Desenvolupament amb MAUI: OmronDrives

Saül Gordillo Garcia

25 de juny de 2024

Resum– OMRON és una empresa multi-disciplinària que destaca en diferents àrees, una d'elles és l'Automatització Industrial. En aquest projecte es desenvoluparà la segona versió d'una aplicació MAUI per a Windows i Android que sortirà al mercat al Juliol. OmronDrives és una aplicació ja existent que serveix per fer Backups de Drives del tipus M1 (Propietat d'OMRON)[8], els Backups són XMLs que guarden valors de paràmetres, que després es poden escriure al Drive. A la segona release d'aquesta aplicació s'incorporarà la possibilitat de compartir aquests Backups juntament amb millores de UI i UX. A més a més es desenvoluparà un Proof of Concept (PoC) de la possibilitat d'afegir sincronització Cloud a l'aplicació.

Paraules clau– MAUI, Microsoft, OMRON, Automatització Industrial, Drives, Programmable Logic Controller (PLC), Desenvolupament de software, Android, Windows.

Abstract– OMRON is a multidisciplinary company that excels in various areas, one of which is Industrial Automation. This project involves developing the second version of a MAUI application for Windows and Android that will be released to the market in July. OmronDrives is an existing application used for backing up he parameters for M1 Drives (owned by OMRON)[8]. The backups are XML files that store parameter values, which can then be written to the Drive. In the second release of this application, the ability to share these backups will be incorporated along with improvements in UI and UX. Additionally, a Proof of Concept (PoC) will be developed for the possibility of adding Cloud Synchronization to the application.

Keywords– MAUI, Microsoft, OMRON, Industrial Automation, Drives, Programmable Logic Controller (PLC), Software Development, Android, Windows.

1 Introducció

Per entendre de què serveix OmronDrives[10], i perquè és vol treure la segona versió primer s'han d'entendre diverses coses. En primer lloc, què és un Drive? Un Drive és una peça de hardware que permet comunicar-se amb un motor. Sense un Drive, el motor solament podria moure's una velocitat constant, gràcies al Drive aconseguim modificar la velocitat, el torque, potència i fins i tot millorar les vibracions...

Imaginem que una empresa vol construir una cinta transportadora, que mou endevant uns objectes i para cada metre. En aquesta posició els objectes poden rebre accions variades, com veiem els motors han d'anar connectats a uns Drives, donat a què han de variar en velocitat. Els Drives es poden utilitzar per afinar els parametres que rep el motor, sense afinar els paràmetres abans, segur que la velocitat no és l'esperada, i segur que no es para on s'hauria de parar.

Una vegada hem entès quin és l'element principal amb el que treballa la nostra aplicació hem d'entendre perquè a vegades, hi ha clients que prefereixen utiltizar OmronDrives en comptes de SysmacStudio[9], que és la aplicació insígnia per configurar Drives, aquesta aplicació permet fer tot tipus d'operacions (És amb aquest software amb el que pots configurar els Drives per fer rutines, afinar els paràmetres, fer traces, etc) no solament amb els Drives de tipus M1[8], sinó que amb tots els Drives d'OMRON.

Imaginem, que s'afinen els paràmetres del primer Drive de tota cinta transportadora (O tenir el Drive ja configurat), el client, podria afinar els paràmetres de tots els Drives amb Sysmac Studio, o podria exportar els paràmetres amb OmronDrives i escriure-ls als Drives restants.

• E-mail de contacte: saul.gordillo@autonoma.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Ruben Martínez Vidal (Àrea de Ciències de la Computació i Intel·ligència Artificial)
• Curs 2023/2024

L'aplicació solament funciona amb el Drive amb més precisió de tota la línea de OMRON, el inverter M1[8]. Donat a què els clients han donat bon feedback de l'aplicació s'ha decidit, treure una segona versió al Juliol.

1.1 Estat de l'art

Abans de parlar de la segona versió de l'aplicació, anem a veure com es veu actualment:

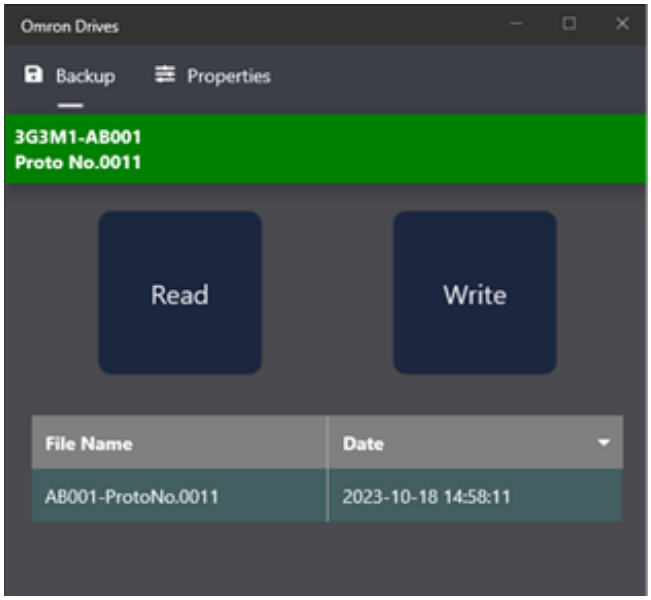

Fig. 1: OmronDrives 1.0

Com podeu veure, tenim dues pàgines, la de Backup i la de Properties:

- A la de Backup, veiem el Model i el nom del Drive, juntament amb els dos botons per escriure (Enviar els paràmetres als Drives) i llegir (Guardar els paràmetres que hi han actualment). Més abaix tenim la llista de Backups, a la columna de l'esquerra tenim el nom del fitxer i a la de la dreta la data on s'ha fet el backup.
- A la pàgina de paràmetres podem trobar informació important del Drive, com la versió del seu Firmware.

1.2 Objectius

Gràcies al feedback rebut per part dels clients, s'han près les següents decisions de disseny, que seran els objectius a aconseguir:

- Renovar el UI de l'aplicació, utilitzant els colors característics de OMRON.
- Afegir una nova funcionalitat: Una vista de Settings on es pugui activar/desactivar els avisos de cautela.
- Afegir la funcionalitat de Import/Export, és a dir, poder Importar arxius de paràmetres, poder exportar-los (guardar-los l'explorador d'arxius)
- Afegir la funcionalitat de Share als backups ja existents (Que salti el popup de Share tant en Windows com en Android). Més endevant parlarem més en detall (2.1).

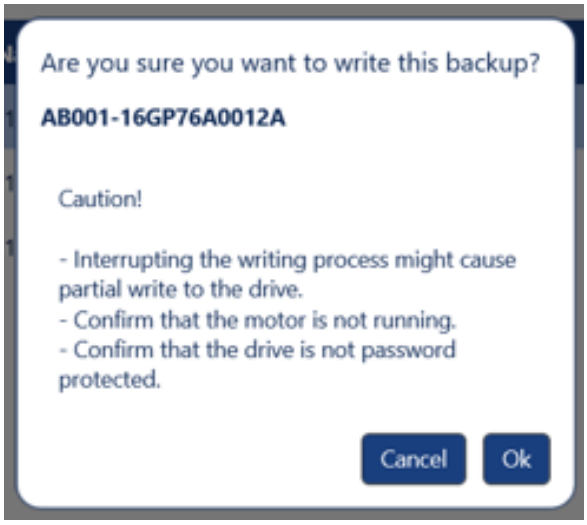

Fig. 2: Avís de cautela a l'aplicació

Més enllà, hi ha diverses característiques que no estan previstes que surtin al mercat, que són desenvolupades solament per el TFG:

- Iniciar sessió tant en Google com en Microsoft. Això estarà en la vista de Settings.
- Cloud Synchronization: Afegir la possibilitat de sincronitzar el dispositiu amb un servei Cloud tant de Google com de Microsoft. Més endevant parlarem més en detall (2.2).

En resum, hi han dos punts principals, el **Share Functionality** i el **Cloud Synchronization**. Abans d'entrar en detall en les especificacions, parlarem de com s'organitza el document.

Primer parlarem del marc de desenvolupment, és a dir el framework i els patrons de disseny que s'han seguit, després de les especificacions de les dues funcionalitats en detall, també parlarem de l'organització del projecte, i finalment dels resultats i les conclusions.

1.3 Metodologia

Aquest projecte es desenvoluparà íntegrament en C#[5], concretament utilitzant el framework de MAUI[4].

1.3.1 MAUI

MAUI (Multi-platform Application User Interface), és una tecnologia desenvolupada per Microsoft que permet als desenvolupadors crear aplicacions multiplataforma amb una única base de codi. Per què MAUI[13]? Molt senzill, perquè gràcies a tenir una única base de codi, és molt més fàcil de mantenir, i de desenvolupar interfícies d'usuari consistents[14].

Tot i que l'elegit hagi sigut MAUI, hi ha d'altres tecnologies que permeten desenvolupar aplicacions multiplataforma, una de les quals podria ser Flutter, que ja en tenia experiència, o Kotlin, desenvolupada per Google, o React Native, que pot ser és la més utilitzada avui en dia. He escollit MAUI donat a què utilitza el lenguatge de programació C#, i és un dels que més experiència tenia, donat a què portava treballant un any en aquest lenguatge.

Tot projecte ha de seguir un (o diversos) patrons de disseny, com TDD (Test Driven Development), MVC (Model-View-Controller) o en aquest cas, DDD (Domain Driven Design)[3].

1.3.2 Patró de disseny: Domain Driven Design

Aquest patró es centra en la comprensió i modelatge del domini de negoci per crear sistemes més eficients, i sobretot més alineats amb les necessitats de l'usuari. Aquí tenim un petit esquema per entendre-ho millor:

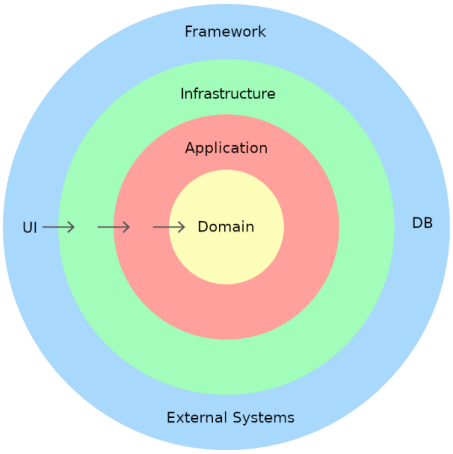

Fig. 3: Esquema de DDD

Com veiem a la figura, aquest patró divideix el nostre projecte en tres:

- **Application Layer**: Actua com intermediari entre la capa de Domain i Infrastructure, implementa la lògica de l'aplicació.
- **Domain Layer**: És on resideix la lògica de negoci. És independent de la tecnologia i solament s'enfoca en la comprensió del domini de negoci.
- **Infrastructure Layer**: Gestiona els detalls tècnics del sistema, com les I/O, la persistència de dades etc etc.

Al desenovlupament però, solament he hagut d'implementar funcionalitats a la capa d'aplicació, on l'estructura que seguia el projecte era la del MVVM (Model-View-ViewModel). En aquest patró tenim tres capes principals:

- Al View, tenim la part de interfície d'usuari, en aquest cas un XAML.
- ViewModel és la classe que implementa les propietats i comandes que la View pot utilitzar, on a vegades en comptes de tenir les comandes al ViewModel, si és massa gran es genera un CommandManager específic per la classe.
- Per últim tenim els Models, que són les classes que emmagatzemen les dades que utilitza el ViewModel.

És a dir, l'aplicació en general utilitza el patró DDD, però per implementar la capa d'aplicació, s'utilitza el patró MVVM.

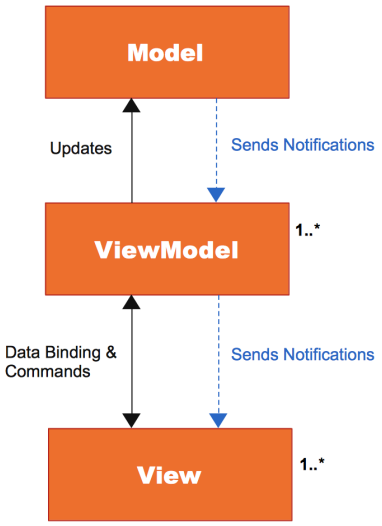

Fig. 4: Esquema de MVVM

2 Especificacions i desenvolupament

En aquesta secció parlarem de les especificacions finals del projecte, i temes de desenvolupament que són importants a mencionar.

2.1 Share Feature

En aquest punt del projecte, aquesta funcionalitat està acabada amb el Unit Testing implementat, a falta del System Test, que és extern a mi, i és farà probablement al Juliol.

2.1.1 Especificacions

Les especificacions d'aquesta Feature no han sigut sempre les mateixes, de fet han anat canviant constantment. Primerament es volia que l'aplicació pogués Importar / Exportar Backups, és a dir, poder agafar un Backup de l'explorador d'arxius, i tenir-ho a l'aplicació (Importar), i poder guardar un Backup de l'aplicació a l'explorador d'arxius (Exportar). Una vegada es va aconseguir això, es va pensar en afegir l'opció de Share a Android, el típic popup que surt quan li dones a compartir qualsevol arxiu. Quan es va aconseguir, vam preguntar-nos si val la pena afegir-ho a Windows també, i vam arribar a la conclusió que amb la incorporació de Windows 11, sí valia la pena.

2.2 Cloud Synchronization PoC

Aquesta Feature és la més extensa del treball, està actualment totalment acabada, menys com ja he comentat anteriorment el System Test, la idea general és tenir una forma de sincronitzar els Backups amb una solució Cloud, per així poder tenir diversos dispositius amb la mateixa informació.

2.2.1 Especificacions

Les especificacions d'aquesta feature són senzilles: Primer de tot ha d'haver una forma d'inici de sessió a l'aplicació, tant amb Google com amb Microsoft, s'ha decidit que aquests botons siguin part de la View de Settings.

La segona part de les especificacions d'aquesta feature són les de sincronització, és vol que l'app tingui l'opció de poder pujar/baixar arxius dels diferentes serveis cloud. S'ha decidit que l'aplicació tingui dues formes de sincronització:

- Una activa: On s'ha creat un botó a la View de Backup que primer de tot puja tots els Backups que té el dispositiu al Cloud, i després es baixa tots per aconseguir que el Cloud i el dispositiu estiguin sincronitzats.

- I una passiva: On cada vegada que es llegeixin els paràmetres, aquest arxiu que es guarda al dispositiu també es pujarà al Cloud corresponent.

S'ha decidit per especificacions que es pugui iniciar sessió amb un o els dos serveis, és a dir, que podem fer tant sincronització amb Google, amb Microsoft i ambdós junts.

2.3 Desenvolupament

En aquesta secció m'agradaria recalcar algunes parts del desenvolupament que m'han semblat clau i crec que són importants mencionar.

Primer de tot voldria parlar de com he hagut de configurar tant Google Cloud[6] com Microsoft Azure Portal[7]. I com mentre feia això m'he adonat que la implementació serà molt diferent depenet del proveïdor.

Al començar aquesta part del desenvolupament, primerament vaig documentar-me extensament sobre com funcionen tant l'API de Google Drive, com la de One Drive, i les dues tenien un punt en comú, la necessitat d'iniciar sessió amb els seus serveis corresponents: Google i Microsoft, i la necessitat de registrar l'aplicació en els seus serveis Cloud: Google Cloud i Microsoft Azure Portal.

Aquí van començar els problemes, vaig començar el desenvolupament utilitzant la llibreria de Microsoft MSAL[2], que dona una implementació molt sençilla per a comunicar-se amb Microsoft Graph[12], l'API de Microsoft per a qualsevol servei que ells proveeixen, i per tal de mantenir la consistència entre els dos log ins, vaig decidir intentar implementar tots dos log ins amb aquesta llibreria, però investigant, vaig adonar-me'n que necessitava registrar la app de Google Cloud al Azure també, i la empresa no hem podia donar aquests privilegis, així que vaig haver de buscar una altre forma d'autenticar-me amb Google: Google.Apis.Auth[12].

La lògica amb Microsoft a Windows ha sigut mitjanament fàcil una vegada entès com funciona el procés d'iniciar sessió, he pogut utilitzar la llibreria de Microsoft MSAL[2] tant per Android com per Windows, cosa que altres llibreries no deixaven, o canviava massa la implementació, l'únic que canviava en la implementació era que havia d'afegir un Activity, és a dir un punt d'entrada per interactuar amb l'usuari, i algunes linees per acceptar l'autentificació quan es crea l'aplicació, en si, no és difícil, però donat a què hi havia tant poca documentació, vaig acabar perdent masses hores en això.

Amb Google no ha sigut tant fàcil, poso un exemple: Amb Microsoft, jo simplement especifico una URL, per exemple: "myonedrive/folder1/file1", i automaticament em crea "folder1"si no està creada i si "file1"ja existeix, la sobreeescriu. A Google he hagut crear la lògica per fer totes les comporbacions manualment: Si existeix la carpeta OmronDrives, si el fitxer que estic afegint existeix, si existeix, busca el que té el mateix nom i borra-ho, etc.

Abans de seguir, hauria d'explicar un parell de conceptes claus del desenvolupament amb Android, un és l'Activity d'una aplicació, a Windows un programa s'inicia a partir del *main()* o alguna funció similar, en canvi a Android s'inicia una instància d'una Activity invocant el seus mètodes. Un altre concepte clau és el de CodeReciever, un CodeReciever gestiona com retornen els tòkens d'autorització d'un servei, en aquest cas de Google, a la teva aplicació, cosa que no és trivial.

Una vegada aclarat aquests conceptes podem dir que la dificultat no solament va ser aquesta, sinó que la implementació amb Android va canviar de forma considerable, no solament vaig haver de modificar l'Activity principal d'Android, sino que vaig haver d'implementar el CodeReciever de l'API de Google.

2.3.1 Unit Tests

L'únic Testing que he fet és el Unit Testing, que sempre és responsabilitat del desenvolupador. Aquest Unit Testing no s'ha implementat seguint metodologies com el TDD (Test Driven Development), sinó que cada vegada que implementava una funció feia els Unit Tests corresponents.

L'unit testing va incloure la comprovació de la validació d'entrades, l'exactitud de les sortides i el maneig adequat de corner cases. Es va utilitzar el framework MSTest[18], que va facilitar la creació de proves automatitzades i va proporcionar informes detallats sobre la cobertura de les proves. Es van emprar mocks gràcies a la llibreria JustMock, assegurant entorns de prova aïllats. Es van adoptar pràctiques d'integració contínua, executant proves unitàries amb cada commit de codi per detectar i abordar problemes de manera ràpida. Com a resultat, l'aplicació va demostrar una alta estabilitat i rendiment, contribuint a una experiència d'usuari òptima.

L'equip de QA de l'empresa s'encarregarà en un futur pròxim de fer el testing real, tant exploratory com integration testing i tot el que faci falta per assegurar que l'aplicació surti al mercat amb el nivell esperat.

2.3.2 Android vs Windows

Un dels reptes més difícils de superar en aquest projecte van ser les diferències en codi entre Android i Windows. Encara que ja hagi comentat algunes de les diferències més tècniques, voldria comentar perquè això va ser un problema i va afectar a la planificació.

En la primera part del desenvolupament no vaig haver de fer gaires canvis entre plataforma gràcies a les pròpies integracions de MAUI[4]. El problema va arribar quan vaig començar la part de crides a APIs tant amb Google com amb Windows. Com en tota la primera part del desenvolupament no vaig haver-me de preocupar per les diferències en el desenvolupament entre plataformes, no vaig pensar amb antel·lació que aquí sí hi hauria canvis. Vaig començar el desenvolupament centrant-me molt amb Windows, juntament amb els seus propis problemes i en comptes d'anar provant el desenvolupament amb Android, fins que no vaig acabar tota la implementació, no vaig passar a Android.

Al que acabo de comentar, se li va sumar la falta de documentació i exemples de implementacions similars. MAUI[4] és una tecnologia relativament nova, que ve d'una tecnologia més vella, Xamarin[17], va ser introduida fa menys de 3 anys, concretament al Novembre de 2021. Això fa que encara que facis la recerca necessària, no hi hagi exemples concrets de MAUI[4], i acabes trobant exemples de fa 10 anys de Xamarin, on la implementació pot ser semblant, però també enganyossa.

Aquesta falta de documentació (tant a la part de Google com a la de Microsoft) va provocar que intentés seguir diversos camins d'implementacions que no eren correctes, i clar, havia de mantenir la consistència entre les implementacions de Android i Windows, per tant vaig haver de posar un ènfasis extraordinari en trobar una possible solució a aquest cas.

Al final entre implementacions de Xamarin[17], i veure codi en altes llenguatges de les APIs que utilitzava, juntament amb alguns fòrums, vaig aconseguir la implementació.

3 Metodologia

Durant tot aquest projecte, s'ha seguit la metodologia àgil SCRUM[1].

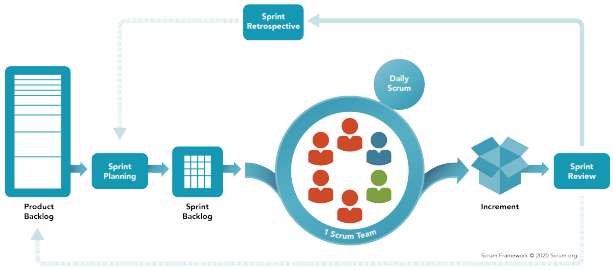

Fig. 5: Diagrama de SCRUM

SCRUM és una metodologia àgil que proporciona un marc de treball per el desenvolupament de productes complexos. Les principals característiques d'aquesta metodologia són: la transparència, inspecció i adaptació, i es centra en l'entrega incremental de funcionalitats que donin valor afegit al producte.

Per seguir aquesta metodologia, primer vam dividir el projecte en Features i User Stories per afegir al Product Backlog, on abans de començar cada Sprint (són de 2 setmanes) s'organitza un Sprint Planning per escollir d'aquest Backlog, que es desenvoluparà en aquell Sprint.

Quan s'acaba un Sprint, es realitza el Sprint Review, on es discuteix si els objectius s'han assolit satisfactòriament i si no s'han assolit el perquè. Finalment en el Sprint Retrospective cadascú és lliure de donar sugerències, coses a millorar o coses que s'han fet bé, i al Meeting es repassen totes.

3.1 Planificació

Encara que OMRON tingui una planificació interna, jo he desenvolupat un Diagrama de Gantt per planificar-me no solament el desenvolupament, sinò totes les tasques necessàries per aconseguir tots els objectius del TFG.

Backup Sharing				
Feature breakdown	Saül Gordillo	100%	2/26/24	2/26/24
Develop Share Specifications	Saül Gordillo	100%	2/27/24	2/29/24
Share Unique File	Saül Gordillo	100%	2/29/24	4/4/24
Import Unique File	Saül Gordillo	100%	2/29/24	4/4/24
Import Multiple Files OUT OF SCOPE	Saül Gordillo	0%	5/7/24	5/23/24
Share Multiple Files OUT OF SCOPE	Saül Gordillo	0%	5/7/24	5/23/24
Cloud Synchronization				
Analyze viability of Cloud storage solutions	Saül Gordillo	100%	4/8/24	4/18/24
Analyze viability of OMRON cloud solution	Saül Gordillo	100%	4/22/24	4/26/24
Define Cloud Synchronization specifications	Saül Gordillo	100%	4/29/24	5/2/24
Develop Cloud Solution Proposal	Saül Gordillo	100%	4/10/24	4/26/24
[*] Develop Solution	Saül Gordillo	100%	4/26/24	6/12/24
Documentation				
First paper delivery	Saül Gordillo	100%	2/26/24	3/1/24
Second paper delivery	Saül Gordillo	100%	4/11/24	4/15/24
Third paper delivery	Saül Gordillo	100%	5/16/24	5/20/24
Final paper delivery	Saül Gordillo	80%	6/1/24	6/15/24
Presentation proposal	Saül Gordillo	0%	6/13/24	6/24/24
TFG Delivery	Saül Gordillo	0%	6/17/24	6/30/24

Fig. 6: Planificació de les tasques

Aquí veiem com vaig dividir les Features en tasques, per poder centrar-me en objectius més petits i anar millorant progressivament l'aplicació. Com podem veure, el primer mes i mig, ha sigut totalment centrat en el desenvolupament de les features necessàries per la release de Juliol de l'aplicació. A Abril, el desenvolupament es centra molt més en la Feature de Cloud Synchronization, que no solament és implementar-ho sinò també molta teoria i documentació, i Maig i Juny el focus està en els últims detalls de la implementació i en la bona documentació. De fet els últims mesos la majoria del temps l'he passat navegant la xarxa en busca de solucions als problemes d'implementació que he tingut, la majoria del "delay"que ha portat a allargar el desenvolupament d'aquesta Feature ha sigut això, la manca de documentació i exemples de la implementació del que es volia fer.

Com veiem a la figura, les tasques de Import Multiple File i Share Multiple File s'han declarat out of scope, això és perquè ha hagut alguns problemes amb l'implementació de la Cloud Synchronization feature i la data d'entrega s'aproximava, he decidit no implementar-les.

4 Resultats

Com ja s'ha comentat anteriorment, els objectius principals d'aquest treball son implementar tant les features que sortiran al mercat com les que solament són de recerca. Tenint això en compte anem a veure com han quedat després dels canvis, i després comentar els resultats:

4.0.1 Share feature

El desenvolupament d'aquesta feature, va ser bastant directe, les especificacions estaven clares (tret dels canvis) i a la implementació no van sorgir masses inconvenients, a continuació veurem com queda l'aplicació després dels canvis:

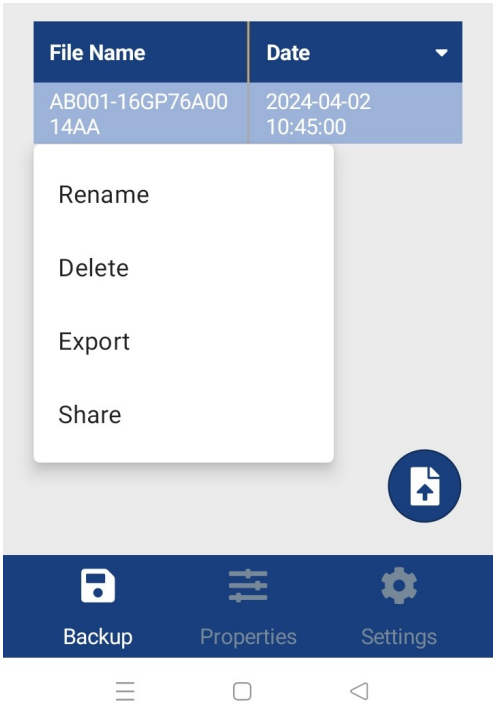

Fig. 7: View del menú a Android

Fig. 8: View del Backup en Windows

Abans de començar el projecte, es va rebre bastant feedback del client que deia que l'interfície d'usuari no estava a l'alçada del que hauria de ser per tant gràcies a aquest projecte l'aplicació ha experimentat una transformació notable en la seva interfície d'usuari, incorporant els colors distintius de l'empresa, que han aportat una estètica cohesiva i atractiva. A més, he dedicat una atenció meticulosa a l'experiència d'usuari, aconseguint una navegació intuïtiva i fluida. Aquestes millores farà que els usuaris puguin interactuar amb l'aplicació de manera més eficient i agradable.

4.0.2 Cloud Synchronization feature

Algunes parts de la implementació han sigut més difícils d'implementar que del que s'havia planejat, sobretot com ja he comentat la part de Log-in tant amb Google com amb Microsoft. A l'ànnex hi ha la imatge que representa el PoC d'aquesta feature (12), i a continuació teniu la implementació real i una foto de com estava l'aplicació abans d'iniciar aquest projecte.

Com veiem, hi haurien tres canvis essencials:

- Primer de tot, a la view de Settings, hem afegit l'opció d'iniciar sessió tant com amb Google com amb Microsoft.
- La sincronització es pot dividir en dos parts, una passiva i l'altre activa:
  - Primer, la sincronització passiva: Quan llegeixes un Backup, directament es puja al Cloud, en qualsevol dels serveis Cloud, és a dir si estàs loggejat amb ambdós, es pujarà als dos.
  - Després, hi haurà una sincronització activa: Abaix a l'esquerra, hi ha un botó per sincronitzar els Backups entre Cloud i dispositiu, és a dir, tenir tot a tots llocs, que també funciona en qualsevol dels serveis Cloud.

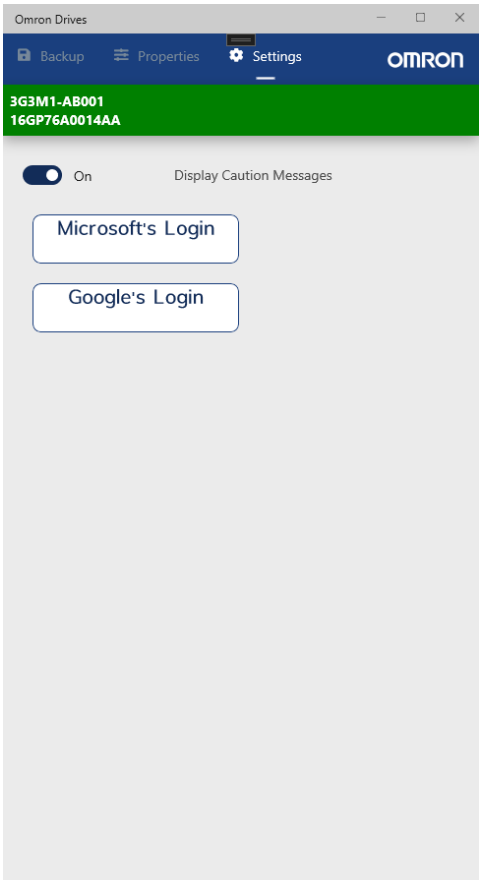

Fig. 9: View de Settings

Fig. 10: View de Backup quan estàs loggejat

Aquesta feature ha sigut acabada satisfactòriament, seguint estrictament les guies de desenvolupament proporcionades per les APIs de Google.Apis.Auth[12] i MSAL[2], he implementat autenticació robusta i gestió de permisos per garantir que només els usuaris autoritzats tinguin accés a les dades. L'ús de Google.Apis.Auth[12] ha assegurat una autenticació sòlida amb OAuth 2.0, mentre que l'API de Google Drive ha facilitat una integració segura per a l'emmagatzematge i la gestió de fitxers. Paral·lelament, amb MSAL[2] i Microsoft Graph[11], he garantit una autenticació segura i accés a les dades de Microsoft 365. Això ha resultat en una aplicació fiable, lliure de vulnerabilitats conegudes, que protegeix eficaçment la privadesa i la seguretat dels usuaris.

L'única cosa que està fora de les especificacions és el fet de que al Logout, en comptes de fer un Logout normal, s'avisa amb un popup a l'usuari de que l'aplicació es tancarà després de loggejar out. Això es deu a un bug persistent en el Logout de Microsoft, on a vegades, quan intentes reloggejar després d'haver tancat sessió, el programa dona una unhandeled exception. Aquest és un error que no dona gaire informació donat a què et diu que el teu debugger no soporta aquest tipus d'exceptions. Després de molt cercar a la web, vam decidir fer aquest workaround donat a què no sabiem perquè passava això.

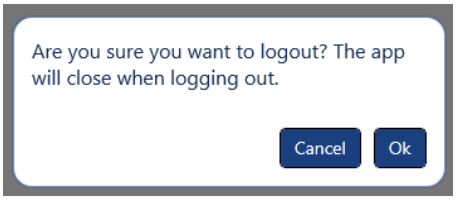

Fig. 11: View del popup que surt quan vols fer logout

5 Conclusions

Ara ens trobem al final del desenvolupament, després de 4 mesos de treball continu, després de més de 300 hores de treball, el desenvolupament ha acabat. S'han aconseguit els objectius, alguns de les especificacions han canviat, però el funcionament és l'esperat, i l'empresa està contenta.

En aquests 4 mesos, he pogut aprendre una tecnologia tant interessant com MAUI, amb un equip tant cohesionat com el meu. En aquests mesos he après com funciona WPF, he après com funcionen els Unit Tests en .NET[5], he pogut treballar amb les APIs de Google i Microsoft i els serveis Cloud de cada una d'aquestes empreses. He hagut de reunir-me amb companys de treball de arreu del món per poder aconseguir certs permisos. He après nous paradigmes de programació, i gràcies a la mentoria dels sèniors he après a pensar no solament com implementar una feature, sinó com implementar correctament i de la forma més flexible i correcte possible aquesta mateixa feature.

Després d'aquests mesos de desenvolupament, puc dir que he desenvolupat una passió pel .NET[5], i vull seguir treballant en incrementar els meus coneixements en aquest framework.

L'aplicació té molt potencial per millores futures, com ja he comentat amb anterioritat, OMRON té el software principal, Sysmac Studio[9], on es poden fer diverses operacions amb els Drives, òbviament, el futur de l'aplicació pasa per implementar algunes d'aquestes operacions. Els principals candidats són els següents:

- **Parameter Editor**: Ja he explicat que l'aplicació es basa en fer Backup dels paràmetres, per després compartir-los o escriure'ls, una de les millores futures podria ser poder editar aquests paràmetres en la pròpia aplicació. Ara mateix, si es vol editar alguns d'aquests paràmetres s'ha de obrir la configuració del Drive al Sysmac Studio[9].

- **Alarmes**: Apart dels paràmetres, els Drives tenen altres classes d'indicadors, com poden ser les Alarmes. Les Alarmes són els indicadors d'error/warning del Drive, per exemple, imagina que el Drive quan li apliques uns paràmetres té un motor, si tu li treus aquest motor, et sortirà un warning. O si un Drive estava connectat a una PLC, si el cable s'ha tret, encara que es torni a posar, sortirà un warning. En un futur l'aplicació podria ensenyar aquestes alarmes, i fins i tot resetejar-les.

- **Altres Drives**: Com ja he comentat abans, l'aplicació només està disponible per Drives del tipus M1[8], que són els últims Drives d'OMRON, en un futur l'aplicació podria soportar altres Drives, tant nous, com per exemple el J1 (Drive que s'està desenvolupant), com més vells com el 1S[15] entre d'altres.

- **Multiple Share/Import**: Òbviament, donat a què ha quedat out of scope en aquest desenvolupament, el poder compartir, importar i exportar múltiples backups, seria una de les prioritats al futur.

Per últim m'agradaria mencionar els beneficis d'utilitzar una metodologia àgil per el desenvolupament progressiu d'aplicacions. Gràcies a seguir els principis d'aquesta

metodologia he pogut organitzar-me de forma en la que encara que han hagut canvis en les especificacions i hi han hagut problemes en el desenvolupament he pogut mitigar-ho i aconseguir satisfactòriament els objectius.

REFERÈNCIES

[1] Prince Patni. “Scrum- Theory, Team, Roles, Artifacts, Events, Scrum Master.” Accedit 20 Abril, 2024. https://princepatni.com/blog/tech/scrum-theory-team-roles-artifacts-events-scrum-master/.

[2] Cilwerner. “Overview of the Microsoft Authentication Library (MSAL) - Microsoft Identity Platform.” Microsoft identity platform — Microsoft Learn. Accedit 20 Abril, 2024. https://learn.microsoft.com/en-us/entra/identity-platform/msal-overview.

[3] HiBit. “Domain Driven Design: Layers.” Accedit 20 Abril, 2024. https://www.hibit.dev/posts/15/domain-driven-design-layers.

[4] Microsoft. ”.NET Multi-Platform App UI (.Net Maui): .NET.” Microsoft. Accedit 20 Abril, 2024. https://dotnet.microsoft.com/en-us/apps/maui.

[5] BillWagner. “Guía de C#: Lenguaje Administrado de .NET.” Guía de C#: lenguaje administrado de .NET — Microsoft Learn. Accedit 20 Abril, 2024. https://learn.microsoft.com/es-es/dotnet/csharp/.

[6] Google. “Google Cloud Console - Web UI Admin.” Google. Accedit 20 Abril, 2024. https://cloud.google.com/cloud-console.

[7] “Create Your Azure Free Account Today: Microsoft Azure.” Create Your Azurcome Free Account Today — Microsoft Azure. Accedit 20 Abril, 2024. https://azure.microsoft.com/en-us/free.

[8] OMRON. “Variadores de Velocidad de La Serie M1 de Omron.” Variadores de velocidad de la serie M1 de OMRON — OMRON, España. Accedit 20 Abril, 2024. https://industrial.omron.es/es/products/m1.

[9] OMRON. “SYSMAC Studio.” OMRON, España. Accedit 20 Abril, 2024. https://industrial.omron.es/es/products/sysmac-studio.

[10] B.V., Omron Europe. “Omron Drives - Official App in the Microsoft Store.” Microsoft Apps. Accedit 20 Abril, 2024. https://apps.microsoft.com/detail/9mwnm6sq7h4s?hl=en-mt&gl=MT.

[11] Microsoft. ”Microsoft Graph Dev Center — APIs and app development.”Accedit 21 Maig, 2024. https://developer.microsoft.com/en-us/graph.

[12] Google. “Namespace Google.Apis.Auth.Oauth2 (1.60.0) — .Net Client Library — Google Cloud.” Google. Accedit 21 Maig, 2024. https://cloud.google.com/dotnet/docs/reference/Google.Apis/latest/Google.Apis.Auth.OAuth2.

[13] Liberty, Jesse, and Rodrigo Juarez. ”.Net maui for C# developers: Build cross-platform Mobile and Desktop Applications”. Birmingham: Packt Publishing, 2023.

[14] Nathan, Adam. WPF 4 Unleashed: Full Color. Indianapolis, Ind: SAMS, 2010.

[15] OMRON, España. Çontroladores 1s.”Accedit 16 Juny, 2024. https://industrial.omron.es/es/products/1s-servo-drive.

[16] Google. “Interface Icodereceiver.” Google API support libraries. Accedit 16 Juny, 2024. https://googleapis.dev/dotnet/Google.Apis.Auth/latest/api/Google.Apis.Auth.OAuth2.ICodeReceiver.html.

[17] Microsoft. “Xamarin.” Accedit 16 Juny, 2024. https://dotnet.microsoft.com/en-us/apps/xamarin.

[18] Microsoft. “Unit Testing C# with MSTest and .Net - .NET.” .NET — Microsoft Learn. Accedit 16 Juny, 2024. https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest.

APÈNDIX

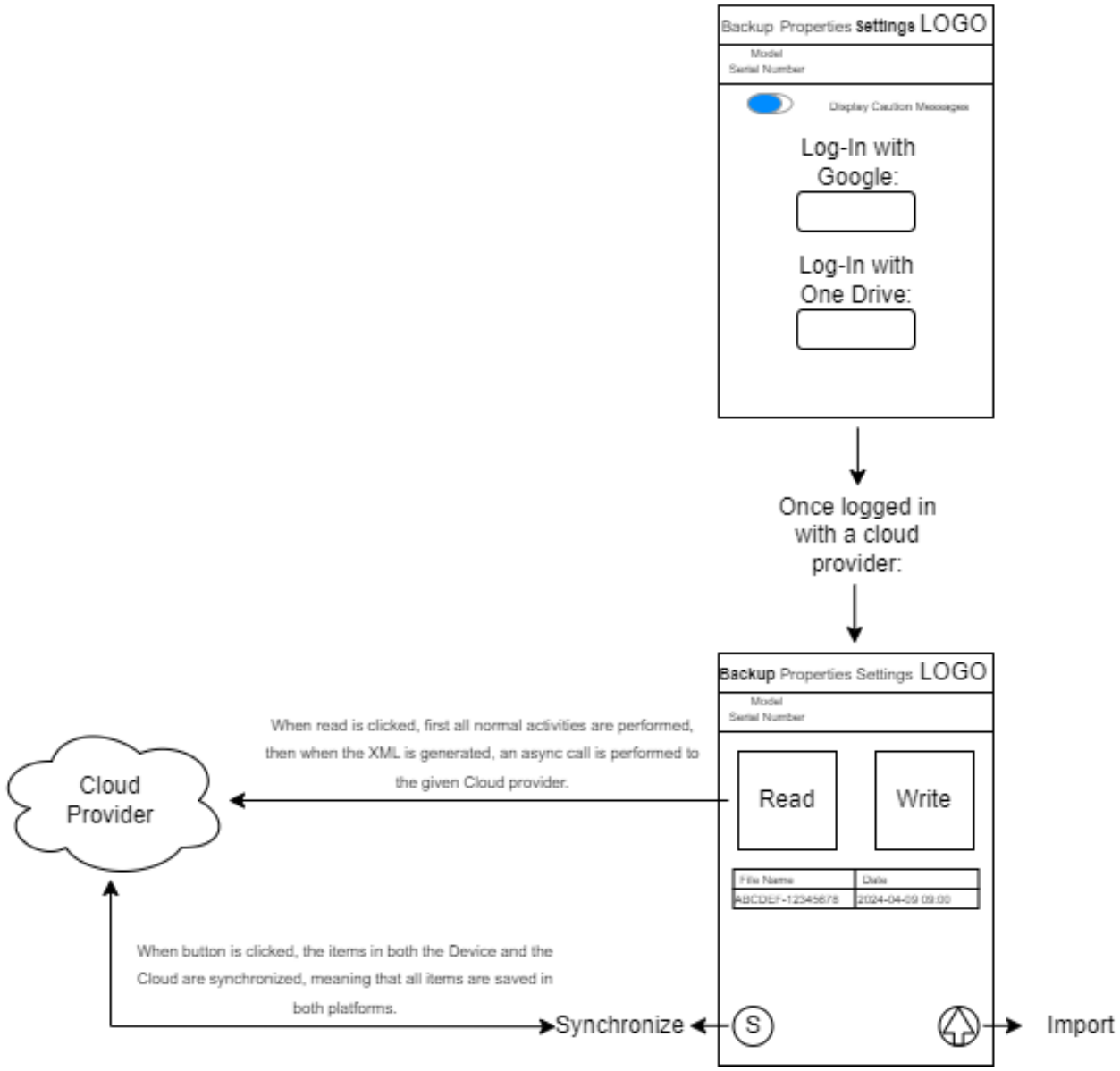

Fig. 12: PoC del Cloud Synchronization