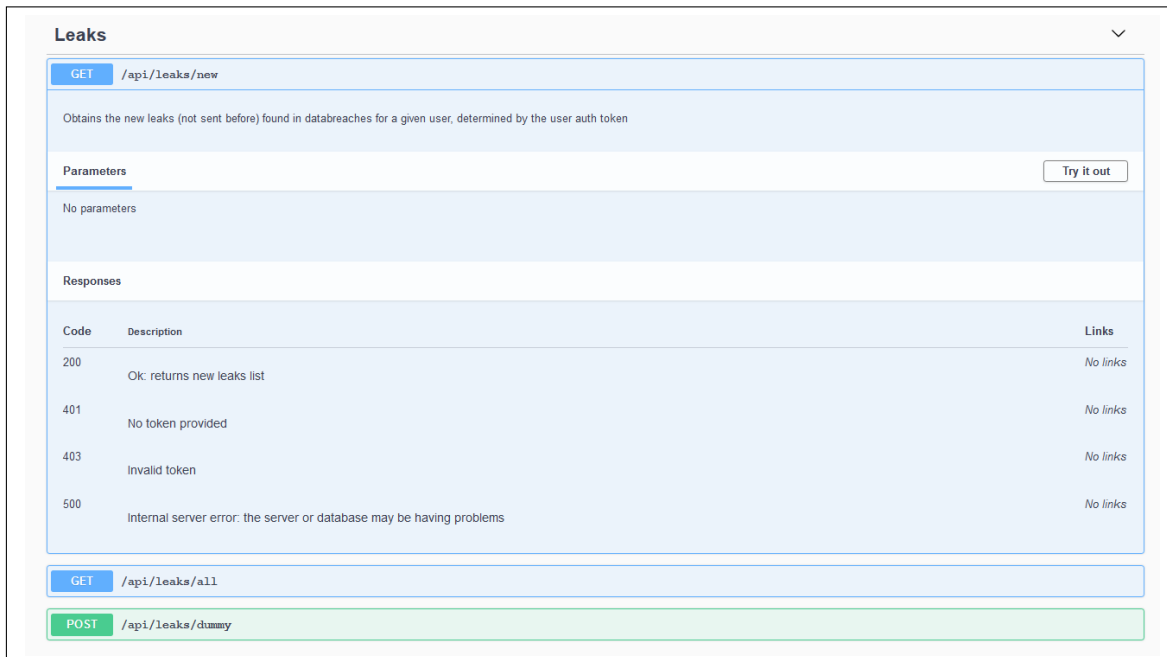


Figura 3.8: Vista de un recurso particular en swagger

Para no saturar al lector con todos los endpoints se adjuntan los mismos al anexo.

3.8. Errores conocidos

- Puede ocurrir que tras actualizar las cuentas del usuario no se ajuste bien el primer *new leak*, es decir el mismo aparece solo al solicitar todas las cuentas.
- La clase EmailManager puede recibir un refactor para mejorar su optimización o incluso mejor mover algunas operaciones a la capa de acceso a datos para eliminar la carga en memoria.
- Al actualizar las cuentas del usuario no se eliminan sus cuentas anteriores de la base de datos, por mas que a efectos prácticos para el usuario este todo correcto la base de datos se comienza a llenar de información innecesaria. Se planea eliminar primero antes de la actualizacion para la siguiente iteración.
- Debería extraerse el código para obtener un usuario basado en su Id para el session manager y retirar esa lógica inapropiada de los controller.
- No es necesariamente un error pero como los proyectos de la aplicación fueron creados en la versión 3.1 se tuvieron problemas para separar las uri de la creación de los archivos a un App.config, en versiones futuras se ha de rearmar el proyecto en versiones actuales y separar las direcciones harcodeadas a un archivo externo.

4. Implementación de la Solución: Frontend

A continuación se dará lugar a la explicación de la implementación del frontend realizado en android studio. Se mencionaran los paquetes utilizados, la arquitectura general, la arquitectura UI y el diseño de vistas.

4.1. Paquetes de solución

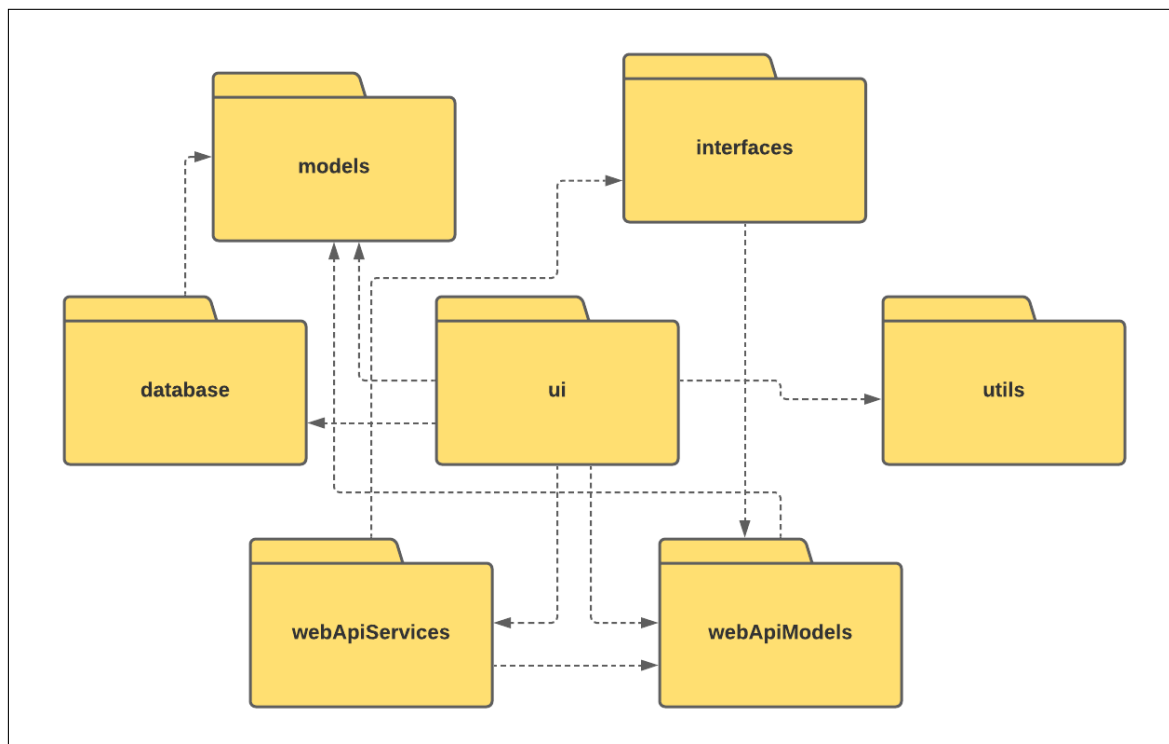


Figura 4.1: Diagrama de paquetes user interface

4.1.1. Organización de paquetes

Breve explicación de la organización de cada paquete, mas adelante se explicara en detalle el funcionamiento de cada paquete.

- **Paquete utils:** El paquete utils fue utilizado para definir las utilidades comunes dentro del sistema. Su objetivo principal fue el de reutilizar código, dentro de ella se encuentra por ejemplo el encriptado y desencriptado de texto el cual es usado recurrentemente dentro de la aplicación.
- **Paquete models:** Dentro de este paquete se encuentran los modelos de las entidades de ROOM. Como se puede observar en el diagrama, es un paquete muy utilizado por los paquetes ui, database y webApiModels. Todos los paquetes del frontend dependen de cierto tipo de modelos.
- **Paquete UI:** Este paquete contiene todo lo relacionado a las vistas y su cargado, con esto nos referimos a las actividades, los adaptadores para las recycler view y por ultimo los diálogos personalizados que se construyeron.
- **Paquete database:** Contiene todas las clases relacionadas con la persistencia de datos del frontend, las clases de repositorios, los data access objects (DAOs), y la clase de generación de la base de datos ROOM. Todas las clases dependen de las entidades ROOM (dentro del paquete models).
- **Paquete webApiModels:** Contiene los modelos utilizados en el backend, dentro de este paquete se encuentran otros dos paquetes mas que son in y out los cuales refieren a los los modelos in y out de la webapi del backend. Como el paquete models, este es requerido por varios paquetes.
- **Paquete webApiServices:** Este paquete es el encargado de realizar las conexiones con la api en backend, hace uso de webapiModels y models.

4.2. Arquitectura general

A continuación definiremos la arquitectura general utilizada dentro del frontend. Como en el backend se utilizaron patrones para diseñar las distintas clases y vistas.

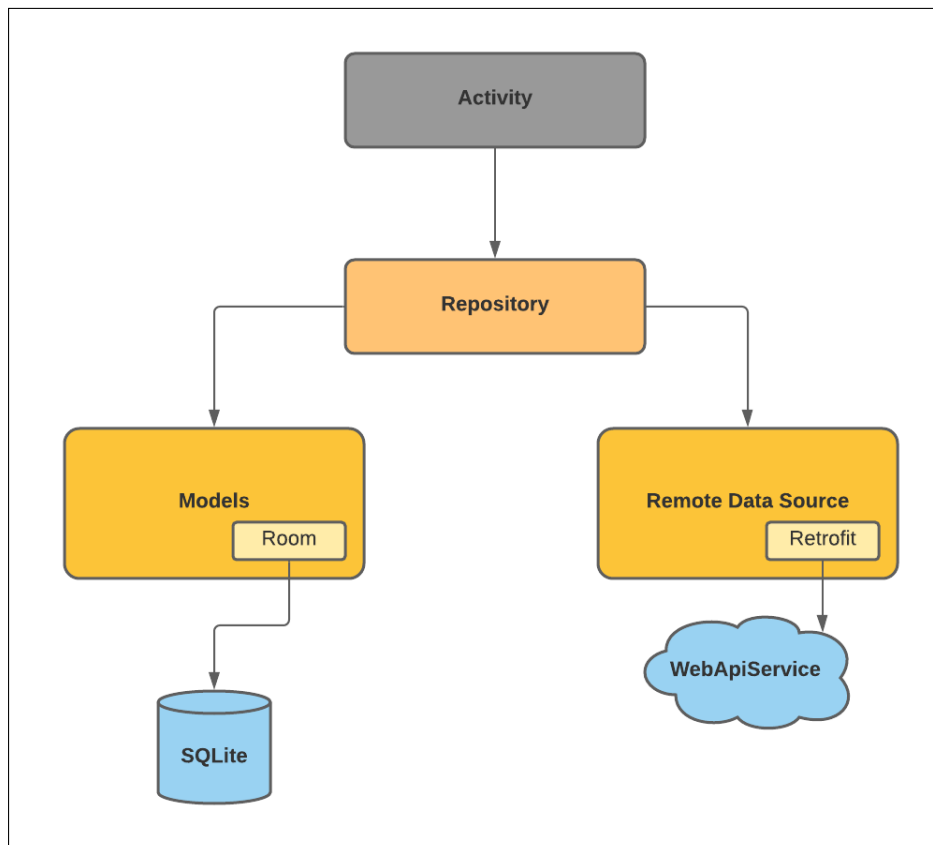


Figura 4.2: Diagrama de arquitectura general

Para realizar y diseñar la aplicación frontend decidimos seguir consejos de aplicaciones ya creadas aconsejadas por las mismas guías de android studio y las otorgadas por el curso de Ingeniería de software en la práctica.

Comenzando desde arriba, se encuentran las actividades, estas son las clases encargadas de llevar la lógica realizada en los paquetes a las pantallas del dispositivo. Las mismas se encargan de reorganizar la información y permitir que cada pantalla se muestre de manera correcta (con esto nos referimos al correcto funcionamiento de botones, inserción, eliminación, encriptación, etc), Consideramos que estas actividades son centrales para la aplicación dado que permiten el correcto funcionamiento de la app.

Estas actividades se encuentran conectadas a los repositorios los cuales son el equivalente a la lógica de las clases del dominio del backend. Dentro de estas clases se manejan los datos de la base de datos del frontend (el modelo), y se puede efectuar la comunicación con otros servicios web, como la api expuesta por el backend. Para esto último es necesario utilizar Retrofit, el cual es un cliente para apis REST de java y android.

Finalmente se tiene la base de datos en el frontend, el modelo. Para el modelo se decidió implementar ROOM, la cual es una librería de persistencia que utiliza SQLiteDatabase (base de datos en android), pero que facilita su uso e implementación.

ROOM funciona de la siguiente manera:

ROOM es una gran herramienta, facilita y simplifica mucho el uso de una base de datos. En principio se tienen las entidades. Estas se utilizan para construir una tabla para el objeto dentro de la base de datos. Estas entidades definen las variables de la clase y las propiedades de las mismas para la base de datos, como por ejemplo, ser la Primary Key del objeto (atributos por los que se pueden reconocer un objeto dentro de la tabla) y no aceptar campos en nulo entre otras tantas propiedades. Además, estas entidades funcionan como objetos, por lo que se le puede definir el comportamiento como constructores o métodos.

```
@Dao
public interface NoteDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    public void insert(Note note);

    @Update(onConflict = OnConflictStrategy.REPLACE)
    public void update(Note note);

    @Delete
    public void delete(Note note);

    @Query("SELECT * FROM notes_table WHERE user_identifier = :user_identifier")
    List<Note> getUserNotes(Integer user_identifier);

    @Query("SELECT * FROM notes_table WHERE user_identifier = :user_identifier and id = :id")
    Note getUserNoteById(Integer user_identifier, Integer id);

    @Query("SELECT * FROM notes_table ORDER BY id ASC")
    public List<Note> getAll();

}
```

Figura 4.3: Ejemplo NoteDao

Por otra parte, se hace uso del patrón Data Access Objects, por lo que se hace uso de estas clases (DAOs). Los DAOs son interfaces que exponen operaciones sobre la base de datos, pero sin exponer detalles sobre la misma. En el ejemplo se pueden ver operaciones básicas como insert, delete, update o queries sobre la tabla de productos de la base de datos. Estas interfaces son las instanciadas por los repositorios para poder efectuar las operaciones sobre la base de datos.

```

@Entity(tableName = "notes_table",
        indices = {@Index(value = "id", unique = true)})
public class Note implements Serializable {

    @PrimaryKey(autoGenerate = true)
    @NonNull
    public Integer id;

    @NonNull
    @ColumnInfo(name = "user_idenfifier")
    public Integer user_idenfifier;

    @NonNull
    @ColumnInfo(name = "title")
    public String title;

    @NonNull
    @ColumnInfo(name = "content")
    public String content;

    public Note(Note note) {
        this.id = note.id;
        this.user_idenfifier = note.user_idenfifier;
        this.title = note.title;
        this.content = note.content;
    }
}

```

Figura 4.4: Ejemplo tabla de notas

Por ultimo, se tiene la clase para la base de datos. Esta clase delimita las tablas que se van a persistir en la base de datos listando las entidades creadas anteriormente. Como se puede ver, la clase es abstracta y extiende de RoomDataBase, y posee como variables abstractas a las DAOs correspondientes de la entidades. Como esta clase es la encargada de mantener la base de datos, se utiliza el patrón Singleton para asegurar que la misma versión de la base de datos se utilice en todo el frontend.

```

import ...

@Database(entities = {User.class, Note.class, Password.class}, version = 1)
public abstract class IspDb extends RoomDatabase {

    public abstract UserDao userDao();
    public abstract NoteDao noteDao();
    public abstract PasswordDao passwordDao();

    private static volatile IspDb INSTANCE;
    private static final int NUMBER_OF_THREADS = 3;
    public static final ExecutorService databaseWriteExecutor = Executors.newFixedThreadPool(NUMBER_OF_THREADS);

    public static IspDb getDatabase(final Context context) {
        if (INSTANCE == null) {
            synchronized (IspDb.class) {
                if (INSTANCE == null) {
                    INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                        IspDb.class, name: "isp_database")
                        .allowMainThreadQueries()
                        .fallbackToDestructiveMigration()
                        .build();
                }
            }
        }
        return INSTANCE;
    }
}

```

Figura 4.5: Ejemplo base de datos IspDb

4.3. Arquitectura UI

La UI fue diseñada usando un arquitectura en donde cada funcionalidad del sistema contiene una actividad para uso personal. (Por ejemplo, Login, Notes y Passwords). Cada actividad se ve definida en el android manifest de forma ordenada donde se declara la actividad que se ejecuta en primer lugar. Nos hubiese gustado realizar una arquitectura basada en Single Activity Architecture pero por falta de tiempo y conocimiento fuimos incapaces de lograrlo. La misma trata sobre la utilización de una sola MainActivity de la cual se vayan cargando distintos fragmentos de la aplicación. Esta arquitectura hubiese favorecido la performance en el cambio de vista ya que los fragmentos son mucho mas livianos que una actividad. Sin dudas esta funcionalidad quedara anotada para próximas versiones de la aplicación.

No establecimos una clase navegación pero pese a esto proporcionamos un toolbar en cada pantalla el cual se encargara de orientar al usuario y el mismo fuera capaz de navegar entre las distintas pantalla sin problema, también se configuro el botón para atrás que ofrece android para que la navegación fuera fluida y sin problemas.

4.4. Diseño de vistas

Para el diseño de las vistas no fue utilizado un único formato, dependiendo de la cantidad de información que tuviera la pantalla, el layout podía ser Linear Layout,

Coordinator Layout o Relative Layout. El motivo de selección se baso en que la app pueda ser utilizada en la mayor cantidad de dispositivos posibles. A continuación la explicación de porque la selección de cada layout.

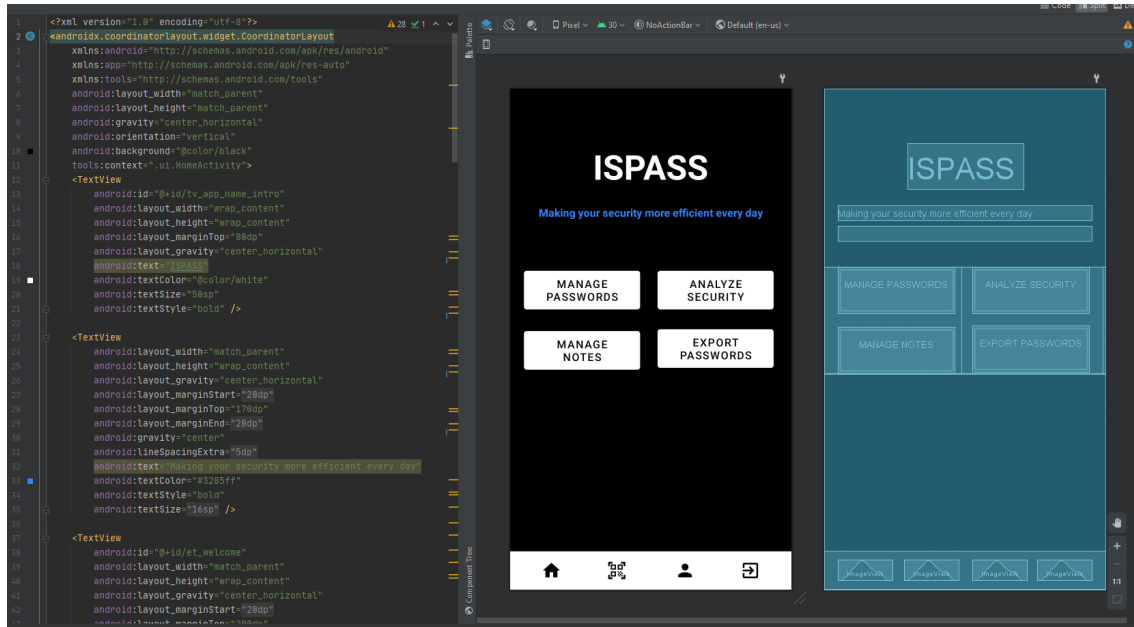


Figura 4.6: Ejemplo layout home

Linear Layout fue el menos utilizado dado que no es ajustable para todo tipo de pantallas, este layout fue utilizado en pantallas como login o user, donde la información a mostrar es poca y toda centrada. Sin importar las dimensiones de la pantalla que contenga el usuario, la información sera visible de forma ordenada y completa.

Coordinator Layout fue de los mas utilizados dado que permite una mejor organización de la pantalla y se adapta a cualquier tipo de dimensión. Este layout permite orientar objetos al principio o al final de pantallas sin importar sus dimensiones, también permite el uso de `scrollview` el cual en nuestro caso fue importante para por ejemplo poder mostrar las distintas contraseñas, nota y filtraciones.

Por ultimo se encuentra el relative layout el cual fue utilizado para el agregado y actualización de nota y contraseñas. Fue utilizado este tipo de layout dado que permitía una mejor organización de los datos y era útil como pantalla rápida.

La utilización de estos layout permitió crear una única vista para cada actividad del frontend, y que la vista se modifique de acuerdo a las medidas del dispositivo utilizado, manteniendo la proporción de la vista intacta. Como consecuencia, se elimino la necesidad de crear varias vistas para una misma actividad dependiendo del tamaño del dispositivo, ahorrando grandes cantidades de retrabajo, de esta forma se logro que la experiencia de usuario sea la misma sin importar el tipo de dispositivo celular que se utilice.

Otro detalle relevante es el diseño en torno a las cardViews, es un widget que nos ofrece android y que esta siendo bastante utilizado en la industria. Le da estilo y profundidad a las vistas, haciendo la experiencia de usuario mas agradable. Este mismo fue utilizado para el ingreso de datos del usuario. Otro widget que también se utilizo fue el de toolbar también otorgado por android, el mismo fue utilizado como barra y navegador de muchas pantallas brindándole al usuario información sobre en que pantalla se esta y la capacidad de navegar de forma prolija y fácil.

5. Visualización de la Solución

5.1. Primeros mockups

Con motivo de plasmar lo mencionado anteriormente a continuación se mostrará el despliegue de algunos mockups para mostrar un prototipo de lo que se esperaba obtener. Para los mockups se intentó representar las ventanas más importantes de la aplicación pero no esta de más decir que no todas las ventanas y funcionalidades se encuentran representadas en los mockups. Si bien estas imágenes dan una buena guía hacia la aplicación que queremos realizar, las mismas no muestra el resultado final de la solución sino el esqueleto de lo que se deseaba obtener como resultado.

Estos prototipos nos han servido como guía para la realización de la primera versión y sin duda serán nuestro objetivo para el producto final.

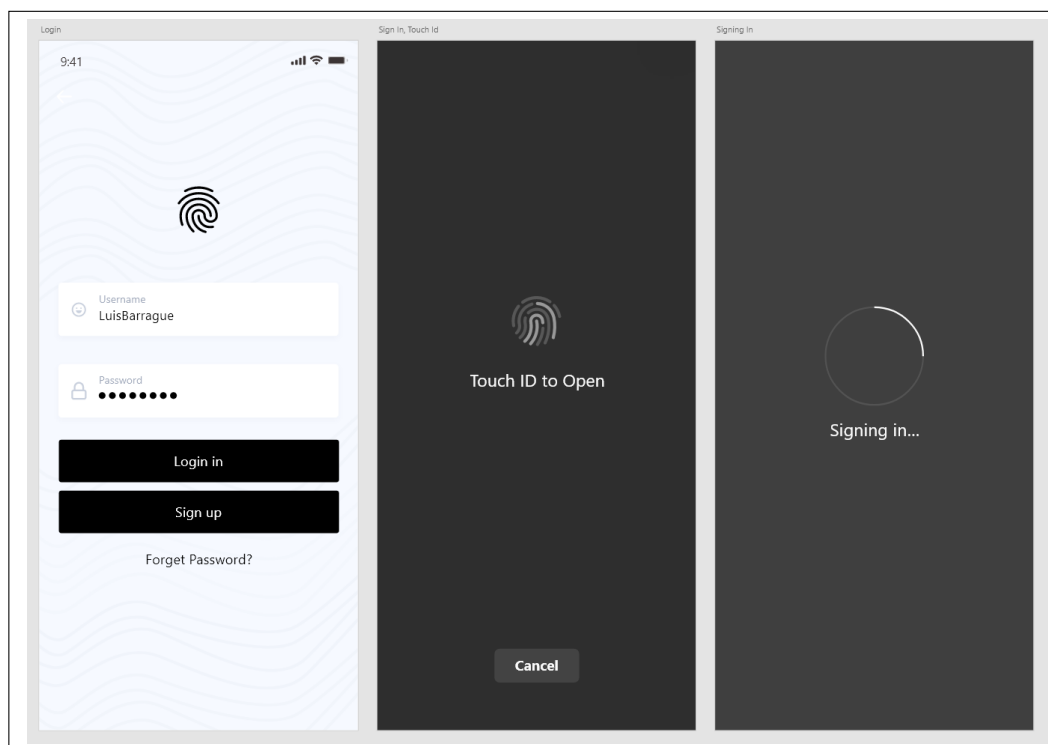


Figura 5.1: Ventanas de Login

Empezando por lo primero, las ventanas del Login serán las primeras que verá el usuario al momento de ingresar a la aplicación. En la figura 5.1 se puede observar el proceso de izquierda a derecha en la imagen.

Al ingresar el usuario observa la ventana login donde se le solicita un usuario y una contraseña (por motivos de visualización se *autocompletó* el usuario y la contraseña pero se espera que solo se sugiera el usuario y la contraseña deba ser escrita manualmente). Este usuario y contraseña son fundamentales debido a que son la llave maestra al resto de las funcionalidades de la aplicación. Una de las funcionalidades que se presentan en el Login es el uso de la huella digital para móviles que tenga disponible esta funcionalidad. Si el usuario tiene la opción de ingresar con su huella, al momento de ingresar a la aplicación, solo debe presionar su huella y así ingresar a la misma.

Para finalizar con estas ventanas, también se encuentran los botones: Log in, Sign up. Los cuales indican como dice las palabras, para ingresar a la aplicación, para registrarse (el cual abriría una ventana donde se le permitiría crear una cuenta).

En segundo lugar se tiene la ventana fundamental de la aplicación siendo esta la pantalla principal.

Como se observa en la figura 5.2, la pantalla principal cuenta con un mensaje de bienvenida y debajo del mismo 3 botones que muestran las funcionalidades claves de la aplicación, estos botones se encargan de dirigir a ventanas secundarias donde se interactúa con estas funcionalidades. A continuación veremos estas más detalladamente.

Debajo de estos botones, se encuentra una barra la cual estará presente en todas las ventanas de la aplicación. Esta misma cuenta con 3 funcionalidades las cuales son, observar contraseñas (más hacia las izquierda) cuyo objetivo es simplemente mostrar una ventana donde se encuentren las contraseñas del usuario, luego en el centro de la barra se encuentra la opción scan QR cuyo objetivo es escanear una contraseña desde otro dispositivo (funcionalidad la cual hablaremos más adelante) y por último y no menos importante, a la derecha la opción de configuración la cual dirige a la ventana configuración de la aplicación con la cual el usuario puede interactuar (información del usuario).

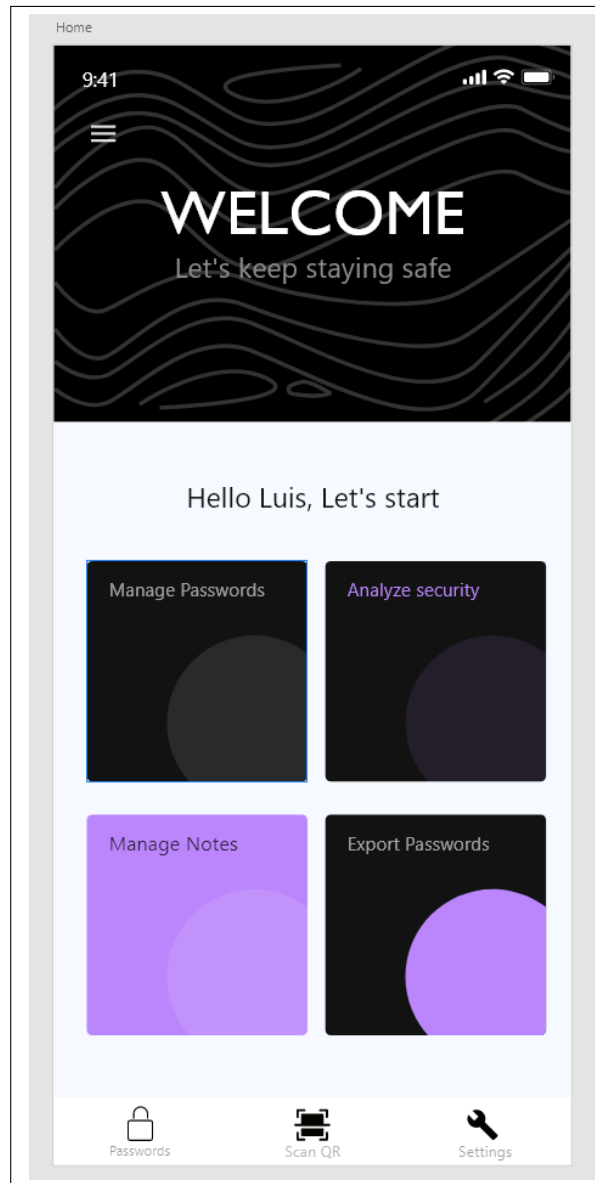


Figura 5.2: Ventana principal

En tercer lugar como se observa en la figura 5.3 se pueden observar las pantallas secundarias las cuales son las ventanas administrar contraseñas (izquierda), administrar notas (centro) y analizar correo electrónico (derecha) las cuales se observaron en la pantalla principal.

Comenzando por la ventana de administración de contraseñas se puede observar las destinas contraseñas (con un ícono y su nombre al costado indicando a que sitio corresponden) las cuales contienen posterior a su nombre, la opción de ver y editar la contraseña. Para poder hacer uso de cualquiera de estas 2 opciones, el usuario debe volver a ingresar su huella. Sumado a estas funcionalidades, se puede observar un botón llamado Add Password el cual ofrece la posibilidad de agregar una nueva contraseña a la base de datos.

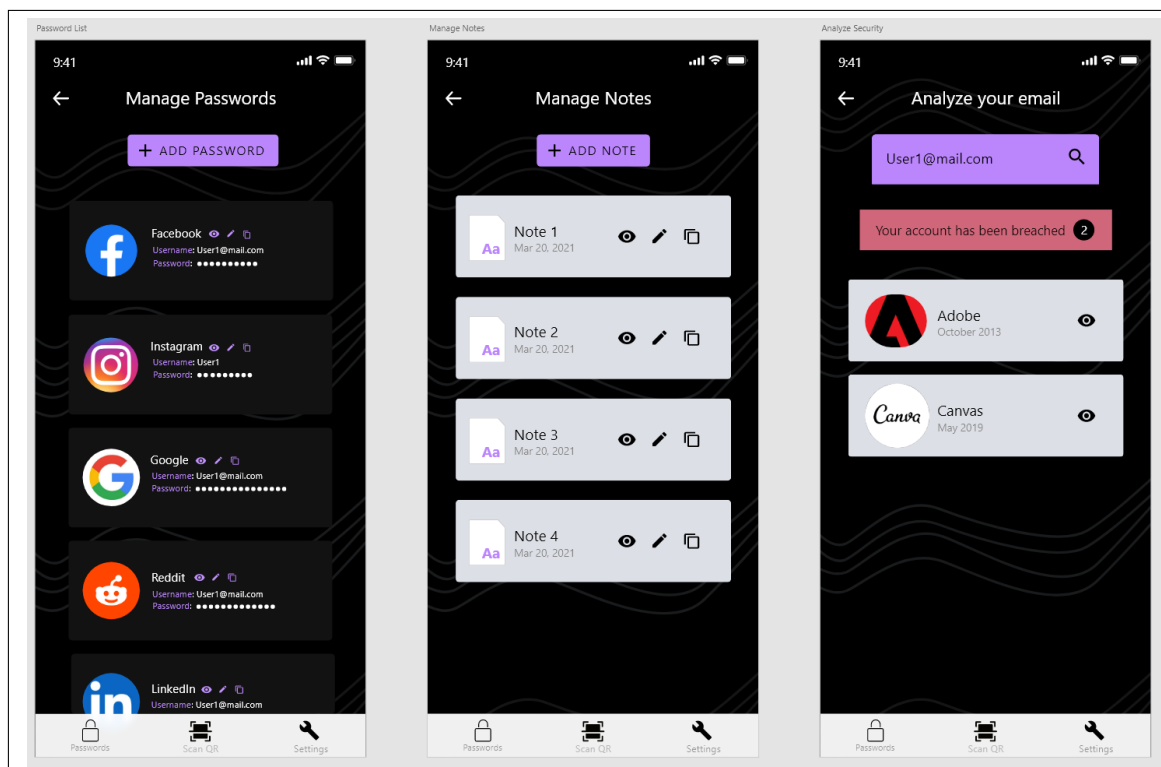


Figura 5.3: Ventanas de contraseñas, notas y análisis

Siguiendo con la ventana de administración de notas, como se vió en el product backlog, esta brinda la posibilidad de guardar notas de forma segura. Esta misma es parecida a la ventana administración de contraseñas en el sentido que tiene opciones de visualización y edición, como así también la posibilidad de agregar nuevos elementos. Como en el caso de administración de contraseñas, ninguna nota es observable hasta que el usuario brinde autenticación de su huella digital.

Por último se encuentra la ventana de análisis de correo electrónico la cual como fue mencionado anteriormente en este documento, brinda la posibilidad de al ingresar vuestro correo electrónico, observar las vulnerabilidades y ataques recibidos.

Cabe mencionar que si bien no fue creado un prototipo para la opción de exportar contraseñas del menú principal, la idea es que cuando el usuario presione la opción, se exporte de la aplicación un documento el cual contenga todas las contraseñas del usuario.

Pese a que ya fue mencionado, es importante aclarar que esto es simplemente un prototipo que debe tomarse como ejemplo y no como el resultado final del producto.

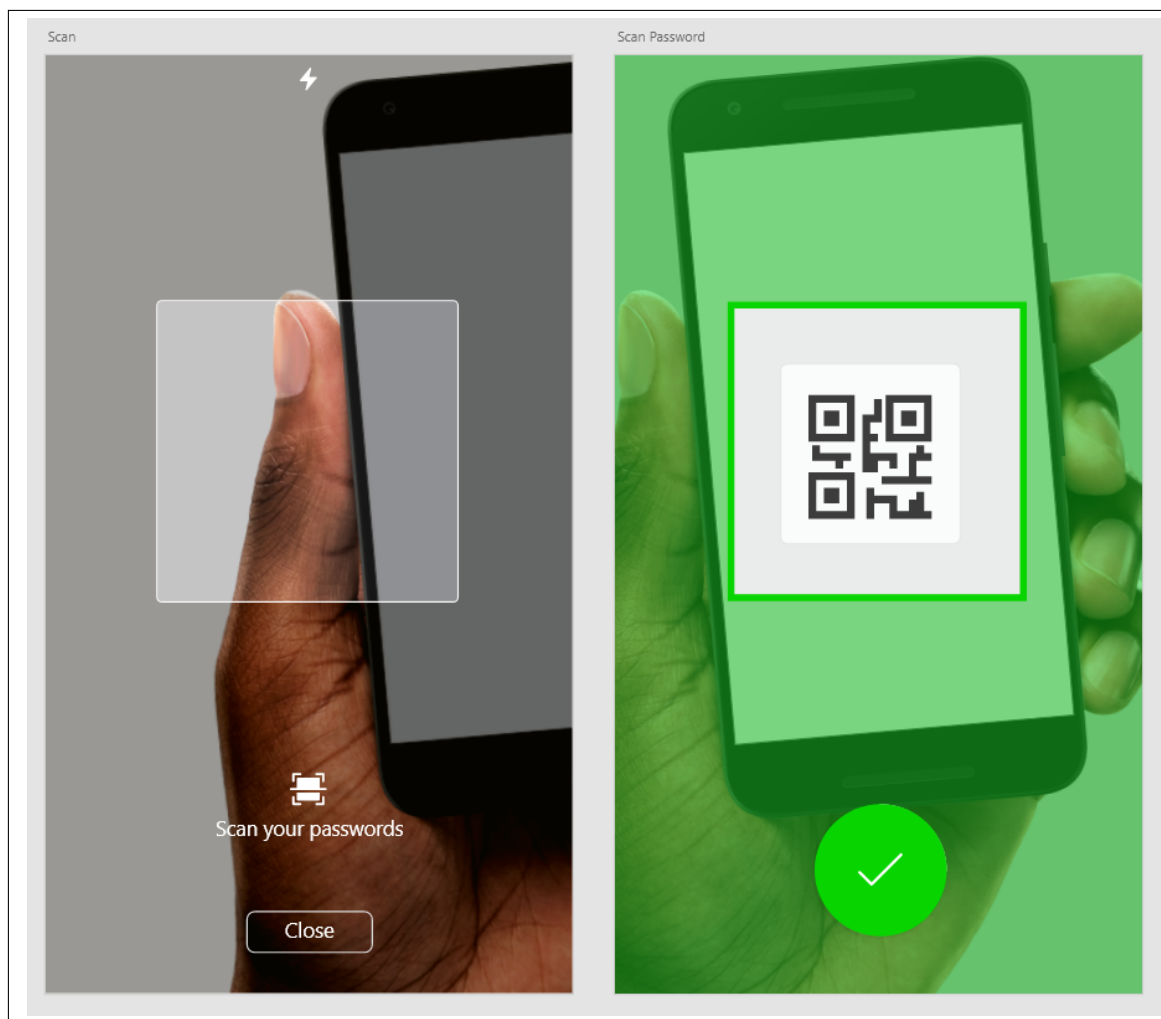


Figura 5.4: Ventanas de lectura de código QR

Por último y no menos importante se encuentra la opción de la barra baja ya mencionada sobre el lector de el código QR. Como fue mencionado anteriormente en el product backlog, esta opción brinda al usuario la posibilidad de escanear una contraseña y guardarla en la aplicación. Como se ve en la Figura 5.4 el usuario debe colocar el código QR dentro del cuadrado que se muestra.

5.2. Primera solución real

Como fue mencionado en el prototipo inicial, era muy difícil lograr que se cumpliera con una aplicación igual a la prototipada. Efectivamente eso ocurrió y no fuimos capaces de realizar una igual, igualmente se logro cumplir con la mayoría de las funcionalidades propuestas y si bien el diseño no es el mismo, esta es una primera versión de la aplicación la cual a futuro (bajo la ayuda de un diseñador gráfico), puede parecerse mas al prototipo diseñado.

A continuación se dará la explicación de la nueva distribución pantalla por pantalla y el funcionamiento de la aplicación.

5.2.1. Ingreso y registro

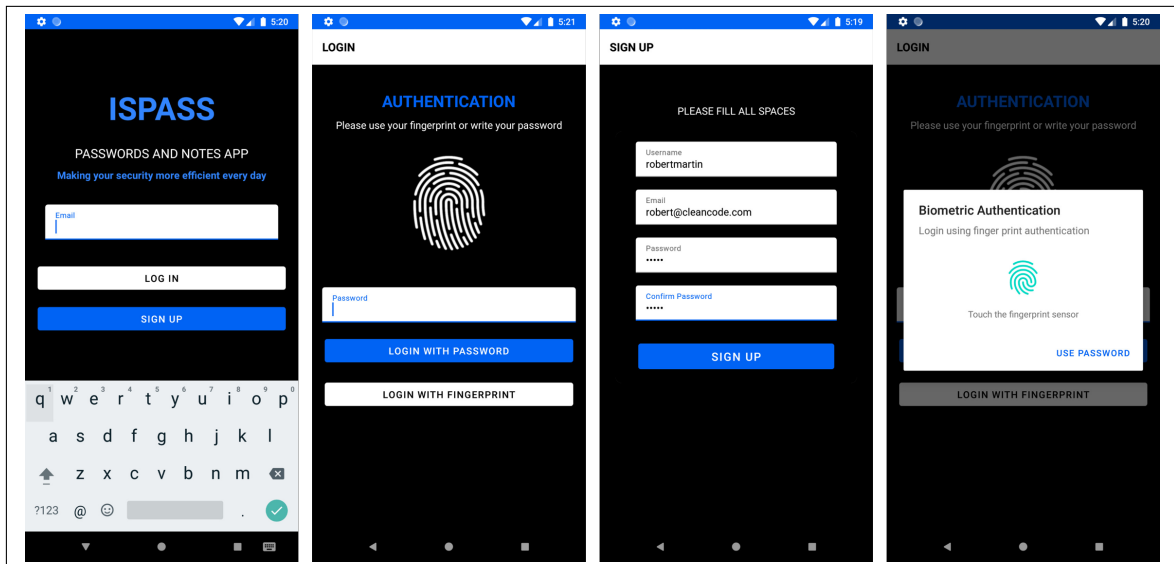


Figura 5.5: Ventanas de ingreso y registro

En cuanto al ingreso y registro a la aplicación, se realizaron cambios en cuanto al modelo del prototipo pero se mantuvo las funcionalidades planteadas. Al ingresar a la app, la primera pantalla que se ve es la de ingreso donde se permite ingresar un correo, si el correo existe en la base de datos, entonces se pasara a la pantalla de autenticación donde se permitirá ingresar mediante la contraseña guardada para ese usuario o mediante la huella digital (si es que el usuario contiene una huella registrada en su dispositivo).

En caso de que el usuario no se encuentre registrado, se indicara mediante un mensaje y se le permitirá presionando el botón de sign up, registrar un nuevo usuario como se ve en la tercer pantalla desde la izquierda. El sistema esta rodeado de mensajes de advertencias y error con tal de que el usuario pueda reconocer que es lo que esta sucediendo.

5.2.2. Pantalla principal

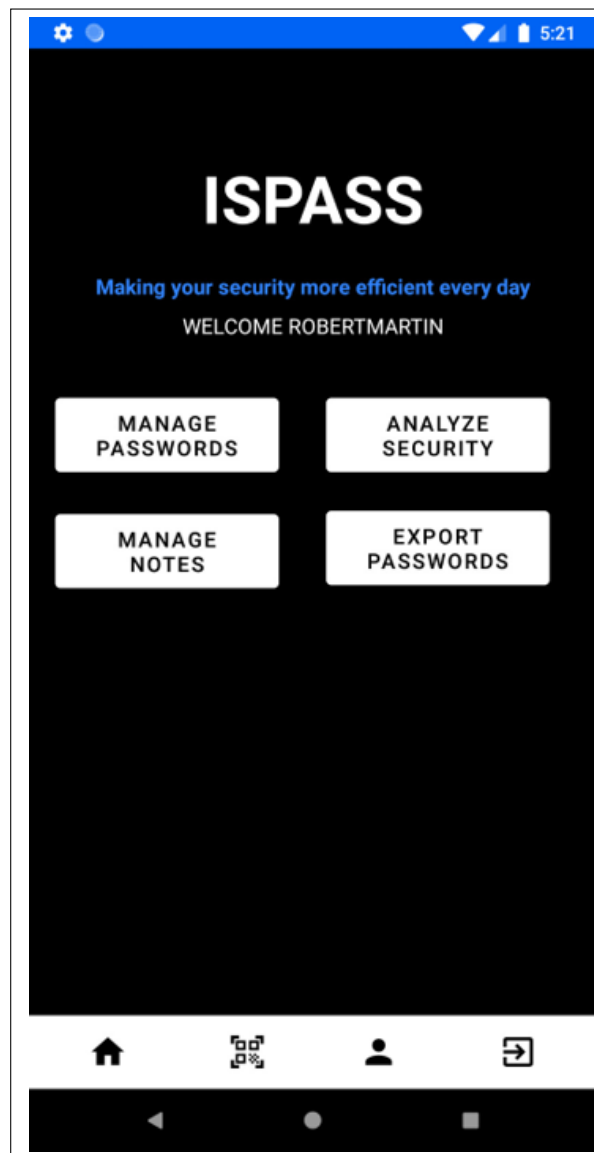


Figura 5.6: Ventana pantalla principal

Al igual que como en la pantalla login, se mantuvieron las funcionalidades planteadas pero no se logro repetir el diseño establecido. Como había sido planteado en el prototipo, esta pantalla sirve como raíz para el resto de la aplicación. A diferencia de el prototipo inicial se brindo una opción para *deslogearse* de la aplicación, como se ve en la barra inferior

5.2.3. Contraseñas

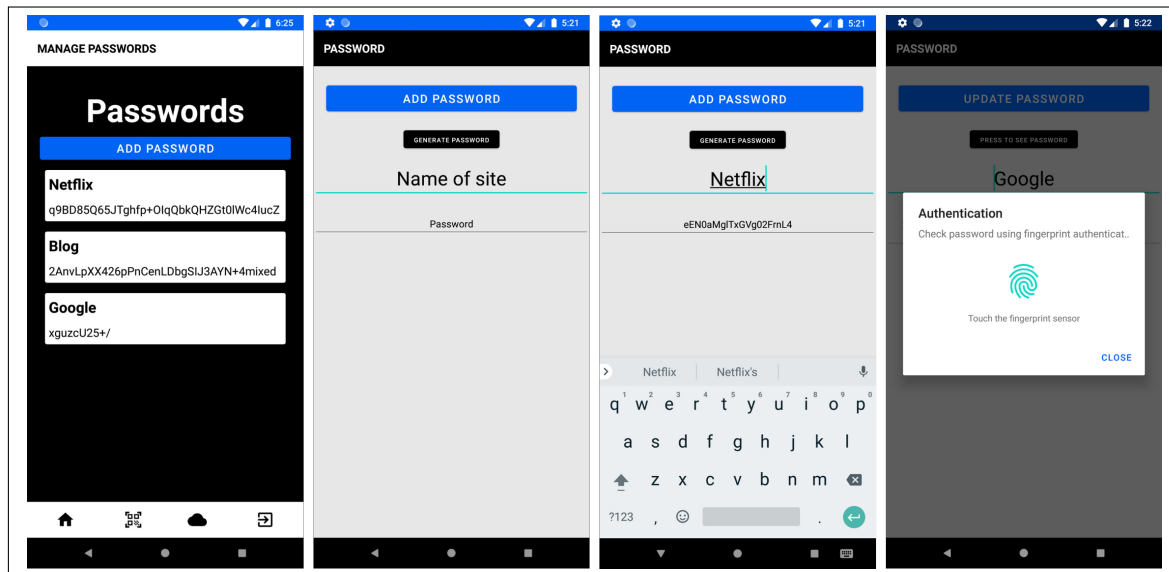


Figura 5.7: Ventanas de contraseñas

Para las pantallas de contraseñas se repite lo mismo sobre el mantenimiento de las funcionalidades pero no del diseño. En esta nueva versión no se puede ver la imagen del sitio de la contraseña, ni copiar una contraseña, pero si el nombre de la pagina y su contraseña encriptada debajo.

Al agregar una contraseña se abre una nueva pantalla la cual permite introducir el nombre del sitio y la contraseña del mismo. A su vez, si es que se desea generar una nueva contraseña, al presionar el botón de generar nueva contraseña, el sistema proporcionara una contraseña aleatoria alfanumérica para que el usuario pueda utilizar, al guardar, se encriptara la contraseña y se retornara a la pagina de contraseñas donde se podrá ver la nueva contraseña registrada.

Al presionar una contraseña, la misma se abrirá en una nueva pantalla donde se podrán ver los datos de la misma (con contraseña encriptada). Si se presiona el botón **ver contraseña**, se pedirá una huella y al ser correcta el usuario podrá ver su contraseña.

Si se desliza una contraseña hacia la derecha, la misma se borrara del sistema. El sistema contiene un algoritmo que notifica la eliminación de una contraseña y espera unos segundos para borrarla definitivamente, dando así la posibilidad de que el usuario se retracte de lo hecho y deshaga el borrado.

También a diferencia del prototipo inicial, por motivos de implementación se implemento un botón en la barra inferior con forma de nube la cual como fue ya mencionado en el documento, al presionarlo se guardaran todas las contraseñas en la nube para luego ser analizadas.

Por ultimo se encuentra el botón de escáner, cuyo propósito sera mencionado mas

adelante.

Por motivos de demostrar la utilización de encriptado en la versión inicial, se decidió mostrar en pantalla el mensaje encriptado pero para próximas versiones lo mejor sera simplemente encriptar la contraseña y ocultar la contraseña mediante puntos para así ocultarla de forma definitiva.

5.2.4. Notas

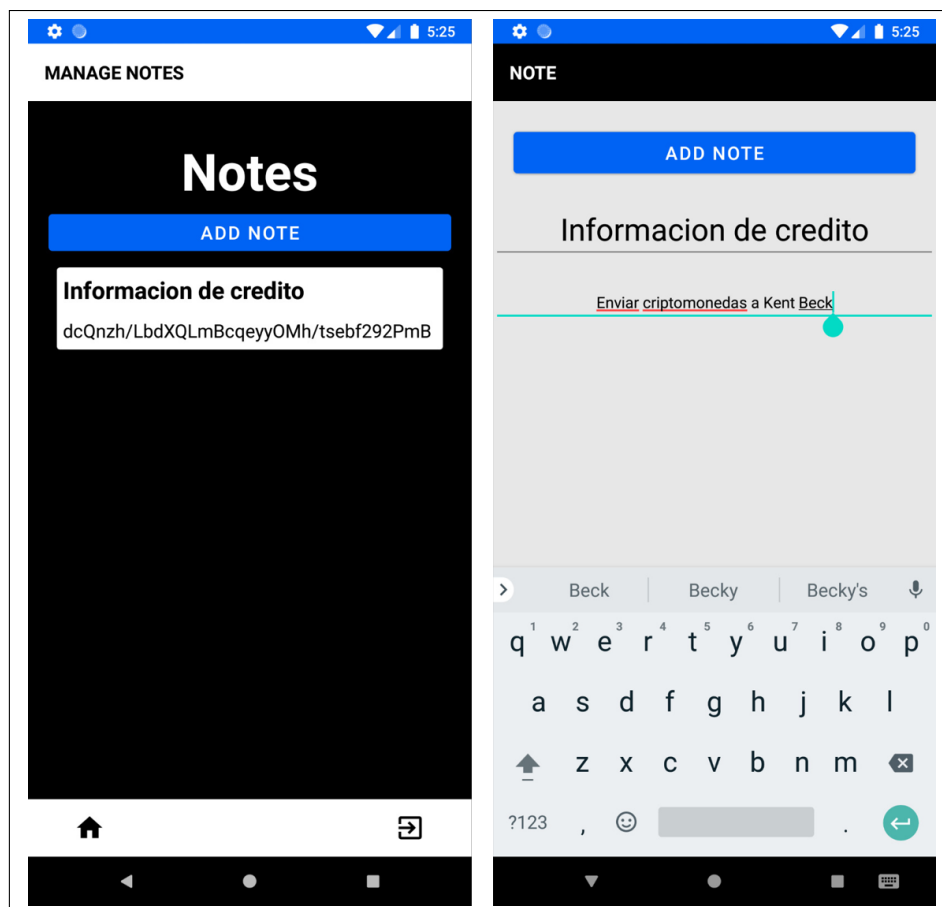


Figura 5.8: Ventanas de notas

La pantalla de notas es muy similar a la de contraseñas. Se permite la inserción, actualización y eliminación de notas. Al crearlas se encriptan y cuando se desea editarlas es necesario primero descryptar mediante la huella.

5.2.5. Scan de qr

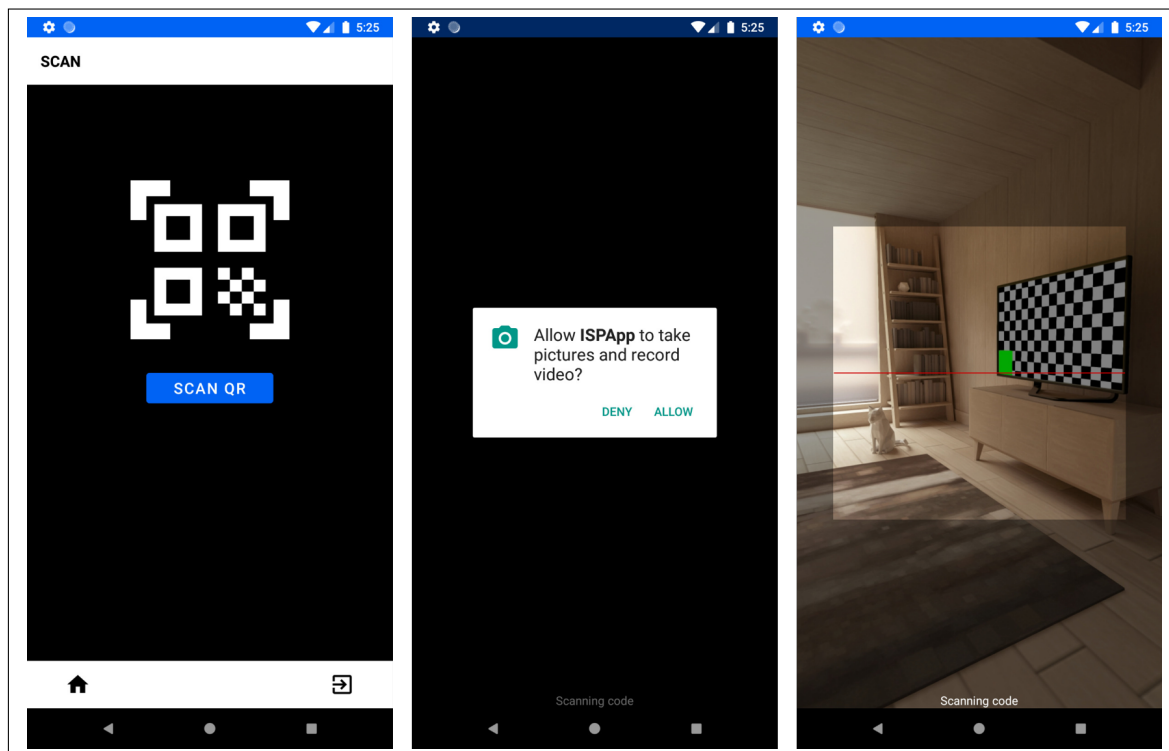


Figura 5.9: Ventanas de scanner

Para la pantalla de escáner, no se pudo cumplir la funcionalidad deseada en el prototipo inicial. Esta nueva versión permite leer un código qr con la cámara del dispositivo. Este código qr contiene una contraseña que al ser leído se mostrara como una contraseña a agregar.

Para que sea mas claro, esta funcionalidad se demostrara en la demo y además se brindara a continuación un vídeo en el cual se podrá ver como funciona la misma.
<https://youtu.be/5WRgIXPupm4>

5.2.6. Exportación de contraseñas

Como fue mencionado en la implementación del backend y en el protitopo esta pantalla se encarga de la exportación de las contraseñas.

A diferencia del prototipo, por motivos de funcionalidad y prolijidad, decidimos que lo mejor seria que el usuario ingresara a una pantalla para presionar el botón de exportar contraseñas, en vez de exportar desde la pantalla principal.

Como se ve en las imágenes, al presionar el botón exportar, el dispositivo informara la exportación de las mismas y se descargarán automáticamente en el dispositivo todas las contraseñas guardadas. A futuro sera necesario agregar algún sistema de autenticación antes de que el usuario exporte las contraseñas.

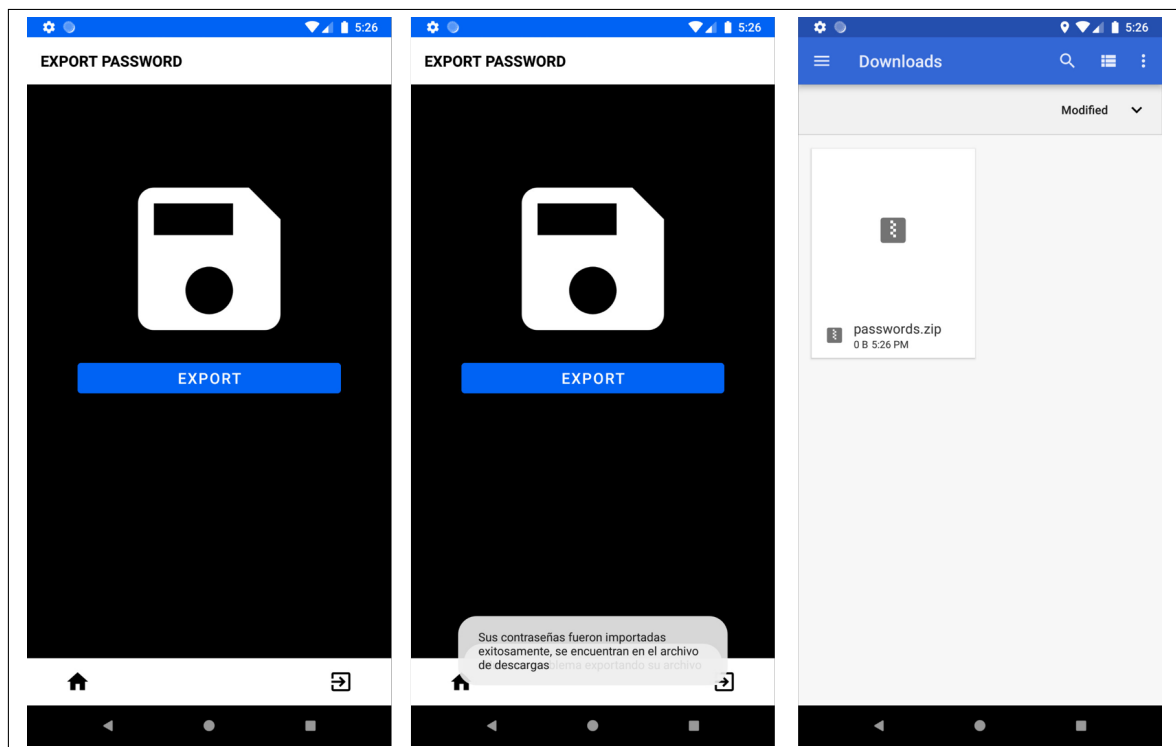


Figura 5.10: Ventanas de exportación de contraseñas

5.2.7. Análisis de seguridad

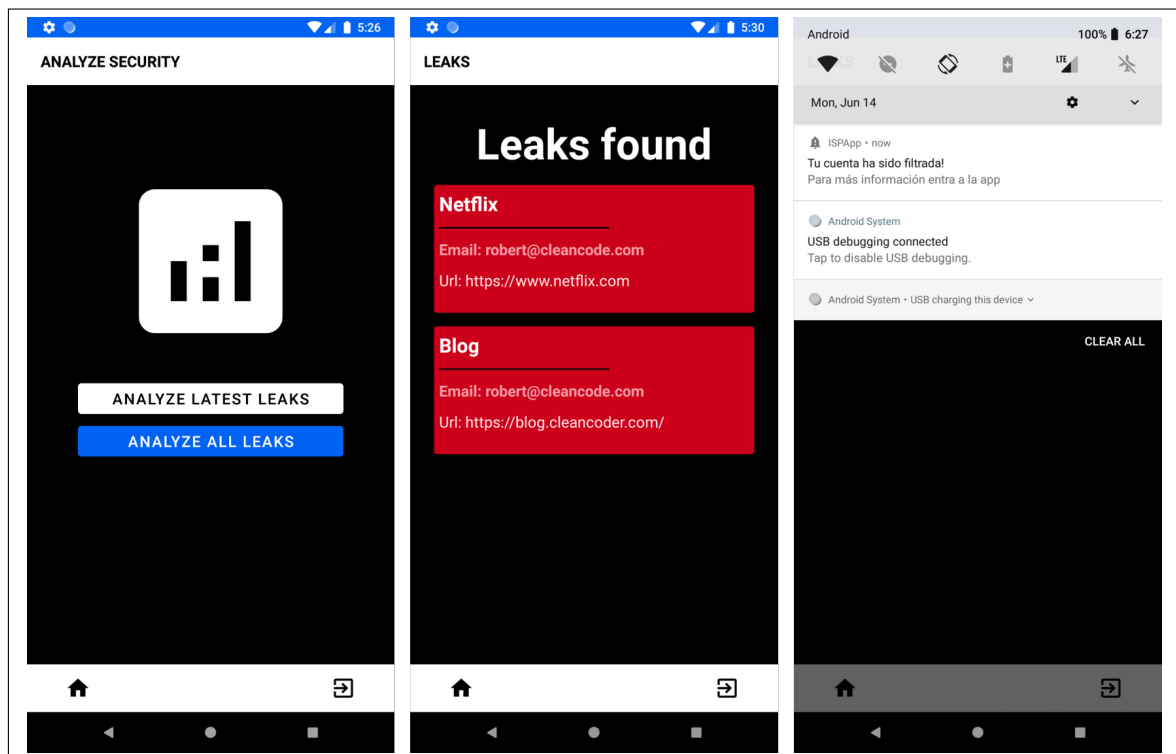


Figura 5.11: Ventanas de análisis de seguridad

Para esta pantalla se agregaron nuevas funcionalidades las cuales fueron el análisis de de las ultimas filtraciones y el análisis de todas las filtraciones históricamente.

Al presionar la ultimas filtraciones, se podrán ver en pantalla y aparecerá una notificación con las ultimas filtraciones analizadas. Esta metodología fue utilizada por motivos de facilidad de implementación. La misma como ya fue mencionado sera mejorada en próximas versiones.

Por ultimo al presionar el botón de analizar todas las filtraciones, se mostraran el historial de todas las filtraciones encontradas. Como se ve en la imagen del medio, se muestra la pagina la cual fue filtrada, el mail y el link de la misma.

6. Tecnologías utilizadas

6.1. Backend

Para realizar el backend utilizamos el lenguaje de programación C# en conjunto del framework .NET Core, como se solicita en la letra del proyecto. Para la base de datos usamos como motor de esta a Entity Framework Core y como base de datos a SQL Server 2014.

Para la creación de la web api REST, se hizo uso de del framework de código libre y multiplataforma ASP.NET Core. Para ser más exhaustivos en las pruebas de la web api y sin necesidad de tener un servicio que consuma la web api, usamos Postman, que simula ser un cliente y permite enviar diferentes requests, mostrando así de forma más descriptiva el funcionamiento del sistema.

6.2. Frontend

Para el desarrollo del frontend y del servicio que consume nuestra web api, usamos el IDE Android Studio. Se optó por el lenguaje Java para desarrollar, debido a que es la primera vez que realizamos un proyecto mobile, y consideramos que agregarle la complejidad de aprender también un nuevo lenguaje (Kotlin) no era apropiado

Se realizó una base de datos en el frontend, que guarda datos importantes y permite que la mayoría de las funcionalidades de la aplicación no se caigan en el caso que el usuario se encuentre desconectado de internet. Para realizar esta base de datos, se utilizó ROOM, que se basa en SQLite y es más simple para desarrollar.

Como cliente REST para nuestra web api, se hizo uso de Retrofit y para poder enviar notificaciones en tiempo real hacia los dispositivos de los usuarios, se utilizó Firebase Cloud Messaging, una solución de mensajería multiplataforma provista por Google.

6.3. Proceso

Para mantener un orden y poder seguir SCRUM de manera correcta, se decidió usar Trello como Task Board, y Discord para poder tener las videollamadas correspondientes a las reuniones establecidas.

Como repositorio del código se hizo uso de GitHub, de manera de poder trabajar todos los desarrolladores de manera independiente y al mismo tiempo.

Para realizar la documentación, se utilizó overleaf utilizando latex, en el cual todos podemos intervenir y modificar el texto en simultáneo.

Para realizar los mockups antes presentados de la aplicación, se hizo uso de la página Adobe XD.

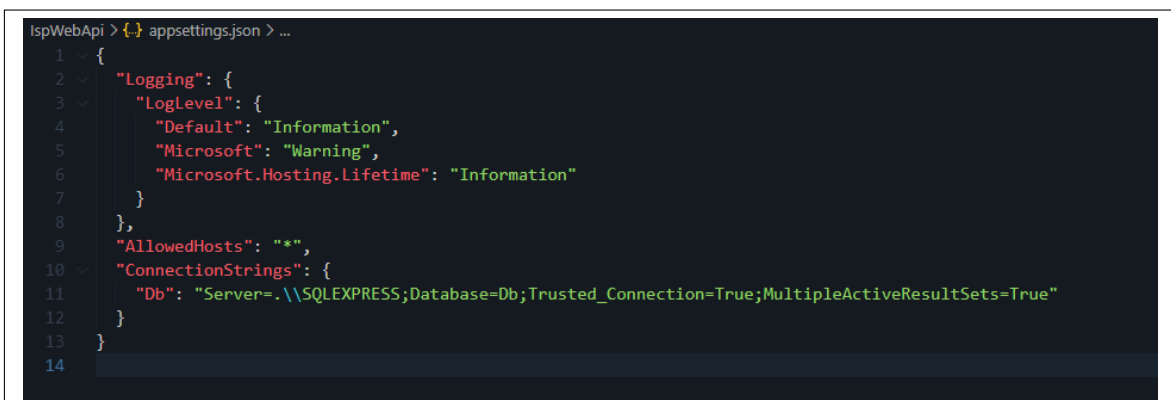
7. Guía de instalación

7.1. Requisitos API

- SQL Server express edition
- .Net Core 3.1 con SDK y Runtime
- Tener instalado entity framework en nuestra terminal (correr: dotnet tool install --global dotnet-ef)
- Tener una carpeta llamada Temp en la base de nuestro disco C.

7.2. Modificando elementos del runtime

Una vez cumplidos nuestros requisitos podemos comenzar la configuracion de la webapi. Para esto debemos buscar los elementos que acompañan a esta en su carpeta (ya sea en su forma de release o base). Estos son **appsettings.json** para editar la configuración del string de conexion a sql server o **launchSettings.json** para modificar la configuracion de la url donde se corre la aplicacion (esto no es un deploy).



```
IspWebApi > { } appsettings.json > ...
1 {
2   "Logging": {
3     "LogLevel": {
4       "Default": "Information",
5       "Microsoft": "Warning",
6       "Microsoft.Hosting.Lifetime": "Information"
7     }
8   },
9   "AllowedHosts": "*",
10  "ConnectionStrings": {
11    "Db": "Server=.;\\SQLEXPRESS;Database=Db;Trusted_Connection=True;MultipleActiveResultSets=True"
12  }
13 }
14
```

Figura 7.1: Configuracion del string de conexion


```

{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:49629",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IspWebApi": {
      "commandName": "Project",
      "launchBrowser": true,
      "launchUrl": "swagger",
      "applicationUrl": "http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}

```

Figura 7.2: Configuración de ip host

7.3. Corriendo la api por primera vez

La forma más rápida es ejecutar el siguiente comando parados en la misma webapi desde nuestra terminal favorita:

```
dotnet ef database update -p ..\DataAccess\
```

Luego de esto ejecutamos el siguiente comando para que corra nuestra api

```
dotnet run
```

De no contar con el código base y solo estar trabajando con la forma de release se

brinda un respaldo .bak para la base de datos. Para restaurarla hacemos lo siguiente:

- Abrimos nuestro SQL Management studio
- Entramos como usuario al servidor .\SQLEXPRESS
- Hacemos click derecho en nuestras bases de datos y elegimos restaurar una
- Elegimos desde archivo y seleccionamos nuestro .bak brindado

7.4. Corriendo la aplicación desde el emulador

La conexión a nuestra api es realizada dentro del paquete webApiServices/IpManager. Aqui se mantiene como constante la IP en la que esta alojada nuestra API. Por defecto tenemos la direccion: `http://10.0.2.2:5000/api/`. La misma es tomada por el emulador del dispositivo android y traducida como `localhost:5000/api`. Esto es importante ya que el dispositivo virtual técnicamente no corre en la misma red que nuestra api hasta utilizar ese *truco*.

Ahora si se realiza deploy de la api se ha de sustituir esta ip por la apropiada.

7.5. Notas finales

No se entró en detalle sobre elementos relacionados al deploy de la webapi ya que la misma puede ser levantada desde diversos motores como ISS, Ngix u similares en los que no vale la pena entrar en detalle. Queda a cargo del lector determinar que forma de ejecucion le es conveniente para la situación que requiera.

8. Anexo

ExportList		
GET	/api/exportedLists/zip	
Obtains users accounts and zips them in a base64 file with a password protection (the password is the user's email)		
Parameters		Try it out
No parameters		
Responses		
Code	Description	Links
200	Ok: returns base64 string	No links
401	No token provided	No links
403	Invalid token	No links
500	Internal server error: the server or database may be having problems	No links

Figura 8.1: Endpoint para la exportación en formato zip

GET	/api/leaks/new	
Obtains the new leaks (not sent before) found in databreaches for a given user, determined by the user auth token		
Parameters		Try it out
No parameters		
Responses		
Code	Description	Links
200	Ok: returns new leaks list	No links
401	No token provided	No links
403	Invalid token	No links
500	Internal server error: the server or database may be having problems	No links

Figura 8.2: Endpoint para la obtención de nuevas filtraciones

GET

/api/leaks/all

Obtains the all leaks (even those that where sent before) found in databreaches for a given user, determined by the user auth token

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Ok: returns all leaks list	No links
401	No token provided	No links
403	Invalid token	No links
500	Internal server error: the server or database may be having problems	No links

Figura 8.3: Endpoint para la obtención de todas las filtraciones para un usuario

POST

/api/leaks/dummy

Allows the user to post a dummy leak for a given user, in final build this should be deleted and use a real leaks API like HavelbeenPwnd

Parameters

Try it out

No parameters

Request body

application/json

Example Value | Schema

```
{
  "site": "string",
  "url": "string",
  "email": "string"
}
```

Responses

Code	Description	Links
200	Success	No links
201	Created	No links
401	No token provided	No links
403	Invalid token	No links
500	Internal server error: the server or database may be having problems	No links

Figura 8.4: Endpoint temporal para la creación de una filtración

POST

/api/sessions

Performs login with username and password

Parameters

Try it out

No parameters

Request body

application/json

Example Value | Schema

```
{
  "email": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	Ok: returns auth token	No links
400	Bad request: invalid username or password	No links
500	Internal server error: the server or database may be having problems	No links

Figura 8.5: Endpoint para el inicio de sesión

POST

/api/users

Creates an user

Parameters

Try it out

No parameters

Request body

application/json

Example Value | Schema

```
{
  "deviceId": "string",
  "accountModels": [
    {
      "username": "string",
      "password": "string",
      "site": "string",
      "email": "string"
    }
  ],
  "mainEmail": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	Success	No links
201	Created: user information	No links
400	Bad request: validation errors	No links

Figura 8.6: Endpoint para la creación de un usuario

PUT

/api/users/accounts

Updates an user accounts. Needs to be the new list of all the accounts the user has.

Parameters

Try it out

No parameters

Request body

application/json

Example Value

Schema

```
{
  "accountModels": [
    {
      "userName": "string",
      "password": "string",
      "site": "string",
      "email": "string"
    }
  ]
}
```

Responses

Code	Description	Links
200	Ok: user accounts updated	No links
400	Bad request: validation errors in account model	No links
401	No token provided	No links
403	Invalid token	No links

Figura 8.7: Endpoint para la actualización de cuentas de un usuario

Bibliografía

- [1] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 2008.
- [2] C. Code, “Clean code : Robert martin site.” [Online]. Available: <http://cleancoder.com/products>
- [3] “Las 10 heurísticas de nilsen.” [Online]. Available: <https://www.uifrommars.com/10-reglas-heuristicas-como-aplicarlas/>
- [4] “Clean code developer check.” [Online]. Available: <https://github.com/dev-aritra/clean-code-developer-checklist>
- [5] “Firebase cloud messaging.” [Online]. Available: <https://firebase.google.com/docs/cloud-messaging>
- [6] “Guia de android studio.” [Online]. Available: <https://developer.android.com/studio/intro>