PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito parcial para obtener el Título de INGENIERO DE SISTEMAS

DESARROLLO MULTIPLATAFORMA USANDO XAMARIN FORMS. CASO DE ESTUDIO DATALOG NEULIFT DE SISTEMA PARA GESTIÓN DE JORNADAS LABORALES.

Por

Br. ROBI ANTONIO RONDÓN PEÑA

Tutor: Rafael Rivas Estrada

Septiembre 2019

Desarrollo multiplataforma usando Xamarin Forms. Caso de estudio Datalog Neulift de sistema para Gestión de Jornadas Laborales.

Br. ROBI ANTONIO RONDÓN PEÑA

Proyecto de Grado — Departamento de Sistemas Computacionales, 86 páginas

Resumen: La empresa Datalog Srl Italia, con el fin de satisfacer las necesidades de sus clientes, ha tomado la decisión de realizar el desarrollo de una aplicación Multiplataforma orientada a dispositivos móviles con sistemas operativos Android y IOS, para esto se decidió utilizar el IDE Xamarin Forms de Microsoft que viene integrado con el entorno de desarrollo Visual Studio, con el lenguaje de programación C#. Este framework simplifica la creación de aplicaciones móviles, al ser capaz de exportar la aplicación a los sistemas operativos Android y IOS. Neulift es una aplicación existente que posee comunicación con uno de los sistemas principales de Datalog Srl Italia llamado KING, esta aplicación deberá ser desarrollada y actualizada a nuevos requerimientos entre los cuales destacan: Inicio de sesión único para cada cliente, registro las horas de trabajo diarias, un calendario de trabajo, confirmación de reportes de trabajo, consulta de reporte de trabajo, registro de mantenimiento por zona geográfica, entre otras características. Adicionalmente deberá almacenar los datos en el dispositivo móvil de forma local, a través del uso de archivos JSON y posteriormente sincronizar los datos con la base de datos principal de KING, deberá prever la configuración de los parámetros de conexión, este desarrollo se realizará bajo la metodología ágil de desarrollo SCRUM.

Palabras clave: Multiplataforma, Android, IOS, Xamarin Forms, Visual Studio, Microsoft, C#, JSON, SCRUM.

Índice

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

El continuo avance en la tecnología, en el área de los dispositivos móviles, ha llevado a que este sector prospere y sea popular entre los usuarios, con la aparición de nuevos Smartphone o teléfono inteligente y tablets, con características como cámara de video, un gran almacenamiento, buena capacidad de procesamiento y un sistema operativo robusto y sencillo de usar, lo cual facilita el día a día de las personas, ya sea para comunicarse, tomar notas, grabar videos, entre otros, esto ha sido causa de la implementación de su uso dentro de las empresas, cada día es más frecuente encontrar aplicaciones destinadas a los dispositivos móviles y orientadas a facilitar el trabajo de sus usuarios mediante la automatización de procesos o registro de datos en tiempo real. La mayoría de los dispositivos móviles posee un sistema operativo Android de Google y una minoría IOS de Apple, esto implica que el desarrollo de aplicaciones para estos dos Sistemas Operativos (SO) es independiente, dificultando su producción por el tiempo que toma desarrollar la misma aplicación para ambos SO.

Por todo lo anteriormente descrito, muchas empresas de desarrollo se han visto en la necesidad de crear Frameworks para facilitar el desarrollo de aplicaciones móviles; una de estas herramientas es Xamarin de Microsoft, la cual permite el desarrollo de aplicaciones para Android, IOS o Windows, que realiza la comunicación de código compartido, en otras palabras es un Framework Multiplataforma, teniendo en cuenta que se desarrolla sólo una vez y luego se exporta para cualquier SO (Android, IOS Windows).

Con el uso del framework de Xamarin Forms se propone crear una aplicación que gestione las jornadas laborales, aplicación destinada para el cliente Neulift de la empresa Datalog Srl, la empresa Neulift requiere una aplicación que sea capaz de realizar: el registro de las horas de trabajo, el calendario de trabajo, el registro de manteniendo que realizan sus técnicos, entre otras características adicionales. La elección de este Framework se basó principalmente por dos razones, la primera es que el lenguaje usado es C# y la segunda motivada por la facilidad del framework en la exportación de la aplicación a

otros SO de dispositivos móviles, además de que dispone de una generalización de la interfaz de usuario de la aplicación para los diversos sistemas operativos, lo que permite realizar la implementación de la interfaz una sola vez y luego es reutilizable (a excepción de casos particulares en los elementos personalizados).

1.1 Datalog Italia Srl

Datalog fue fundada en 1977 por dos analistas de aplicaciones de gestión dedicada a las empresas de contabilidad, almacén, producción, facturación, clientes/proveedores, estados financieros, entre otros. La filosofía es que el software se adapte al usuario "software a nivel humano", primer lema de la compañía, que representó desde el inicio una constante producción Datalog.

Con el transcurso del tiempo, la compañía ha desarrollado una extensa familia de productos destinados a la categoría de contadores, consultores laborales y empresas, de diferentes sectores, creando soluciones que se adaptan a las necesidades de una amplia categoría de usuario; cada desarrollo es llevado a cabo en divisiones formadas por especialistas, dedicados a diversos sectores.

1.1.1 Nuestra historia y pasión a lo largo de los años

Utilizando con el tiempo las herramientas tecnológicas más adecuadas y manteniendo el objetivo principal de la flexibilidad del software, la compañía ha estado en constante evolución, desde los sistemas operativos para mini y maxi computadoras y hasta aquellos con micro computadoras y PC; desde DOS a UNIX a WINDOWS; desde lenguajes tradicionales como: Basic CP/M, Cobol, a lenguajes para la producción de paquetes de objetos como C y Visual Basic, y archivos indexados hasta DBMS relacional.

1.1.2 Estrategia a lo largo de los años

Por primera vez en Italia, Datalog ha utilizado, desde 1980, la metodología de los paquetes de software paramétricos modernos, que se adaptan a las necesidades de una amplia categoría de empresas, sin tener que escribir programas desde cero. Según Datalog, es el programa que se adapta a la empresa y

no viceversa, como sucede en la gran mayoría de los casos. Por este Motivo a lo largo de los años, Datalog siempre ha sido un protagonista del mercado italiano y elegido como socio por empresas como:

- En 1985 Italware, una compañía del grupo Fininvest, eligió a Datalog GA85, el paquete de gestión de Datalog, para su distribución exclusiva para todo el país.
- En 1990 Symantec (USA), le confió a Datalog la distribución exclusiva en el territorio italiano de Q&A, una base de datos.
- En 1999 Datalog ha sido distribuidor para Italia de Preactor, el paquete de programación de producción más popular al nivel mundial, producido por Preactor International Ltd (Gran Bretaña) que también se ha integrado en el sistema de gestión de Datalog.

Datalog produce y distribuye software de gestión para los mercados de pequeños y medianas empresas (PYME), medianas empresas y firmas de contabilidad. Hoy Datalog es **Certified Solution Partner MICROSOFT.**

1.2 NEULIFT S.p.A

Son un equipo de especialistas que pueden diseñar soluciones personalizadas para instalar ascensores, escaleras mecánicas, salva escaleras, plataformas elevadoras en cualquier tipo de propiedad.

Se evalúa cuidadosamente el uso previsto del sistema que se nos solicita y el contexto arquitectónico que se va a operar, siempre respetando las restricciones ambientales y regulaciones de referencia. En NEULIFT ponemos toda la experiencia a disposición de los clientes, ofreciendo un servicio exclusivo y personalizado.

1.3 Antecedentes

Los sistemas de gestión de jornadas laborales en la actualidad están orientados a computadoras o a sistemas web, pero con el proceso de globalización de las nuevas tecnologías apreciamos que es mayor la cantidad de tiempo en el cual las personas pasan usando sus dispositivos móviles, por las diversas

aplicaciones de uso empresarial que se están desarrollando para ellos mismos, debido a la comodidad que puede representar su uso, al no tener la necesidad de estar frente al computador para realizar el registro de ciertas actividades. Por esta razón, migrar las aplicaciones del computador a los dispositivos móviles se ha convertido en una de las prioridades de diversas empresas. Para este tipo de desarrollos existen tecnologías nativas o no nativas en cuanto a la implementación de aplicaciones según el sistema operativo al cual esté destinado. Una de ellas es Xamarin, una plataforma de desarrollo de aplicaciones multiplataforma, es decir, que se pueden desarrollar sin importar el sistema operativo objetivo de la aplicación (Android, IOS o Universal Windows Plataform), usando el lenguaje de programación C#, creado y estandarizado por Microsoft, como parte de su entorno de desarrollo de .NET, facilitando el desarrollo de aplicaciones que posteriormente pueden compilarse y exportarse a la plataforma de destino. En la Universidad de Los Andes no se encuentran registros o documentos de proyectos realizados con la plataforma Xamarin Forms, con lo cual este proyecto pasa a ser una innovación dentro de nuestra casa de estudios.

Realizando búsqueda de otros proyectos que puedan servir como antecedentes de este trabajo en Venezuela no se consigue mayor información, sólo que hay empresas que han hecho el desarrollo de algunas aplicaciones, pero sin acceso a manuales o documentación de las mismas, esto puede deberse a que son trabajos desarrollados bajo la metodología Freelance y pueden tener derechos o acuerdo de confidencialidad en sus desarrollos, ya que la mayoría son para empresas privadas en el extranjero.

La falta de documentación acerca del desarrollo de proyectos usando el IDE Xamarin Forms puede deberse también a que se encuentra ganando campo en el área del desarrollo de aplicaciones por lo reciente y novedosa que es su tecnología, además del poco tiempo que tiene en el mercado luego de su lanzamiento por Microsoft. Incluso a nivel de búsqueda en foros de desarrolladores se puede observar que es muy reducido el grupo de programadores que tiene experiencia en su uso, pero es una realidad que debido a su versatilidad y potencial cada día crece más y más.

A nivel de la empresa Datalog Italia Srl existe un desarrollo de una versión previa de la aplicación Neulift desarrollado en la plataforma Xamarin Android, la cual sirve de apoyo fundamental para el avance de este proyecto. La diferencia fundamental entre este IDE de desarrollo previo y el destinado a utilizar en este proyecto (Xamarin Forms) es que el sistema operativo destino para la aplicación es sólo Android, lo cual no permite exportar la aplicación a otras plataformas y se complementa el desarrollo con el lenguaje Java, no compatible con la nueva tecnología a aplicar para realizar esta aplicación.

1.4 Definición del problema

Este proyecto de grado tiene como finalidad satisfacer la necesidad de la empresa Datalog Italia Srl, que presenta el problema de poseer una aplicación móvil que está destinada sólo a un sistema operativo (Android) desarrollada bajo el IDE Xamarin Android; la iniciativa nace de la necesidad de su cliente NEULIFT, de poseer la aplicación en otros dispositivos que funcionen en los sistema operativos diversos como IOS o UWP. La aplicación mencionada tienen como prioridad la traducción y el desarrollo de su aplicación Neulift, la cual se encarga de ofrecer a su cliente el control, la gestión y el registro de jornadas laborales en trabajos realizados fuera de oficina a través del uso dispositivos móviles con conexión a internet o sin conexión a internet; para este servicio la aplicación se apoya en la estructura ya existente de uno de los sistemas principales de la empresa Datalog llamado King, en los cuales se pueden gestionar los registros de reportes para una jornada de trabajo (intervenciones, horas de trabajo, permisos) de técnicos, asignaciones de tareas y de asistencias.

Para esto la aplicación debe interactuar con el sistema de escritorio (KING), el cual es un sistema de gestión fiscal desarrollado por Datalog, se requiere que la aplicación cuente con un inicio de sesión que permita cargar los datos en KING, que se encuentran en la versión de escritorio, esto debe hacerse para sincronizar los datos con la base de datos del dispositivo móvil, con el objetivo que pueda ser usada con conexión a Internet (online) o sin conexión a Internet (offline).

En este proceso de comunicación para la autentificación y la sincronización de los datos entre la aplicación Neulift y el sistema de escritorio KING, la aplicación deberá contar con un sistema de comunicaciones del tipo Application Programming Interface (API) con un servicio web de la versión de escritorio, de esta manera los datos se mantienen actualizados en ambos sistemas.

1.5 Justificación

Satisfacer las necesidades de Datalog Italia Srl al crear una aplicación Multiplataforma Neulift usando el Framework Xamarin Forms de Microsoft, con el entorno de desarrollo Visual Studio, por la solicitud de la empresa, esto con la finalidad de simplificar uno de los módulos del Software KING. Para el registro de jornadas laborales de los clientes, la empresa Datalog Italia Srl, tomando esta nueva tecnología como es el Framework de Xamarin Forms, facilita el desarrollo y migración de la aplicación previa existente para agregarle nuevas características, esto también implica su simplicidad de exportarla

a cualquiera de los sistemas operativos con los que framework tiene compatibilidad como son Android, IOS y Windows, sólo realizando pequeños ajustes para cada SO, satisfaciendo la demanda de los clientes de Datalog Italia Srl.

1.6 Objetivo General

Desarrollar una aplicación usando el IDE Xamarin Forms, para la gestión, registro, confirmación de jornadas de trabajo, para los SO Android y IOS, con el uso de los controles actuales del Software King de Datalog Srl Italia.

1.6.1 Objetivo específicos

- Desarrollar un módulo de calendario, donde se visualicen los días que se realizó un reporte de trabajo (completado, confirmado, incompleto o no definido).
- Desarrollar un módulo de confirmación de reporte diario.
- Desarrollar un módulo de resumen de reporte, presentado diariamente, semanalmente o mensualmente.
- Desarrollar un módulo de hoja de trabajo el cual genere un reporte para un día específico.
- Desarrollar un módulo de mantenimiento mensual.
- Desarrollar un módulo de mantenimiento por zona geográfica.
- Desarrollar un módulo de alerta para consultar e insertar.
- Desarrollar un módulo donde permanezca registradas las visitas que se realizan cada 2 años.
- Desarrollar una base de datos local, la cual se encuentra sincronizada con los datos del web services de King.
- Desarrollar un módulo para la configuración de los parámetros de conexión.
- Desarrollar un inicio de sesión, el cual carga los datos específicos del usuario desde el web services.
- Desarrollar un módulo de sincronización de los datos locales, con los datos que se encuentran en el web services.

- La aplicación debe funcionar, online (en línea, con uso de Internet) u offline (fuera de línea, sin uso de conexión a Internet).

1.7 Alcance

Este proyecto de grado tiene como alcance desarrollar una aplicación usando el Framework Xamarin Forms, el cual permite exportar a diferentes sistemas operativos de dispositivos móviles compatibles como Android, IOS o Windows. Esta aplicación debe realizar la gestión de jornadas laborales para los clientes de Datalog Italia Srl, para realizar la consulta de las horas de trabajo diarias, semanal o mensual; la confirmación de los reportes de las jornadas laborales; un calendario para representar los días que se cumplieron, que no se cumplieron o no se realizaron las jornadas laborales, entre otros, permitiendo obtener las mimas características del KING, en la comodidad de un dispositivo móvil para que sea de fácil acceso.

1.8 Metodología

La aplicación serán desarrollada en Xamarin Forms, usando el entorno de trabajo de Microsoft Visual Studio, el cual permite desarrollar aplicaciones multiplataforma, sean para dispositivos móviles (IOS, Android, Windows Phone), como para escritorio (Mac iOS y Windows), o sistemas web, para el caso, la creación de la aplicación móvil se podría optar por Xamarin Forms, el cual permite un solo desarrollo de la aplicación y este es compilado para las diferentes plataformas que allá sida configurada el proyecto, con esto en mente, y la participación en el proyecto de una persona para el desarrollo; la metodología elegida es SCRUM.

El método SCRUM es un conjunto de buenas prácticas para realizar trabajo en equipo y colaborativamente, para obtener un mejor resultado del producto, este método consta de entregas parciales y una entrega final del mismo, está orientado a proyectos complejos, en estos casos los requisitos son cambiantes, donde la productividad, la flexibilidad y la innovación son lo fundamental para el desarrollo del producto.

SCRUM está conformado por ciclos o interacciones y de duración corta y tiempos definidos, estas interacciones que está conformada entre 2 hasta 4 semanas, depende de la complejidad de proyecto y la cantidad de personas que conforman el equipo. Al final de cada interacción se debe entregar el resultado completo el cual fue colocado como meta en el inicio del ciclo, el cual debe ser presentado al cliente para su revisión y sus posibles correcciones, o la adición de un nuevo requisito.

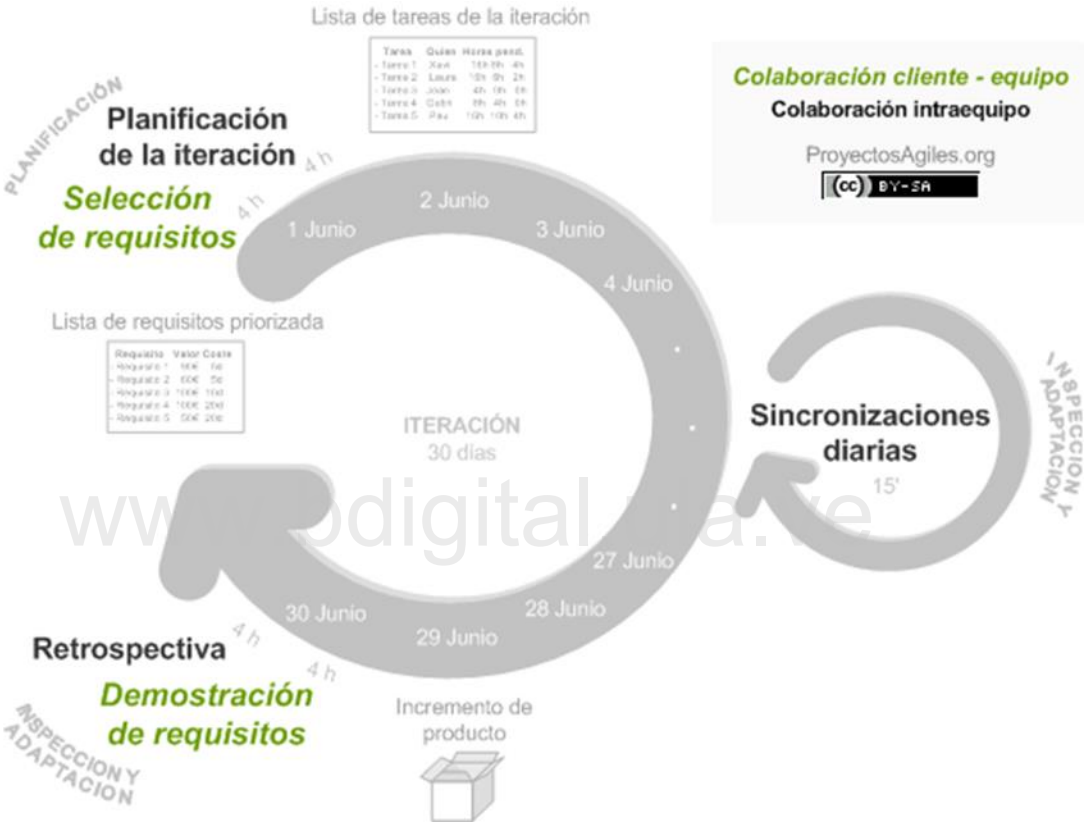


Figura 1: Diagrama de la Metodología SCRUM.

1.8.1 Roles de SCRUM.

Product Owner: es el responsable de revisar o verificar el producto resultante, esto para la revisión de cada resultado de las interacciones o el resultado final (entrega final), encargado de ofrecer las corrección y posibles nuevas adiciones al desarrollo, el cual se encarga de priorizar los objetivos, que

se separa de la historia de usuario, este también se encarga que el equipo trabaje de forma correcta desde una matiz de negocio

Scrum Master o facilitador: es el encargado de liderar el equipo de desarrollo facilitando la eliminación de obstáculos del equipo para que este alcance su objetivo de cada interacción para llegar al objetivo final que es la entrega final del producto.

Equipo de desarrollo: el equipo es el encargado de entregar al final de cada interacción el resultado del desarrollo de cada clico y realizar las correcciones (de haberlas) del mismo que se le solicitan y que son priorizadas para la próxima interacción, además, entregan el producto final, el equipo está conformado de entre 3 a 9 personas para grupos pequeños, además, se encargan de realizar el análisis, diseño, desarrollo, pruebas y documentación del proyecto.

1.9 Acuerdo de Confidencialidad

El producto final que será usado como caso de estudio en el desarrollo de este proyecto está registrado bajo un acuerdo de confidencialidad con la empresa Datalog Italia Srl, motivo por el cual podrá ser visto en el desarrollo a nivel de casos de uso y funcionalidad pero el código es 100% propiedad de la empresa.

1.10 Cronogramas

1.10.1 Cronograma de actividades.

- Actividad 1. Reunión semanal con el tutor industrial y tutor académico.
- Actividad 2. Realización un modelo de ciclos con el tutor industrial.
- Actividad 3. Definir al alcance de cada ciclo.
- Actividad 4. Investigación de las tecnologías que se van a utilizar.
- Actividad 5. Diseño de las primeras vistas de la aplicación.
- Actividad 6. Correcciones del diseño de las vistas de la aplicación.
- Actividad 7. Diseño de la base de datos.

- Actividad 8. Desarrollo de la funcionalidad de la aplicación en los diferentes dispositivos (Android y IOS).
- Actividad 9. Pruebas de la aplicación.
- Actividad 10. Redacción del informe de proyecto de grado.
- Actividad 11. Entrega del informe final con las correcciones realizadas.
- Actividad 12. Presentación del Proyecto de grado.

Actividad/ Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2	X	X	X	X	X	X	X	X	X	X	X	X	X			
3	X		X		X		X		X		X					
4		X	X													
5			X	X	X											
6					X	X										
7						X	X									
8							X	X								
9								X	X	X						
10											X	X	X			
11														X	X	
12																X

1.10.2 Cronograma de Evaluación.

- Evaluación 1: Inscripción del proyecto de grado.
- Evaluación 2: Presentación del proyecto de grado a los jurados.
- Evaluación 3: Entrega de la primera versión del proyecto de grado a los jurados.
- Evaluación 4: Entrega final del proyecto de grado.

- Evaluación 5: Defensa del proyecto de grado.

Evaluación/ Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X	X														
2							X									
3									X							
4													X			
5																X

1.11 Estructuración del Documento

La estructura del presente documento viene dada de la siguiente forma.

Capítulo 1. Introducción, capítulo conformado por los conceptos que definen el proyecto, en los cuales se encuentra: antecedentes, definición del problema, justificación objetivo general y objetivos específicos, alcance, metodología, estructura del documento, acuerdo de Confidencialidad y cronogramas.

Capítulo 2. Marco Teórico, capítulo que presentan los fundamentos teóricos que son necesarios para comprender el desarrollo de este proyecto.

Capítulo 3. Desarrollo del software, capítulo en el cual se presenta el desarrollo de la aplicación y la implementación de las diferentes herramientas que se usaron para el mismo.

Capítulo 4 .

Capítulo 5 Conclusión.

Capítulo 2

Marco Teórico

En este capítulo se presentan los fundamentos teóricos que son necesarios para comprender el desarrollo de este proyecto. Los conceptos a revisar son los siguientes: Microsoft Visual Studio, entorno de desarrollo para Windows de Microsoft; C# lenguaje de programación orientado a objetos; .NET Framework, es un entorno de compilación para diferente lenguajes de programación; MVC (Modelo Vista Controlador) es un patrón de desarrollo de software; Xamarin es un software multiplataforma para el desarrollo de aplicaciones móviles en Android, IOS o Windows; POO o programación orientada a objetos, es un paradigma de programación, donde se organiza el códigos en clases, en el cual los objetos se relacionan; metodología SCRUM, es un conjunto de buenas prácticas para realizar trabajo en equipo , con la finalidad de obtener un mejor resultado del producto.

2.1 Microsoft Visual Studio

Visual Studio es un entorno de desarrollo para Windows, Linux y MAC OS, compatible con múltiples lenguajes de programación como C++, C#, Visual Basic .NET, F#, Java, Python y PHP, como entorno para el desarrollo web como ASP.NET y para el desarrollo de aplicaciones móviles como Xamarin, este cuenta como una seria de herramientas la cual facilita a los programadores el diseño e implementación de proyectos, este también permite la inclusión de nuevos plugins a los proyectos solo con la búsqueda en el Administrador de paquetes que este posee, además, la compilación del producto para las diferentes plataformas la cual es compatible, ya sea, para versión de escritorio Windows, Linux, MAC OS, versión web, o para los dispositivos móviles, Android, IOS, UWP entre otra herramientas.

Además, de contar con 3 distribuciones para Windows como son: Community, gratuito para desarrolladores particulares, uso académico y de código abierto; Professional, no gratuito, para desarrolladores particulares, Enterprise, no gratuito, para uso de Empresas o grupos de desarrolladores

particulares. AL escribir código, se realiza de manera más rápida y precisa, utilizando IntelliSense, es una función de autocompletado que se usa en Microsoft Visual Studio y también en Visual Studio Code, realizar Debug, para encontrar y corregir errores rápidamente, permite pausar el código en ejecución, para realizar una inspección del código, utilizando un Point Break (punto de interrupción), en una línea de código específica.

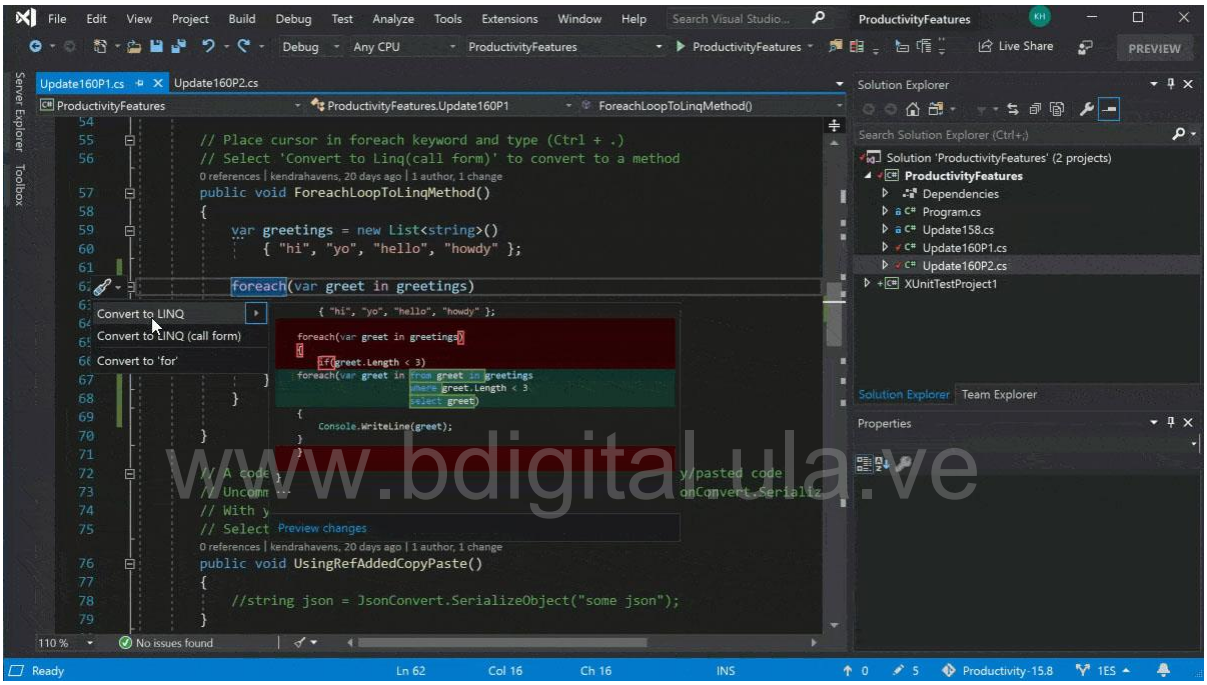


Figura 2: Entorno grafico de Visual Studio.

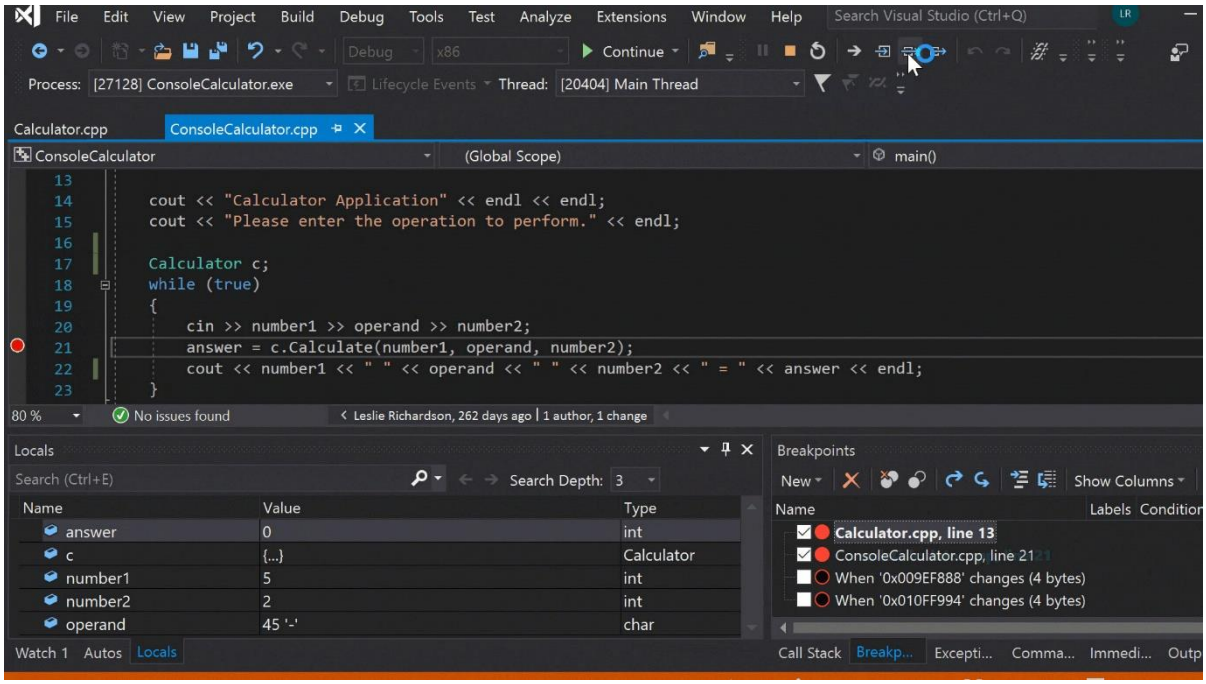


Figura 3: Entorno grafico de Visual Studio.

2.2 Lenguaje C#

C# desarrollado y estandarizado por Microsoft Corporation, es un lenguaje orientado a objetos y que se ejecuta en .NET Framework, es usado para crear aplicaciones para Windows, servicios web XML, componentes distribuidos, aplicaciones de base de datos, aplicaciones cliente-servidor entre otras. Es un lenguaje de fácil sintaxis y con una curva de aprendizaje sube, esto quiere decir, que es fácil de aprender, tiene sintaxis familiar a las de C, C++ y Java, los conceptos que son usados los lenguajes orientados objetos, también son usados en este, como polimorfismo, herencia y encapsulación, esto facilita a los programadores el crear aplicaciones que sea más robustas con un entorno más amigable. Con la existencia de un compilador que provee el marco Mono-DotGNU, el que genera programas para diferentes plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux.

2.3 .NET Framework

.NET Framework es un entorno que admite la creación y ejecución de aplicaciones Windows y de servicios Web, para los diferentes lenguajes de programación que este es compatible, dispone de un

entorno de programación orientado a objetos, con los objetivos de: minimizar la implementación de software externos y con posibles conflictos de versiones, para que la experiencia del desarrollo sea coherente en diferente tipos de aplicaciones Windows, Web o móvil.

Posee sistema de ejecución virtual llamado Common Language Runtime (CLR) y un conjunto unificados de bibliotecas de clases, brindando servicios básicos de administración de memoria, administración de subprocesos y comunicación remota, además, proporciona seguridad de tipos estricta, la colección de bibliotecas de clases orientadas a objetos, son utilizadas para el desarrollo de aplicaciones tradicionales de líneas de comandos o interfaz gráfica de usuario, hasta aplicaciones basadas en ASP.NET, Web Forms y servicios web XML.

Cuando es ejecutado un código de programación en C#, el ensamblado del mismo se carga en la CLR para realizar ciertas acciones de acuerdo a la información del manifiesto, si este cúmplelos requisitos de seguridad de la CLR para a realizar la compilación en Just-In-Time (JIT), para convertir en instrucciones nativas de la máquina.

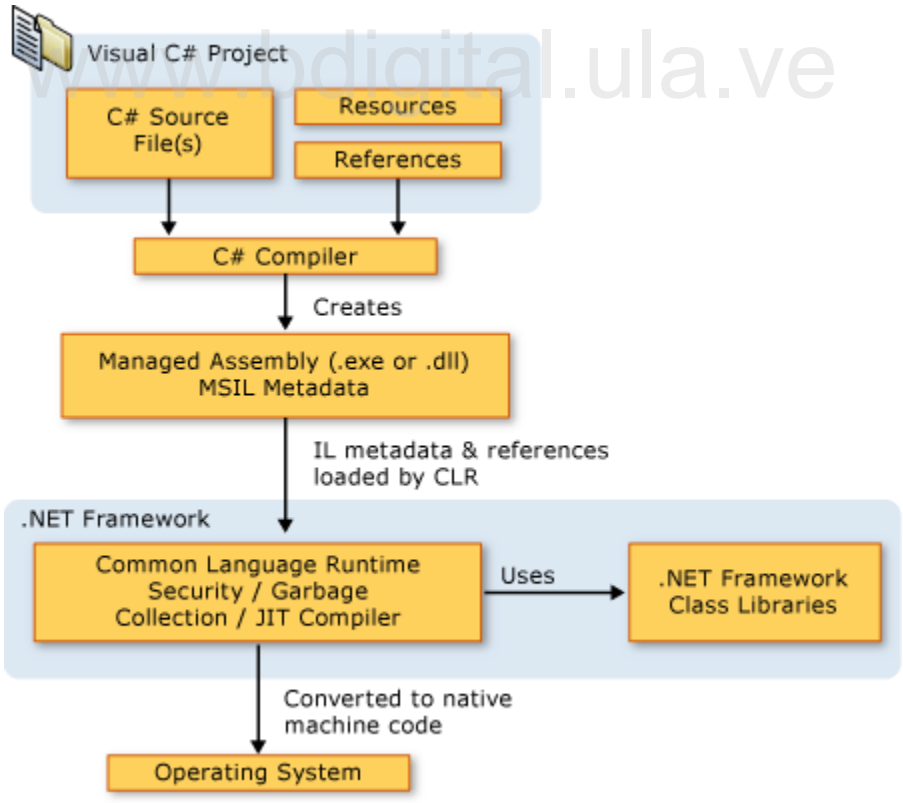


Figura 4: Estructura de un Proyecto en C# interactuando con .NRT Framework.

2.4 Xamarin

Xamarin es un software multiplataforma de código abierto y gratis, con un código compartido en C#, de esta forma se pueden crear aplicaciones nativas para dispositivos móviles como Android, iOS y Windows, permitiendo compartir código con la aplicación .NET, facilitando la creación de aplicaciones mucho más rápido y para todas las plataformas.

Las aplicaciones compiladas en Xamarin tienen un acceso a toda la funcionalidad que provee los dispositivos, además aprovecha la aceleración de hardware específica de la plataforma.

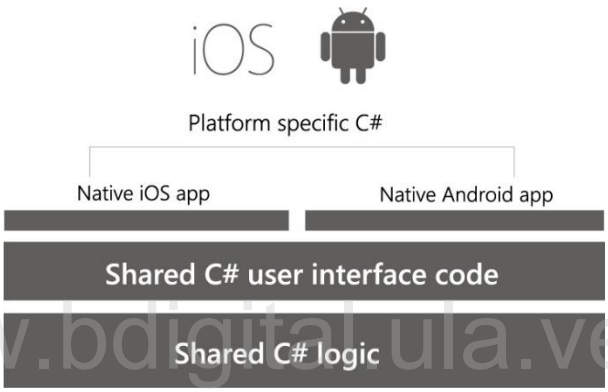


Figura 5: Xamarin

2.4.1 Xamarin Android

Tomando ventaja del SDK de Android se puede crear aplicaciones para cualquier dispositivo usando C# como lenguaje de programación, además, pudiéndose usar las bibliotecas de Android y Google Service nativas del mismo, este también tiene soporte con Android XML, para crear todos los ítem gráficos propios del SDK, como los LinearLayout, TextView entre otros, facilitando el diseño de las aplicaciones, el código de programación se usa C# tiene similitudes con Java, además, este permite la incorporación de código Java, haciendo más versátil el desarrollo, esto sucede cuando se necesita realizar una acción que solo es comprendida por el dispositivo. Este genera el producto para cualquier dispositivo Android como: teléfono, Tablet, relojes y televisores, sin ningún tipo de restricción solo realizando las configuraciones necesarias para su compilación.

Además, las aplicaciones Xamarin.Android se compilan desde C# en Intermediate Language (IL), luego es compilando en Just-in-Time (JIT) que se compila en ensamblador nativo, las cuales se ejecutan en un entorno Mono, junto con una máquina virtual Android Runtime (ART), el enlace que se crea entre .NET y Android.* y Java.* se denomina Managed Callable Wrappers (MCW) y Android Callable Wrappers (ACW) con ART, permite que ambos entornos se comuniquen entre sí, como se refleja en la siguiente imagen.

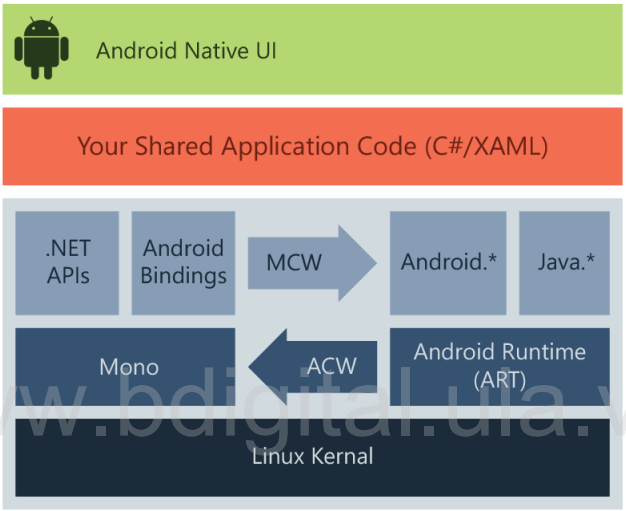


Figura 6: Estructura de Xamarin Android.

2.4.2 Xamarin iOS

Permite desarrollar aplicaciones nativas de iOS utilizando los mismos controles de interfaz de usuario que están presente en Objective-C y Xcode, con la flexibilidad del lenguaje de programación C#, usando .NET Base Class Library (BLC) y los IDEs de Visual Studio para Windows y Visual Studio para Mac, esto realizando pocas configuraciones en el proyecto en el cual se va a desarrollar, para el caso de Windows es necesario tener acceso a Xcode por medio la Red, para visualizar la aplicación en el emulador o en el momento de generar la aplicación. Las aplicaciones Xamarin.iOS están compiladas con Ahead-of-Time (AOT) desde C# en código ensamblador ARM nativo, y la comunicación entre C# y Objective-C se denomina Bindings (enlaces), como se observa en la siguiente imagen.

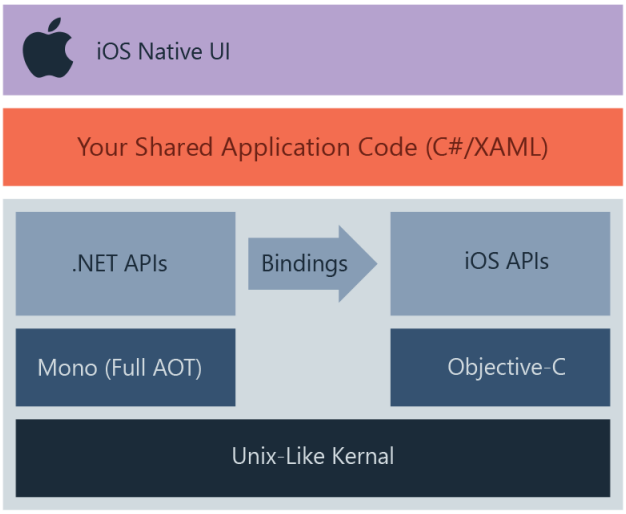


Figura 7: Estructura de Xamarin iOS.

2.4.3 Xamarin Forms

Permite a los desarrolladores crear aplicaciones de Xamarin.iOS, Xamarin.Android y Windows, desde un único conjunto de clases compartidas, permitiendo crear interfaces de usuario en XAML con código en C#. Estas interfaces de usuario se interpretan como controles nativos de la plataforma destino. Xamarin Forms posee con repositorio de paquetes Nuget, en el cual se encuentras miles de bibliotecas que son usadas para crear una entorno similar a los controles de las plataformas nativas, facilitando el desarrollo de aplicaciones, esto con pequeñas configuraciones, además, de ser capaces de personalizar las clases de que definen los propios controles de las diferentes plataformas, algunas características propias de son:

- Lenguaje de interfaz-usuario XAML.
- Enlace de datos.
- Gestos (Gestures).
- Efectos (Effects).
- Estilo (Styling).

2.5 Modelo Vista Controlador (MVC)

El Modelo Vista Controlador, es una arquitectura de software que divide los datos de los programas en tres componentes distintos: modelo, vista y controlador. Este modelo ha sido aceptado a los largos de los años para el desarrollo de aplicaciones, en una gran variedad de lenguajes y frameworks de desarrollo.

- **Modelo**: gestiona todo el acceso a la información, consultas y actualización, también implementa los privilegios de acceso de la aplicación (lógica de negocio), junto con la funcionalidad del sistema.
- **Vista**: interfaz de usuario, que se encarga de la intención con el usuario, este se encarga de visualizar los datos enviados desde el modelo.
- **Controlador**: responde a los eventos e invoca peticiones a los modelos, también es encargado de enviar órdenes a la vista, asociada a la solicitud del modelo, gestionando el flujo de información entre ellos, por lo tanto el controlador actúa como un intermediario entre el modelo y la vista.

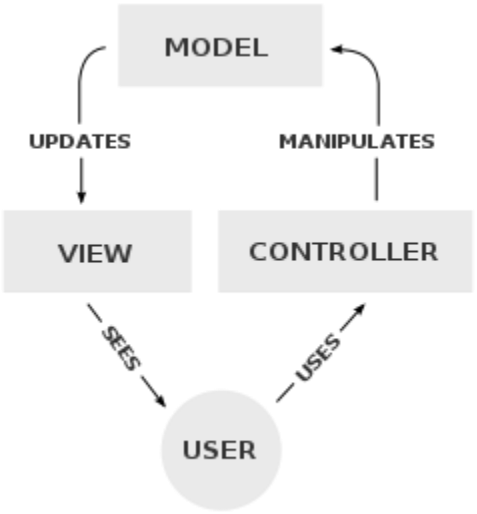


Figura 8: Modelo Vista Controlador (MVC).

La interacción con los componentes del MVC, sigue un flujo general, siendo los siguientes:

- El usuario interactúa con la interfaz de usuario de alguna forma (botones, enlaces, imágenes, etc.)
- El controlador recibe las notificaciones de la vista (interfaz de usuario), a través de algún evento.
- El controlador accede al modelo, para realizar la acción asociada al usuario.
- El controlador delega a los objetos de la vista (interfaz de usuario), los datos que fueron suministrados por los modelos, para que este visualice el contenido de los objetos.
- La vista (interfaz de usuario) espera nuevas interacciones con el usuario, para reanudar de nuevo el clico.

2.5.1 Modelo Vista Vista-Modelo en Xamarin Forms

Con el fin de permitir el desarrollo de aplicaciones de escritorio mucho más completas, complejas y un aspecto visual muchos más estilizado, Microsoft publica un artículo en MSDN, mostrando el patrón MVVM, este patrón tiene una variación con respecto al MVC, ajustadas a las necesidades de Microsoft.

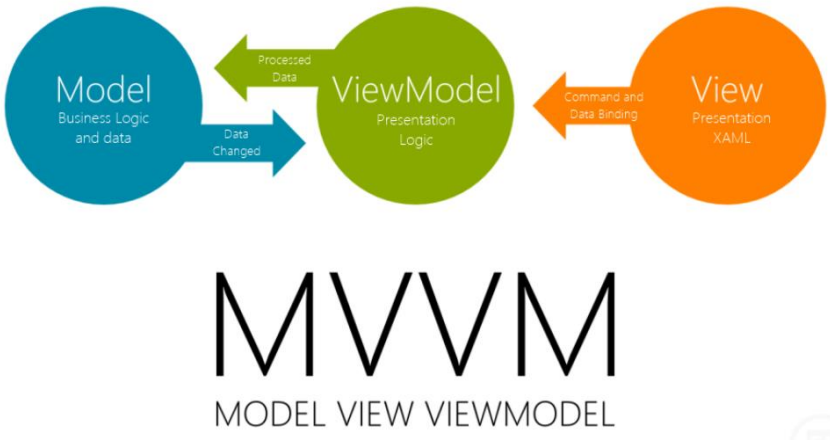


Figura 9: Mocelo Vista Vista-Modelo (MVVM).

Con el fin de desacoplar lo más que se pueda la interfaz de usuario de la lógica de la aplicación, este se dividió en tres componentes: modelo, vista y vista-modelo.

- **Modelo**: representa la lógica del negocio o la capa de datos, contiene la información, pero nunca las acciones que la manipulan.
- **Vista**: visualiza la información a través de los componentes visuales, estas contienen comportamiento, evento y enlaces de los datos, para los modelos subyacentes.
- **Vista-Modelo**: actúa como intermediario entre el modelo y la vista, contiene la lógica de presentación comunicándose por medio de enlaces de datos (binders). El Vista-Modelo se comunica con la vista a través de Data Binding, Commands y notificaciones por medio de la interfaz INotifyPropertyChanged.

```
namespace mycars.ViewModel
{
    public class CarsViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        protected void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }

        private List<Cars> _cars;
        public List<Cars> Cars
        {
            get { return _cars; }
            set
            {
                _cars = value;
                OnPropertyChanged("Cars");
            }
        }
```


Figura 10: Definicion de un Modelo.

- **Binding**: relaciona dos propiedades entre sí, de tal manera que se mantengan sincronizadas.

2.6 Programación Orientada a Objetos (POO)

Es un paradigma de la programación, que implementa una manera de programar específica, donde el código es organizado en grupos denominados clases, la cuales se crean objetos que se relacionan entre sí, estos objetos manipulan los datos de entrada, para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especifica de la clase.

En la actualidad existe una gran variedad de lenguajes de programación orientados a objetos, entre los cuales podemos mencionar los siguientes: C++, Objetive C, Java, Ruby, Visual Basic, C#, PHP, Python entre otros. Estos lenguajes de programación permiten la agrupación de bibliotecas, además, permite crear sus propias bibliotecas.

La programación orientada a objetos introduce nuevos conceptos, como son los siguientes:

- **Clases**: es una estructura de datos, en la cual se define los atributos y métodos propios de la clase.
- **Herencia**: es la transferencia de código entra una clase y otra, para que herencia se cumpla de debe tener dos clases, una clase padre y otro clase hija, mediante la cual la case padre comparte el código con la clase hija, esto transfiere el código tal cual de una clase a la otra.
- **Objetó**: es una instancia de la clase, provista de los atributos, propiedades y métodos.
- **Método**: es una subrutina con una serie de sentencias para llevar a cabo una acción.

La programación orientada a objetos (POO), contempla las siguientes características importantes:

- **Abstracción**: cada objeto sirve como un modelo abstracto el cual puede cambiar de estado, informar y comunicarse con otros objetos, sin revelar como implementan estas características, la abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes, siendo un proceso clave para el análisis y diseño, ya que mediante de ella podemos llegar a armar un conjunto de clases permitiendo modelar la realidad o un problema, que se busca una solución.
- **Encapsulamiento**: conjunto de elementos que pertenecen a la misma entidad, es decir, que se encuentra en el mismo nivel de abstracción.
- **Polimorfismo**: conjunto de objetos, los cuales pueden contener diferentes tipos de objetos, la invocación de un comportamiento de una referencia producirá el comportamiento correcto para el tipo real del objeto referido.
- **Herencia**: las clases se encuentras relacionadas entres si, formando una jerarquía de clasificación, dichos objetos heredan las propiedades y el comportamiento de todas las clases padres a las que pertenecen.

- **Modularidad**: es subdividir el programa en pequeñas partes de programas llamadas módulos, cada una independiente tanto como sea posible, los módulos pueden ser compilados por separados, pero pueden conectarse con otros módulos.

2.7 SCRUM

Es un conjunto de buenas prácticas para trabajar colaborativamente y en grupo, para obtener el resultado de la mejor manera posible para un proyecto. En esta metodología se realizan entregas parciales del producto, priorizando la opinión del cliente, Scrum está indicado para proyectos complejos, en donde se necesita tener resultados con rapidez, donde los requisitos funcionales son cambiantes o no se encuentran definidos con claridad, además, donde la innovación, flexibilidad, productividad son fundamentales.

2.7.1 Proceso

El proceso está conformado por ciclos temporales cortos y de duración fija, internaciones o sprints que van entre 2 semanas, 3 semanas o 4 semanas; tomando como tiempo máximo para el feedback del producto real. Las listas de requerimiento llamadas Blacklog, son usadas para la asignación de sprints, la cual son tareas divididas, que luego son asignadas a los miembros del equipo para que le den solución, una vez finalizado el sprint, se obtiene el resultado para incremento en el total del producto terminado, de esta manera se ira añadiendo mejoras para el producto final.

Figura 11: Diagrama de la metodología SCRUM.

2.7.2 Planificación de sprint.

La reunión de planificación de sprint se realiza el primer día, teniendo dos partes:

- **Selección de requisitos**: lista de requisitos priorizadas del producto proporcionada por el cliente, la cual se selecciona el requisito de mayor prioridad que desean se completadas en el sprint para ser entregada al cliente cuando este le solicite.
- **Planificación del sprint**: el equipo desarrolla una lista de tareas para el sprint necesario para cumplir con los requisitos seleccionados, los miembros del equipo se asignan tareas para cumplir con los objetivos.

2.7.3 Inspección y adaptación

En el último día del scprint sea ce una revisión con el cliente:

- **Revisión**: se presenta los requisitos completados durante el sprint, según los resultados mostrados y los cambios en el proyecto, el cliente puede hacer ajustes necesario de manera objetiva desde el primer sprint para volver a planificar el proyecto.

- **Retrospectiva**: el equipo analiza la manera de trabajar y los problemas que pueden ocasionar para impedir el progreso adecuado, con el fin de mejorar continuamente la eficiencia del trabajo.

2.7.4 Roles del Scrum

La metodología Scrum posee varios roles, asignados a su proceso de desarrollos:

- **Project Owner**: se encarga que el proyecto se esté desarrollando acorde a la estrategia del negocio, escribe las historias de usuarios, las prioriza y las coloca en el Blacklog.
- M**aster Scrum o Facilitador**: elimina los obstáculos que impiden que el equipo cumpla su objetivo.
- **Development team member**: los encargados de crear el producto para que pueda estar listo con los requerimientos necesarios.

Capítulo 3

Análisis de los requerimientos

El análisis de los requerimientos ayuda a comprender mejor la solución de un problema planteado (función, datos y rendimientos), realizando un estudio y análisis del desarrollo que se debe realizar y estableciendo las restricciones establecidas que se deben cumplir, de esta manera se obtiene un panorama más claro de la implementación correcta de la solución elegida para resolver el problema planteado.

Los requerimientos de software se pueden dividir en dos grupos, funcionales y no funcionales:

- Requisito funcional: son componentes del sistema de software que tiene un conjunto de entradas, comportamientos y salidas.
- Requisito no funcional: se refieren directamente a cambios en el diseño o en la implementación.

3.1 Requisitos de Software Funcionales

Requisito	Nombre	Tipo
RS-01	Inicio de sesión.	Funcional
RS-02	Carga de datos iniciales.	Funcional
RS-03	Agregar, modificar o eliminar hoja de trabajo.	Funcional
RS-04	Confirmar Reporte.	Funcional
RS-05	Agregar, modificar y eliminar un resumen de reporte.	Funcional
RS-06	Agregar y modificar un mantenimiento.	Funcional
RS-07	Agregar y modificar un mantenimiento por zona geográfica.	Funcional
RS-08	Agregar y modificar una alerta.	Funcional

RS-09	Sincronizar los datos con los servidores de King.	Funcional
RS-10	Agregar, modificar y eliminar los parámetros de conexión.	Funcional
RS-11	Cargar los datos del técnico.	Funcional
RS-12	Cargar los datos contratación.	Funcional
RS-13	Cargar la lista de actividades recurrentes.	Funcional
RS-14	Filtrar las listas de sistemas de elevadores en función a un cliente.	Funcional
RS-15	Leer un código de barras.	Funcional
RS-16	Tomar una foto para un sistema de elevador de acuerdo al tipo de trabajo.	Funcional
RS-17	Cargas la lista de sistemas de elevadores para cada cliente de usuario.	Funcional
RS-18	Filtrar las listas de los sistemas de elevadores del sistema.	Funcional
RS-19	Listar, agregar y modificar visita bianual.	Funcional
RS-20	Calendario para visualizar los resúmenes de reportes diarios.	Funcional

Tabla 1: Requisitos Funcionales.

Nombre: Inicio de sesión.	**Tipo**: Funcional	**Requisito**: RS-01
Descripción: La aplicación debe disponer un inicio de sesión con nombre de usuario y una contraseña		
Justificación: El usuario puede iniciar sesión de acuerdo a sus datos guardados en el servidor de KING, para esto se necesita el *nombre de usuario* y la *contraseña*, solo los usuarios que trabajen para Neulift Spa están registrados y pueden acceder al sistema.		

Tabla 2: Requisito Funcional - Inicio de Sesión.

Nombre: Carga de datos iniciales.	**Tipo**: Funcional	**Requisito**: RS-02
Descripción: La aplicación debe cargar los datos iniciales correspondiente a un usuario específico.		
Justificación: El usuario deberá obtener los datos básicos para que la aplicación funcione correctamente en algunos sub-módulos, y además, para que funcione sin necesidad de conexión a internet. El sistema se encargara de realizar esta carga de datos una vez que el usuario inicia la sesión.		

Tabla 3: Requisito Funcional - Carga de datos iniciales.

Nombre: Agregar, modificar o eliminar hoja de trabajo.	**Tipo**: Funcional	**Requisito**: RS-03
Descripción: El usuario debe poder registrar los datos de su jornada laboral.		

Justificación: La aplicación debe permitir al usuario compilar una hoja de trabajo para cada día trabajado por el técnico, esto permitirá llevar un control del trabajo realizado, este resumen debe poseer: las horas trabajadas, horas de viaje y horas extras, así como otros datos relevantes: la causa, la orden, el lugar de trabajo. Sin embargo, si el reporte contiene algún error o es incorrecto, podrá ser modificado o eliminado por el usuario.

Tabla 4: Requisito Funcional - Agregar, Modificar o eliminar hoja de trabajo.

Nombre: Confirmar Reporte.	**Tipo**: Funcional	**Requisito**: RS-04
Descripción: La aplicación debe ser capaz de confirmar un reporte correspondiente a un día específico.		
Justificación: La aplicación deberá disponer de una funcionalidad que permite confirmar los reportes generados, esto después de ser registrados por los técnicos que realizaron el trabajo, lo cual permite llevar un control de los trabajos realizados y la información registrada, verificando que sea correcta, la confirmación de reportes se realizan diariamente.		

Tabla 5: Requisito Funcional - Confirmar Reporte.

Nombre: Agregar, modificar y eliminar un resumen de reporte.	**Tipo**: Funcional	**Requisito**: RS-05
Descripción: La aplicación debe poder agregar, modificar o eliminar un resumen de reporte, conformada por varias hojas de trabajo.		
Justificación: El usuario debe poder agregar un resumen de reporte para un día especifico, conformados por las hojas de trabajo del día en cuestión, permitiendo así tener las horas contabilizadas y el detalle de los trabajos realizados por los técnicos.		

Tabla 6: Requisito Funcional – Agregar, modificar y eliminar resumen de reporte.

Nombre: Agregar y modificar un mantenimiento.	**Tipo**: Funcional	**Requisito**: RS-06
Descripción: La aplicación debe poder agregar o modificar un nuevo mantenimiento referente a un cliente específico.		

Justificación: La aplicación debe disponer de una sección que permita al usuario agregar una visita de mantenimiento, este registro debe poseer los datos del técnico, los datos de contratación, detalles de mantenimiento y si el mantenimiento tiene actividades recurrentes, estas también deben ser notificadas, generando también un resumen de las horas trabajadas.

Tabla 7: Requisito Funcional – Agregar y modificar un mantenimiento.

Nombre: Agregar y modificar un mantenimiento por zona geográfica.	**Tipo**: Funcional	**Requisito**: RS-07
Descripción: La aplicación debe presentar las coordenadas geográficas donde se realiza el mantenimiento.		
Justificación: El usuario debe ser poder registrar un mantenimiento, como esta descrito en el requisito RS-06, sin embargo, debe poder tomar las coordenadas del GPS del celular, para obtener la posición donde se realiza el mantenimiento, obteniendo otros datos como la distancia con respecto a la posición actual. Esto opción debe funcionar si se tiene los permisos necesarios para la utilización del GPS, en caso de no poseer permiso, deben ser solicitados nuevamente al usuario.		

Tabla 8: Requisito Funcional - Agregar y modificar un mantenimiento por zona geográfica.

Nombre: Agregar y modificar una alerta.	**Tipo**: Funcional	**Requisito**: RS-08
Descripción: El usuario debe poder agregar una alerta asociada a un mantenimiento.		
Justificación: La aplicación debe poder agregar y modificar alertas, las cuales sirven para notificar novedades en el sistema y están conformadas por los datos: tipo de alerta, fecha, código de cliente y nota. Siendo la nota un campo obligatorio.		

Tabla 9: Requisito Funcional - Agregar y modificar una alerta.

Nombre: Sincronizar los datos con los servidores de KING.	**Tipo**: Funcional	**Requisito**: RS-09
Descripción: La aplicación debe sincronizar los datos que se encuentran almacenados en los archivos JSON con los servidores de King.		

Justificación: La aplicación prevé un registro a nivel de archivos JSON, mediante los cuales se pueden registrar los datos si el usuario no posee conexión y la aplicación se encuentra fuera de línea (Offline). Adicionalmente Neulift solicito que los datos se mantuvieran sincronizados con los del servidor de KING, automáticamente cuando el dispositivo recupere el acceso a internet. La aplicación inicia la sincronización cuando se encuentra con conexión a internet (Online).

Tabla 10: Requisito Funcional - Sincronizar los datos con el servidor de King.

Nombre: Agregar, modificar y eliminar los parámetros de conexión.	**Tipo**: Funcional	**Requisito**: RS-10
Descripción: La aplicación debe poder agregar los datos para la configuración de la conexión al servidor.		
Justificación: El usuario debe poder agregar los campos para la conexión con el servidor KING disponible al registro de los datos, estos campos son dirección IP, el usuario y la contraseña, esto motivado a la posibilidad en Datalog de establecer otro servidor u obtener nuevos clientes, permitiendo de esta manera que sea sencillo agregar nuevos datos o modificar/eliminar los datos anteriores.		

Tabla 11: Requisito Funcional - Agregar, modificar y eliminar los parámetros de conexión.

Nombre: Cargar los datos del técnico.	**Tipo**: Funcional	**Requisito**: RS-11
Descripción: La aplicación debe poder cargar los datos del técnico.		
Justificación: El usuario debe poder visualizar los datos de técnico al que tienen asociado un mantenimiento que ha realizado.		

Tabla 12: Requisito Funcional - Cargar los datos del técnico.

Nombre: Cargar los datos contratación.	**Tipo**: Funcional	**Requisito**: RS-12
Descripción: La aplicación debe cargar lo datos de la contracción asociados a un mantenimiento.		
Justificación: El usuario debe poder visualizar los datos de contratación del mantenimiento realizado, estos datos deben ser visualizados en los detalles del mantenimiento por motivos informativos internos.		

Tabla 13: Requisito Funcional - Cargar los datos de contratación.

Nombre: Cargar la lista de actividades recurrentes.	**Tipo**: Funcional	**Requisito**: RS-13
Descripción: La aplicación debe cargar las actividades recurrentes que se asocian a un mantenimiento.		
Justificación: El usuario debe ser capaz de seleccionar 0, 1 o N cantidad de actividades recurrentes, que pueden estar asociadas a un mantenimiento.		

Tabla 14: Requisito Funcional - Cargar la lista de actividades recurrentes.

Nombre: Filtrar las listas de sistemas de elevadores en función a un cliente.	**Tipo**: Funcional	**Requisito**: RS-14
Descripción: La aplicación debe poder filtrar listas de los elevadores de acuerdo a un cliente.		
Justificación: El usuario debe poder filtrar las listas presentadas en las tablas de los sistemas de elevadores según un cliente en específico, la aplicación debe recargar la lista con los nuevos resultados, además de compartir el filtro con el resto de las tablas, facilitando la búsqueda en la tabla.		

Tabla 15: Requisito Funcional - Filtrar las listas de sistemas de elevadores en función a un cliente.

Nombre: Leer un código de barras.	**Tipo**: Funcional	**Requisito**: RS-15
Descripción: La aplicación debe ser capaz de leer un código de barras y cargar un mantenimiento.		
Justificación: El usuario debe ser capaz de usar la cámara del celular para leer un código de barras e interpretarlo de acuerdo el mantenimiento que este tiene asociado, y posteriormente cargar los detalles del mantenimiento.		

Tabla 16: Requisito Funcional - Leer un código de barras.

Nombre: Tomar una foto para un sistema de elevador de acuerdo al tipo de trabajo.	**Tipo**: Funcional	**Requisito**: RS-16
Descripción: La aplicación debe poder tomar una foto del mantenimiento asociado y al tipo de trabajo que este se relaciona, y después retornar a los detalles del mantenimiento.		
Justificación: El usuario debe ser capaz de acceder a la cámara de su dispositivo desde el interno de la aplicación para tomar una foto relacionada al mantenimiento que se encuentra realizando, esto permitirá registrar los datos de la foto tomada y agregarlas al mantenimiento asociado.		

Tabla 17: Requisito Funcional - Tomar una foto para un sistema de elevador de acuerdo al tipo de trabajo.

Nombre: Cargas la lista de sistemas de elevadores para cada cliente de usuario.	**Tipo**: Funcional **Requisito**: RS-17
Descripción: La aplicación debe cargar la lista de sistemas de elevadores asociados a usuario.	
Justificación: El usuario debe ser capaz de visualizar varias tablas con los sistemas de elevadores (de acuerdo al sub-modulo en el que se encuentre), la lista es cargada en una tabla la cual divide los datos de la misma e diferentes columnas. Esta lista se debe cargar inicialmente de acuerdo a cierta configuración por defecto.	

Tabla 18: Requisito Funcional - Cargar la lista de sistemas de elevadores para cada cliente del usuario.

Nombre: Filtrar las listas de los sistemas de elevadores del sistema.	**Tipo**: Funcional **Requisito**: RS-18
Descripción: La aplicación debe ser capaz de filtrar la lista de sistemas elevadores.	
Justificación: El usuario debe ser capaz de filtrar la lista de sistemas de elevadores, de acuerdo, a lo que el usuario escriba en la barra de búsqueda, el campo a usar depende del sub-modulo en el que se encuentre el usuario, la búsqueda es realizada en diversas columnas.	

Tabla 19: Requisito Funcional - Filtrar la lista de sistemas de elevadores, de acuerdo, a ciertos campos de la lista.

Nombre: Listar, agregar y modificar visita bianual.	**Tipo**: Funcional **Requisito**: RS-19
Descripción: El sistema debe permitir el registro, lectura y modificación de las visita bianuales.	
Justificación: La aplicación debe cargar una lista con los sistemas de elevadores destinados a la visita bianual, esta lista solo contiene aquellos sistemas en los que se realizan visitas programadas cada dos años, en los detalles de cada una se debe visualizar los datos de técnico, los datos de la contratación, tomar una foto para el sistema de elevadores el cual se le realizo el mantenimiento, agregar una alerta, la fecha de la visita, el ente que la realiza, una prescripción y una nota, además cada una tiene asociada una hoja de trabajo. El usuario debe poder listar todas la visitas bianuales, poder seleccionar alguna y visualizar sus detalles como datos del técnico, contrataciones tomar foto, ingresar alerta e ingresar las horas de trabajo, horas extras y horas de viaje, después agregar los datos la visita debe cerrar y no volver a visualizarse.	

Tabla 20: Requisito Funcional - Agregar y modificar visita bianual.

Nombre: Calendario para visualizar los resúmenes de reportes diarios.	**Tipo**: Funcional	**Requisito**: RS-20
Descripción: La aplicación deberá presentar una visualización en forma de calendario.		
Justificación: El usuario debe ser capaz de visualizar un calendario donde se visualicen los resúmenes de reportes según las fechas disponibles, señalados con un código de color específico, el cual deberá al ser pulsado, llevar a la vista de visualización de los detalles del resumen, es decir, listar las hojas de trabajo para ese día.		

Tabla 21: Requisito Funcional - Calendario para Visualizar los resúmenes de reportes diarios.

3.2 Requisitos de Software No Funcionales

Requisito	Nombre	Tipo
RS-21	Menú lateral.	No Funcional
RS-22	Ordenamientos de los datos en las tablas.	No Funcional
RS-23	Validación de algunos campos.	No Funcional
RS-24	Color específico para ciertos reportes.	No Funcional

Tabla 22: Lista de Requisitos no Funcional.

Nombre: Menú lateral.	**Tipo**: No Funcional	**Requisito**: RS-21
Descripción: La aplicación después de iniciar sesión debe tener un menú lateral con el acceso a todos los sub-módulos.		
Justificación: El usuario debe ser capaz de navegar por los diferentes tipos de sub-módulos a través del menú lateral, este menú deberá poseer ciertos comportamientos según el sub-módulo donde se encuentre el usuario.		

Tabla 23: Requisito no Funcional - Menú lateral.

Nombre: Ordenamiento de los datos en las tablas.	**Tipo**: No Funcional	**Requisito**: RS-22
Descripción: La aplicación debe ser capaz de ordenar los datos de las tablas.		
Justificación: El usuario debe poder ordenar los datos de las tablas de la aplicación, por alguna columna específica, el orden se hace de mayor a menor o viceversa.		

Tabla 24: Requisito no Funcional - Ordenar las tablas de mayor a menor o viceversa.

Nombre: Validación de campos en formularios.	**Tipo**: No Funcional	**Requisito**: RS-23
Descripción: La aplicación debe ser capaz de realizar validaciones en algunos de los campos de los formularios.		
Justificación: El usuario debe ser capaz de poder ingresar valores en los formularios y el sistema deberá realizar validaciones en diversos campos para notificar al usuario si está registrando algún tipo de datos no esperado por el sistema, por ejemplo, los campos deberán ser numéricos donde se ingresan las horas, también debe tener como máximo 2 decimales, además no debe pasar de 24 horas la suma de todos los registros.		

Tabla 25: Requisito no Funcional - Validación de algunos campos.

Nombre: Color específico para ciertos reportes.	**Tipo**: No Funcional	**Requisito**: RS-24
Descripción: El sistema debe indicar el estado de los reportes a través de un código de colores.		
Justificación: El usuario debe ser capaz de observar que los reportes tiene un color específico para diferenciarlos del resto. La aplicación debe clasificar de acuerdo a un color específico para los reportes que cumplen ciertas condiciones, dichas condiciones serian: confirmados, completados, incompletos y anomalías.		

Tabla 26: Requisito no Funcional - Color específico para ciertos reportes.

3.3 Casos de Uso

Los casos de usos es una herramienta valiosa que sirve para describir las acciones de un determinado sistema desde el punto de vista de usuario, además, para obtener una caso práctico de la aplicación de los requerimientos del sistema, esto permite modelar la funcionalidad usando actores y describiendo situaciones particulares, que son las que los usuarios usaran en el sistema.

Cada caso de uso indica como interactúa el sistema con el usuario y viceversa, para esto se crea un diagrama de casos de usos el cual representa de manera gráfica la interacción de los actores con el sistema, esto modelando la relacione entre los dos.

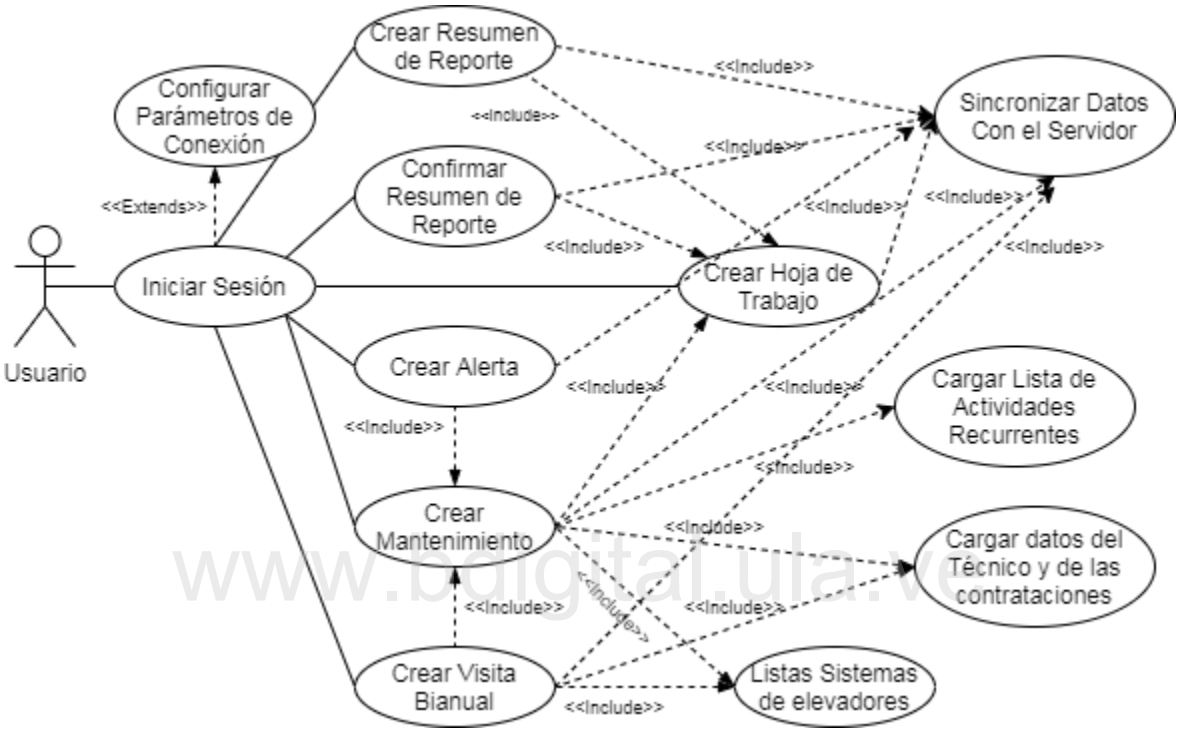


Figura 12: Diagrama de Caso de Uso.

Caso de Uso	Nombre	Prioridad
CU-01	Iniciar Sesión.	5
CU-02	Configurar Parámetros de conexión.	4
CU-03	Crear Hoja de Trabajo.	5
CU-04	Crear Resumen de Reporte.	4
CU-05	Confirmar Resumen de Reporte.	4
CU-06	Crear Mantenimiento.	5
CU-07	Cargar Actividades Recurrentes.	4
CU-08	Cargar Datos del Técnico y de las Contrataciones.	5

CU-09	Lista Sistemas de Elevadores.	5
CU-10	Crear Visita Bianual.	4
CU-11	Crear Alerta.	4
CU-12	Sincronizar datos con el Servidor.	5

Prioridad: (1) No crítica; (2) Baja; (3) Intermedia; (4) Alta; (5) Crítica.

Tabla 27: Lista de Casos de Usos.

Nombre: Iniciar Sesión.	**Caso de uso**: CU-01
Requerimientos Relacionados: RS-01, RS-02	**Prioridad:** 5
Descripción: El usuario acceder a la aplicación.	
Precondiciones: El usuario debe tener los datos (Usuario y contraseña) con anterioridad, para poder iniciar sesión, estos datos son generados por Datalog Italia Srl.	
Flujo Normal: 1. Presionar la aplicación y se coloca los datos; usuario y contraseña en sus respectivos campos.	
Flujos Alternativos: No hay flujos alternativos.	
Postcondiciones: La aplicación realiza las verificaciones para confirmar si el usuario y contraseña con válidos. Esto un una API para establecer la conexión al servidor.	

Tabla 28: Caso de Uso – Iniciar Sesión.

Nombre: Configurar Parámetros de conexión.	**Caso de uso**: CU-02
Requerimientos Relacionados: RS-09, RS-10	**Prioridad:** 4
Descripción: El usuario accede a la pantalla para ingresar una configuración diferente a la que se tiene por defecto, para que la API que usa la aplicación se conecte a un servidor diferente.	
Precondiciones: El usuario debe tener la contraseña para poder acceder a la pantalla donde se modifican los datos de la conexión.	
Flujo Normal: 1. Desde la pantalla de inicio de sesión. 2. Presionar el botón "*Parametri Connessione*" Parámetro de Conexión. 3. Colocar la contraseña y presionar el botón de "*Conferma*" Confirmar.	

4. Colocar los datos correspondientes a IP, PATH, PROTOCOL y PORT, a continuación presionar "*Salva*" Salvar y coloca la pantalla de Inicio de Sesión.

Flujos Alternativos: No hay flujos alternativos.

Postcondiciones: La aplicación guarda los datos en un archivo JSON, además, modifica los valores para la conexión a la API.

Tabla 29: Caso de Uso - Configurar parámetros de conexión.

Nombre: Crear Hoja de Trabajo. **Caso de uso**: CU-03

Requerimientos Relacionados: RS-03, RS-09, RS-19, RS-21, RS-22, RS-23 **Prioridad:** 5

Descripción: El usuario accede a pantalla de crear hojas de trabajo y agrega las horas que se ha trabajado (horas, horas extras y horas de viaje) junto con la causa, orden, lugar y una descripción (opcional), además, se tiene que verificar que los campos son válidos, no aceptar caracteres (solo para los campos de las horas), solo números con máximo dos decimales, además se puede acceder también desde el menú lateral.

Precondiciones: El usuario debe conocer las horas que se van a ingresar, junto a información adicional.

Flujo Normal:

1. Iniciar sesión.
2. Presionar el botón "*Inserimento Ore*" Ingresar horas.
3. Presionar el botón "*Foglio Manodopera*" Hoja de trabajo.
4. Se presiona sobre el botón que tiene el símbolo de "+".
5. Se tiene que visualizar la pantalla para agregar los datos, también se puede cambiar la fecha en la cual se genera la hoja de trabajo, se coloca la información.
6. Se presiona en el símbolo de guardar y después se devuelve a la pantalla de "*Foglio Manodopera*" donde se visualiza todas las hojas de trabajo, o en el símbolo "X" donde cierra la pantalla y regresa a la pantalla de "*Foglio Manodopera*".

Flujos Alternativos:

1. Iniciar Sesión.
2. Presionar el Menú lateral y luego "*Foglio Manodopera*", después seguir los pasos 4 hasta el 6 del flujo normal.
3. Se puede acceder desde Resumen de Reporte.
4. Se puede acceder también desde Mantenimiento.

Postcondiciones: La aplicación almacena la información en un archivo JSON, después de hacer todas la verificaciones de las horas, además, este módulo se puede usar sin conexión a internet para luego sincronizar todo por medio de la API hacia el servidor, en caso de contar con conexión a internet, este almacena la información en local y sincroniza los datos con el servidor.

Tabla 30: Caso de Uso - Crear hoja de trabajo.

Nombre: Crear Resumen de Reporte.	**Caso de uso**: CU-04
Requerimientos Relacionados: RS-05, RS-09, RS-19, RS-21, RS-23.	**Prioridad:** 5
Descripción: El usuario puede visualizar el resumen d reportes de tres maneras: diario, visualiza todas las hojas de trabajo correspondiente al día en cuestión, 7 días, visualiza las hojas de trabajo de los 6 días anteriores incluyendo el día en cuestión, mensual, se visualiza las hojas de trabajo de todo el mes, en caso que el mes no allá acabado solo se visualiza hasta ese día. El resumen de reporte se agrega automáticamente para cada día, es decir, que puede haber resúmenes que no posea ningún elemento, una hoja de trabajo, desde esta pantalla se puede agregar una hoja de trabajo, modificar y eliminar la misma.	
Precondiciones: El usuario debe conocer la información que desea buscar.	
Flujo Normal: 1. Iniciar Sesión. 2. Presionar el botón "*Inserimento Ore*" Agregar Horas. 3. Presionar el botón "*Riepilogo Rapporto*" Resumen de Reporte. 4. Pantalla de "*Riepilogo Rapporto*" Resumen de Reporte, por defecto esta seleccionado "*Giorno*" Día, se puede cambiar de fecha o ingresar una Hoja de trabajo en el símbolo de "*Lápiz*".	
Flujos Alternativos: 1. Iniciar Sesión. 2. Presionar el Menú Lateral y seleccionar "*Riepilogo Rapporto*" Resumen de Reporte, después sigue el flujo normal desde el 3 hasta el 4.	
Postcondiciones: La aplicación actualiza la lista si se agrega una nueva Hoja de Trabajo.	

Tabla 31: Caso de Uso - Crear resumen de reporte.

Nombre: Confirmar Resumen de Reporte.	**Caso de uso**: CU-05
Requerimientos Relacionados: RS-04, RS-09, RS-19	**Prioridad:** 4
Descripción: El usuario visualiza un menú despegable, el cual está conformado por los días en los cuales se hay Resumen de Reportes, solo tiene que seleccionarlo y confirmar el día.	
Precondiciones: El usuario debe conocer que día desea confirmar el Resumen de Reporte.	
Flujo Normal: 1. Iniciar Sesión. 2. Presionar el botón "*Inserimento Ore*" Agregar Horas. 3. Presionar el botón "*Conferma Rapporto*" Confirmar Reporte. 4. Seleccionar del menú despegable un elemento el cual es la fecha de un Resumen de Reporte. 5. Presionar el botón de guarda y ese confirma el reporte de ese día.	
Flujos Alternativos: 1. Iniciar Sesión. 2. Presionar le menú lateral y después seleccionar "*Conferma Rapporto*" Confirmar Reporte. 3. Se continúa con el flujo normal desde 4 hasta 5.	
Postcondiciones: La aplicación guarda la información en un archivo JSON y sincroniza los datos al servidor por un API, de no tener conexión se almacena en el dispositivo para posteriormente sincronizarlo.	

Tabla 32: Caso de Uso - Confirmar resumen de reporte.

Nombre: Crear Mantenimiento.	**Caso de uso**: CU-06
Requerimientos Relacionados: RS-06, RS-07, RS-08, RS-09, RS-11, RS-12, RS-13, RS-14, RS-15, RS-16, RS-17, RS-18, RS-19, RS-21	**Prioridad:** 5
Descripción: La aplicación debe cargar una lista de sistemas de elevadores de los clientes los cuales se les realiza mantenimiento, está dividido en dos mensual y geográfico; al presionar uno de los elementos de la lista se debe cargar los detalles de mismo, además se visualizan los datos del técnico y de la contratación, tomar foto del sistema que se le realizo mantenimiento y una lista con los meses donde se realiza mantenimiento y el tipo, al seleccionar uno de estos se debe culminar con el cierre del mismo, de contar con una actividad recurrente se debe visualizar y seleccionar por lo menos un elemento y proseguir con la introducción de las horas de trabajo, horas extras y horas de viaje (hoja de trabajo), además, la lista principal se puede filtrar por un cliente, ordenar por columnas o realizar una búsqueda por algún campo de las columnas.	

Precondiciones: El usuario debe conocer cual cliente que se encuentra en la lista de sistemas de elevadores desea buscar, para facilitar la búsqueda se tiene un buscador el cual se puede colocar el nombre del cliente, contar con los datos de las horas y si es necesario de las actividades recurrentes, el cliente ya tiene asociado un técnico.

Solo funciona con conexión a internet.

Flujo Normal:

1. Iniciar Sesión.
2. Presionar el botón "*Manutenzioni*" Mantenimiento.
3. Para este se puede presionar uno de los dos botones "*Mensile*" Mensual o "*Geografico*" Geográfico, depende de la necesidad del usuario.
4. Buscar el cliente al cual se le va agregar el mantenimiento y seleccionarlo.
5. En los detalles se puede revisar los datos: del técnico presionando "*Dati Tecnici*", de la contratación "*Dati Contrattuali*" o realizar una fotografía del sistema de elevadores al cual se realizó el mantenimiento en el símbolo de la cámara, además de esto se puede agregar los dato de mantenimiento seleccionando el mes que se le asigno, si este tiene actividad recurrente se debe seleccionar por lo menos una actividad, después se agrega los datos de la hoja de trabajo, juntos los datos de la hoja de trabajo asignados a la actividad recurrente y posteriormente guardar o cancelar la acción la cual devuelve al punto 4.

Flujos Alternativos:

1. Iniciar Sesión.
2. Presionar el menú lateral y seleccionar "*Mensile*" Mensual o "*Geografico*" Geográfico, y después continuar con el flujo normal desde el punto 4 hasta el 5.

Postcondiciones: La aplicación después de verificar todo, sincroniza los datos con el servidor a través de una API, solo funciona con conexión a internet.

Tabla 33: Caso de Uso - Crear mantenimiento.

Nombre: Cargar Actividades Recurrentes.	**Caso de uso**: CU-07
Requerimientos Relacionados: RS-13	**Prioridad:** 4
Descripción: El usuario puede seleccionar por lo menos una actividad recurrente relacionada al mantenimiento, esto solo si el mantenimiento lo tiene.	

Precondiciones: El usuario debe conocer que actividad recurrente va a seleccionar, esto solo para algunos mantenimientos.
Flujo Normal: 1. Desde el punto 4 del mantenimiento en flujo normal, si el mantenimiento posee actividad recurrente, se puede seleccionar por lo menos una. 2. Se presionar el botón "*Salva*" Guardar, después nos dirige a agregar hoja de trabajo y una asociada a la actividad recurrente.
Flujos Alternativos: No hay flujo alternativo.
Postcondiciones: La aplicación asocia al mantenimiento seleccionado la(s) actividad(es) recurrente(s)

Tabla 34: Caso de Uso - Cargar actividades recurrentes.

Nombre: Cargar Datos del Técnico y de las Contrataciones.	**Caso de uso**: CU-08
Requerimientos Relacionados: RS-11, RS-12	**Prioridad:** 5
Descripción: El usuario puede visualizar los datos de técnico y los datos de la contratación relacionado a un mantenimiento o a una visita bianual.	
Precondiciones: El usuario debe conocer el mantenimiento o la visita bianual que quiere ver los detalles, para poder visualizar los datos.	
Flujo Normal: 1. Iniciar Sesión. 2. Se presiona el botón "*Manutenzioni*" Mantenimiento o "*Visite Biennali*" Visita Bianual. 3. Selecciona un elemente de la lista. 4. Se presiona "*Dati Contrattuali*" Datos de Contrataciones o "*Dati Tecnici*" Datos de Técnicos.	
Flujos Alternativos: No hay flujo alternativo.	
Postcondiciones: No hay postcondiciones.	

Tabla 35: Caso de Uso - Cargar datos de técnico y de las contrataciones.

Nombre: Lista Sistemas de Elevadores.	**Caso de uso**: CU-09
Requerimientos Relacionados: RS-13	**Prioridad:** 5
Descripción: La aplicación cargar la lista de los sistemas de elevadores para después visualizarlos y el usuario luego seleccionar alguno, ha esto se le asocia el mantenimiento y la visita bianual.	
Precondiciones: No hay precondiciones.	
Flujo Normal: 1. Caso de uso CU-06 y CU-10	
Flujos Alternativos: No hay flujo alternativo.	
Postcondiciones: no hay postcondiciones.	

Tabla 36: Caso de Uso - Lista de sistemas de elevadores.

Nombre: Crear Visita Bianual.	**Caso de uso**: CU-10
Requerimientos Relacionados: RS-09, RS-11, RS-12, RS-14, RS-15, RS-16, RS-17, RS-18, RS-19	**Prioridad:** 4
Descripción: La aplicación debe cargar la lista de sistemas de elevadores de los clientes, esta lista solo contiene los sistemas que se le realizan visitas cada dos años, al seleccionar alguno de los elementos de la lista se debe visualizar los detalles, los datos del técnico, los datos de la contratación, tomar foto de sistema que se le realiza el mantenimiento, agregar alerta, después se puede proseguir con el cierre de las misma colocando la fecha de la visita, el ente, un requisito y una nota, para después seguir con la introducción de las horas de trabajo, horas extras y horas de viaje (hoja de trabajo), la lista principal debe poder ser filtrada por un cliente, para visualizar todas las visitas del mismo, además, debe contar con un buscador para facilitar la búsqueda en la lista, un ordenamiento por columnas, y un lector de código QR, el cual carga los detalles automáticamente al ser un código valido. Solo funciona con acceso a internet.	
Precondiciones: El usuario debe contar con el cliente que se desea realizar el cierre, los datos para esto sería: el ente, la fecha de visita, el requisito, una nota(opcional), las horas de trabajo, horas extras y horas de viaje (hoja de trabajo);	
Flujo Normal: 1. Iniciar Sesión. 2. Presionar el botón "*Visite Biennali*" Visita Bianual. 3. Seleccionar un elemento de la lista.	

4. Se puede visualizar los datos del técnico, contratación, tomar la foto del sistema o agregar el ente, requisito, fecha de visita y una nota, luego las horas de trabajo, horas extras y horas de viaje, después se presiona el símbolo de guarda o anular, esto cierra la vista y se vuelve a cargar la lista.

Flujos Alternativos:

1. Iniciar sesión.
2. Presionar el menú lateral y seleccionar "*Visite Biennali*" Visita bianual.
3. Se prosigue con el punto 4 del flujo normal.

Postcondiciones: La aplicación verifica los datos ingresados, después sincroniza los datos con la Api con el servidor, este módulo no funciona si no se posee acceso a internet.

Tabla 37: Caso de Uso - Crear Visita Bianual.

Nombre: Crear Alerta.	**Caso de uso**: CU-11
Requerimientos Relacionados: RS-08, RS-09, RS-11, RS-12	**Prioridad:** 4

Descripción: La aplicación debe dividir la alerta en dos tipos: "*Consultazione*" consulta y "*Inserimento*" ingresar; para el primero solo se carga una lista de alertas donde se visualiza: los detalles del cliente del sistema, los datos del técnico, datos e la contratación y tomar foto del sistema que se le realizo el mantenimiento o la visita bianual. El segundo es solo para ingresar una alerta, la cual lista todos los clientes en los cuales se realizó el mantenimiento o tiene visita bianuales, para este se toma el código del sistema la fecha y hora, además, del tipo y una nota(opcional), luego se guarda o anula la misma.

Precondiciones: El usuario debe conocer el cliente que desea agregar una alerta, además del tipo y una nota de ser necesario, el código del sistema se toma del seleccionado, la fecha y hora de toma del celular, esto dos campos no se puede modificar.
Solo funciona con acceso a internet.

Flujo Normal:

1. Iniciar Sesión.
2. Presionar el botón de "*Alert*" Alerta.
3. Depende de lo que desea realizar se presionar:
    a. El botón de "*Consultazione*" Consulta.
        i. Se selecciona algún elemento de la lista y se visualiza los detalles, los datos del técnico, datos de contratación, se puede tomar foto del sistema.
    b. El botón de "*Inserimento*" Ingresar.

i. Se seleccionar un elemento de la lista y se visualiza dos campos para agregar los datos, tipo y nota (opcional). ii. Luego se guarda o anula y se regresa a la pantalla de ingresar.
Flujos Alternativos: 1. Iniciar sesión. 2. Presionar el menú lateral y seleccionar "Consultazione" Consulta o "Inserimento" Ingresar, depende de la opción se sigue con el punto 3 del flujo normal.
Postcondiciones: La aplicación verifica los datos y los sincroniza con la API al servidor. Este módulo solo funciona con acceso a internet.

Tabla 38: Caso de Uso - Crear alerta.

Nombre: Sincronizar datos con el Servidor.	**Caso de uso**: CU-12
Requerimientos Relacionados: RS-09	**Prioridad:** 5
Descripción: La aplicación debe sincronizar todos los datos con el servidor a través de una API. Algunos módulos funcionan sin acceso a internet, por lo cual los datos son almacenados en archivos JSON en el dispositivo para su posterior sincronización cuando esta cuente con acceso al internet. La sincronización descarga los datos del servidor y los envía.	
Precondiciones: El usuario cada vez que visualiza algún modulo este debe descargar estos datos o al hacer algún cambio en los datos se deben enviar estos datos.	
Flujo Normal: 1. Descargar los datos. 2. Enviar los datos si hay algún cambio.	
Flujos Alternativos: No hay flujo alternativo.	
Postcondiciones: No hay postcondiciones.	

Tabla 39: Caso de Uso - Sincronizar datos con el servidor.

Capítulo 4

Diseño e Implementación de la aplicación.

En el presente capítulo se aborda la descripción de las fases referentes al diseño e implementación de la aplicación, el estudio de la aplicación precedente y el estudio de los análisis suministrados, todo esto tomando en cuenta la metodología de implementación que se propuso, para seguir con la organización de los diferentes sprints del desarrollo y obtener el producto final.

Por motivos de acuerdos de confidencialidad firmados con la empresa Datalog Italia Srl, la información presentada en este capítulo va a estar limitada a capturas de pantalla del código y de la aplicaciones que se considere menos relevante, la cual no influya en los derechos de autoría y propiedad intelectual del software desarrollado.

4.1 Fase de planificación.

En la fase de planificación se va a estructurar el plan de ejecución a seguir para desarrollo de la aplicación en el framework elegido para obtener un producto multiplataforma.

Antes de iniciar el desarrollo se va a estudiar el código de la aplicación precedente que está desarrollada en Xamarin Android, verificando que fragmentos del mismo pueden ser de utilidad para el nuevo desarrollo, partiendo de la premisa de reutilización de código, pero partiendo de una adaptación del mismo, debido a que por los requerimientos del nuevo software, deberán ser traducidos del lenguaje de programación Java a C#.

Después de realizar una investigación acerca del framework de desarrollo de Xamarin Forms, se obtuvieron una lista de los componentes que son compatibles y los que no lo son entre la versión anterior

y el nuevo desarrollo, por lo cual la lista de componentes no compatibles debió ser planificada la implementación desde el inicio.

Además se realizó el estudio del análisis para los nuevos cambios de funcionalidad propuestos como son el nuevo diseño de la interfaz gráfica del usuario, con diferencias y cambios visuales respecto a la versión precedente, visto que para el momento la aplicación en Xamarin Android contaba con solo 4 módulos: "*Foglio Manodopera*" (Hoja de Trabajo), "*Calendario Lavoro*" (Calendario de Trabajo), "*Conferma Rapporto*" (Confirma Reporte) y "*Riepilogo Rapporto*" (Resumen de Reporte), mientras que el nuevo diseño prevé de 3 nuevos módulos como "Manutenzioni" (Mantenimientos), "Visite Biennali" (Visitas Bianuales) y "Alert" (Alerta).

Cabe destacar que, para realizar una aplicación que pueda ser exportada a iOS, el desarrollo de los elementos de la aplicación en Xamarin Forms realizado, no deben depender de la plataforma, si se realiza algún desarrollo que utiliza las funciones nativas de un sistema operativo en particular, deberán ser especificadas las funciones en las otras plataformas a exportar la aplicación, de no ser así, se generara un error al intentar realizar la exportación hacia una plataforma, para esto existen formas específicas de escribir estas dependencia de plataformas.

En el desarrollo general, la idea es utilizar los elementos y plugins básicos proporcionados por Xamarin Forms, que son compatibles con Xamarin Android, Xamarin iOS y UWP, facilitando el desarrollo de varias funcionalidades, para algunos elementos particulares se deberá desarrollar la especificación del elemento y su funcionalidad en cada una de las plataformas.

4.1.1 Planificación de Sprint.

La buena planificación del sprint es ajustar los tiempos a 7 días, es decir, cada semana lo que facilita el progreso del proyecto, en este caso para la metodología usada, el sprint es semanal y la revisión se realiza todos los lunes, con el facilitador el cual realizaba una revisión de los puntos desarrollados, esto son los paso que se siguen para el sprint, pueden realizarse cambios en el sprint, pero sin variar la fecha de entrega:

- Entrega del análisis: esto se debe realiza en el principio de la semana, este es estudiado para comprender lo que se va a desarrollar en la semana.

- Lista de errores: una lista de posibles errores que hay que corregir, esto se toma dos días, en caso que el error no se resuelva los días para su entrega se comunica con el facilitador para postergar el nuevo documento de análisis, hasta que los puntos no sean corregidos.
- Posibles entregas: cuando se culmina el desarrollo de los puntos del análisis antes de la fecha de entrega, se comunica con el facilitador, para que este realice las pruebas necesarias.
- Entrega del desarrollo del análisis: se entrega los cambios realizados, para que el facilitador realicen las pruebas, si la aplicación pasa las pruebas, es entregado un nuevo análisis y se vuelve iniciar el sprint.

El documento que suministra por facilitador es el análisis que realiza el Project Owner, donde enumera los nuevos o modificaciones que se deben realizar, este documento se encuentra en italiano, la comunicación con el facilitador se realiza vía Skype y en italiano, el cual se le informa cuales puntos se resolvieron, cuales se tuvieron dudas, en caso de haber dudas son preguntadas y estas aclaradas para proseguir.

DATALOG

Progetti Software DATALOG ITALIA SRL

Data Documento: 09/12/2019
Reparto: King Gestionale
Autore Documento: Mariano Mingon

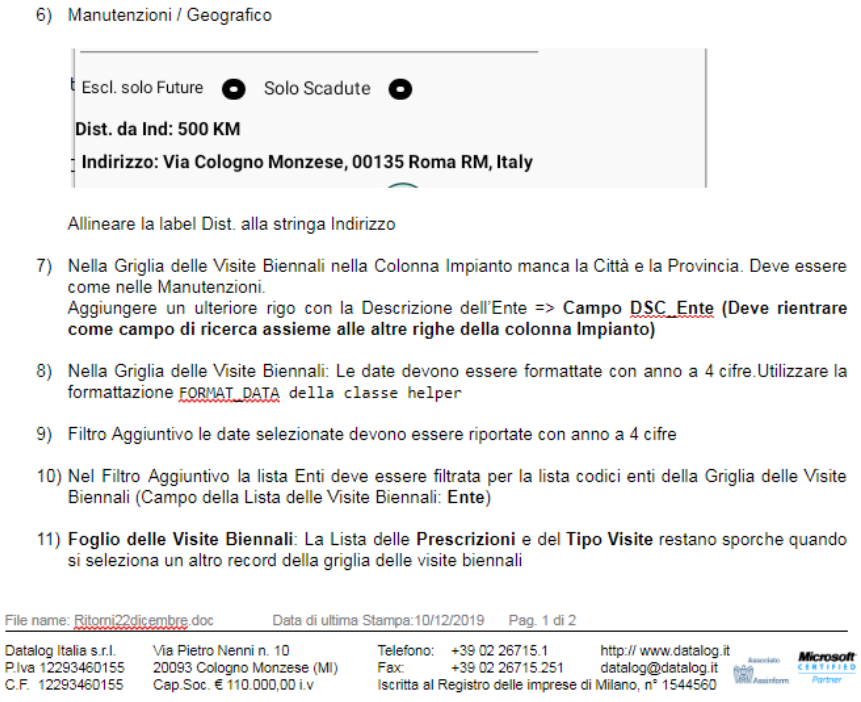
Oggetto : Ritorni APP NEULIFT

1) **Il Chk "Escludi Solo Future" (Manutenzioni / Geografico) ... Non funziona correttamente. Invece funziona correttamente quando si imposta la distanza da Tecnico**
2) Griglia Manutenzioni Mensili: Allineare al centro sia nel valore che nelle intestazioni le colonne Scadute / Mese / Future
3) Nella Griglia del dettaglio delle Manutenzioni il simbolo * (Stella) deve essere sostituito con il disegno di una rotella e il disegno deve essere **più grande** ... Altrimenti l'utente non riesce a visualizzarlo e a cliccare.
4) Alla funzione ChiudiMan passare le coorrdinate Double GPSX, GPSY della posizione del tecnico

```
/* ===================================
     LATITUDINE  => GPSX
     LONGITUDINE => GPSY
=================================== */
```

5) Alle funzioni del Service sia delle Manutenzioni che delle Visite Biennali c'è una anomalia sui decimali => 1.5 viene considerato come 15

Figura 13: Documento de análisis - Corrección de errores y nuevas modificaciones.



6) Manutenzioni / Geografico

Escl. solo Future ○ Solo Scadute ○

Dist. da Ind: 500 KM

Indirizzo: Via Cologno Monzese, 00135 Roma RM, Italy

Allineare la label Dist. alla stringa Indirizzo

7) Nella Griglia delle Visite Biennali nella Colonna Impianto manca la Città e la Provincia. Deve essere come nelle Manutenzioni.
Aggiungere un ulteriore rigo con la Descrizione dell'Ente => **Campo DSC_Ente (Deve rientrare come campo di ricerca assieme alle altre righe della colonna Impianto)**

8) Nella Griglia delle Visite Biennali: Le date devono essere formattate con anno a 4 cifre.Utilizzare la formattazione `FORMAT_DATA della classe helper`

9) Filtro Aggiuntivo le date selezionate devono essere riportate con anno a 4 cifre

10) Nel Filtro Aggiuntivo la lista Enti deve essere filtrata per la lista codici enti della Griglia delle Visite Biennali (Campo della Lista delle Visite Biennali: **Ente**)

11) **Foglio delle Visite Biennali**: La Lista delle **Prescrizioni** e del **Tipo Visite** restano sporche quando si seleziona un altro record della griglia delle visite biennali

File name: Ritorni22dicembre.doc Data di ultima Stampa:10/12/2019 Pag. 1 di 2

Datalog Italia s.r.l. Via Pietro Nenni n. 10 Telefono: +39 02 26715.1 http:// www.datalog.it
P.Iva 12293460155 20093 Cologno Monzese (MI) Fax: +39 02 26715.251 datalog@datalog.it
C.F. 12293460155 Cap.Soc. € 110.000,00 i.v Iscritta al Registro delle imprese di Milano, n° 1544560

Figura 14: Documento de análisis - Corrección de errores y nuevas modificaciones.

4.2 Fase de diseño.

Antes de iniciar la fase de diseño, su tuvo que analizar el diseño de la aplicación precedente verificando que componentes visuales no son compatibles con el nuevo, esto no lo son, por la sencilla razón que estaban desarrollado en Xamarin Android y se debe buscar una solución similar en Xamarin Forms, para esto se verifico correctamente cada uno de los módulos, encontrando varios los cuales se realizó una investigación de plugins similares que cumplan con el objetivo, en caso que no existan se desarrolla esto componentes desde cero.

4.2.1 Diseño de la interfaz de usuario.

Para el inicio del diseño de la interfaz de usuario se consideró la aplicación precedente en Xamarin Android, luego se tiene en consideración los análisis para el diseño de los nuevos módulos, en un principio

solo se iba a contar con 4 módulos: "Foglio Manodopera" hoja de trabajo, "Calendario Lavoro" Calendario de Trabajo, "Conferma Rapporto" Confirmar Reporte y "Riepilogo Rapporto" Resumen de Reporte, pero se vio la necesidad de agregar más módulos, lo cual llevo a colocar los 4 iniciales en mimos módulo llamado "Inserimento Ore" Ingresar Hora y se agregaron "Manutenzioni" Mantenimientos, "Visite Biennali" Visitas Bianuales y "Alert" Alerta, estos son los principales módulos que destacan en la aplicación.

Las características principales que solicitaron para el diseño son las siguientes:

- El diseño del inicio de sesión debe cambiar, por uno más atractivo al usuario, además, agregar un botón para configurar los parámetros de conexión. ***Ver figura 15***.
- Diseño de una pantalla para ingresar los datos para la configuración de nuevos parámetros de conexión, pero antes de acceder a esta pantalla se debe solicitar una contraseña, además, esta pantalla no se encontraba en la aplicación precedente. ***Ver figura 16***.
- EL diseño de la pantalla de inicio (Home), después del inicio de sesión, se debe visualizar los módulos principales: Inserimento Ore (Ingresar Hora), Consultazione Impianti (Consulta de Sistema, ***este módulo no se encuentra desarrollado***), Manutenzioni (Mantenimientos), Visite Biennali (Visitas Bianuales), Alert (Alerta) y Interventi (Intervenciones, ***este módulo no se encuentra desarrollado***), además el módulo Inserimento Ore (Ingresar Hora) posee sub-módulos que son: Foglio Manodopera (Hoja de Trabajo), Calendario Lavoro (Calendario de Trabajo), Conferma Rapporto (Confirmar Reporte) y Riepilogo Rapporto (Resumen de Reporte), estos módulos tienes que estar conformado por una icono que represente la acción del módulo y el nombre, estos también se debe visualizar en el menú lateral. ***Ver figura 17 y figura 18***.
- Para el módulo Inserimento Ore (Ingresar Hora) (***ver figura 17***):
  - Foglio Manodopera (Hoja de Trabajo): se debe visualizar todas las hojas de trabajo ingresadas en una lista, los elementos de esta lista tiene que estar conformado por las características de la hoja de trabajo, un filtro por fecha y un botón para agregar una nueva hora de trabajo. En la aplicación precedente la lista y los campos para agregar se encontraban en la misma pantalla incomodando al cliente, para el nuevo diseño esto se separó en dos pantallas. ***Ver figura 20***.
  - Calendario Lavoro (Calendario de Trabajo): se debe visualizar los días del mes en un calendario, además de cambiar entre meses, este visualiza los días en

diferente colores dependiendo si un Riepilogo Rapporto (resumen de reporte) está confirmado, completado, incompleto o si tiene alguna anomalía (***Ver figura 22***), también cuanta con un botón que abre Foglio Manodopera (Hoja de Trabajo). Para esto se usó un plugin gratuito para replicar esta característica en Xamarin Forms.

- Conferma Rapporto (Confirmar Reporte): se debe visualizar un menú despegable el cual debe contener las fechas en los cuales hay reporte. En este caso se tuvo que realizar el componente desde cero al no tener compatibilidad con Xamarin Forms. ***Ver Figura 19***.
- Riepilogo Rapporto (Resumen de Reporte): se debe visualizar 3 listas de las hojas de trabajo, cada de una de estas lista se visualizan: para el día actual, para 7 días, y por mes, en este se tiene un filtro por fecha el cual recarga la lista en curso, solo seleccionando una de las 3 opciones (***ver figura 21***), existiendo la posibilidad de agregar una nueva hoja de trabajo la cual abre Foglio Manodopera (Hoja de Trabajo).

- Manutenzioni (Mantenimientos): se debe visualizar una lista con todo los cliente, cada elemento de la misma debe contener varios datos del cliente, contando con varios tipos de filtro y un buscador, además, al seleccionar un elemento de la lista se debe visualizar los datos con más detalles, junto con los datos del técnico, los datos de la contratación, el icono de la cámara, una lista con todos los mantenimientos, donde se puede seguir con el cierre del mismo. Para el caso de Manutenzioni (Mantenimientos), este posee dos Mensile (Mensual, ***ver figura 24***) y Geografico (Por Zona Geográfica, ***ver figura 25***), los detalles visuales cambian muy poco.
- Visite Biennali (Visitas Bianuales): igual que en Manutenzioni, sebe visualizar una lista con los clientes que solo posean un mantenimiento bianual, es decir, cada dos años, también tiene varios tipos de filtros que son visualizados con CheckBox o listas, además, al seleccionar una elemento de la lista de clientes, se debe visualizar los detalles del mismo, junto con datos de técnico, los datos de la contratación, la cámara y otros campos para cerrar el mantenimiento. ***Ver figura 26***.
- Alert (Alerta): el módulo de alerta está divido en dos, consultar e ingresar, para ambos se deben visualizar la lista de clientes, pero el detalle de la misma, los datos del técnico,

los datos de la contratación e ingresar una alerta solo se puede visualizar en Consultazione (consultar), mientras para Inserimento solo se lista los clientes para ingresar alertas a estos. ***Ver figura 29***.

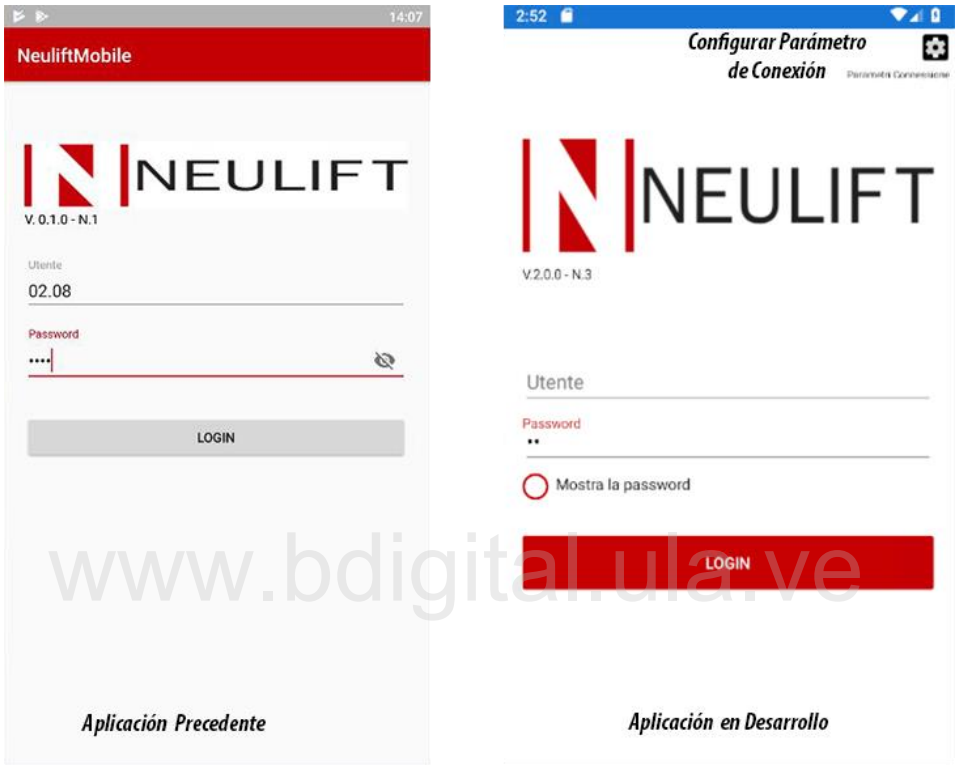


Figura 15: Inicio de Sesión.

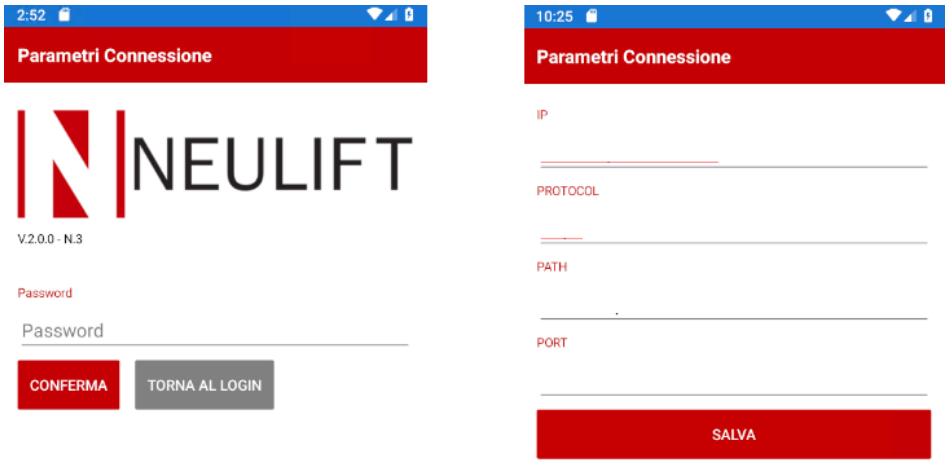


Figura 16: Configurar Parámetro de Conexión.

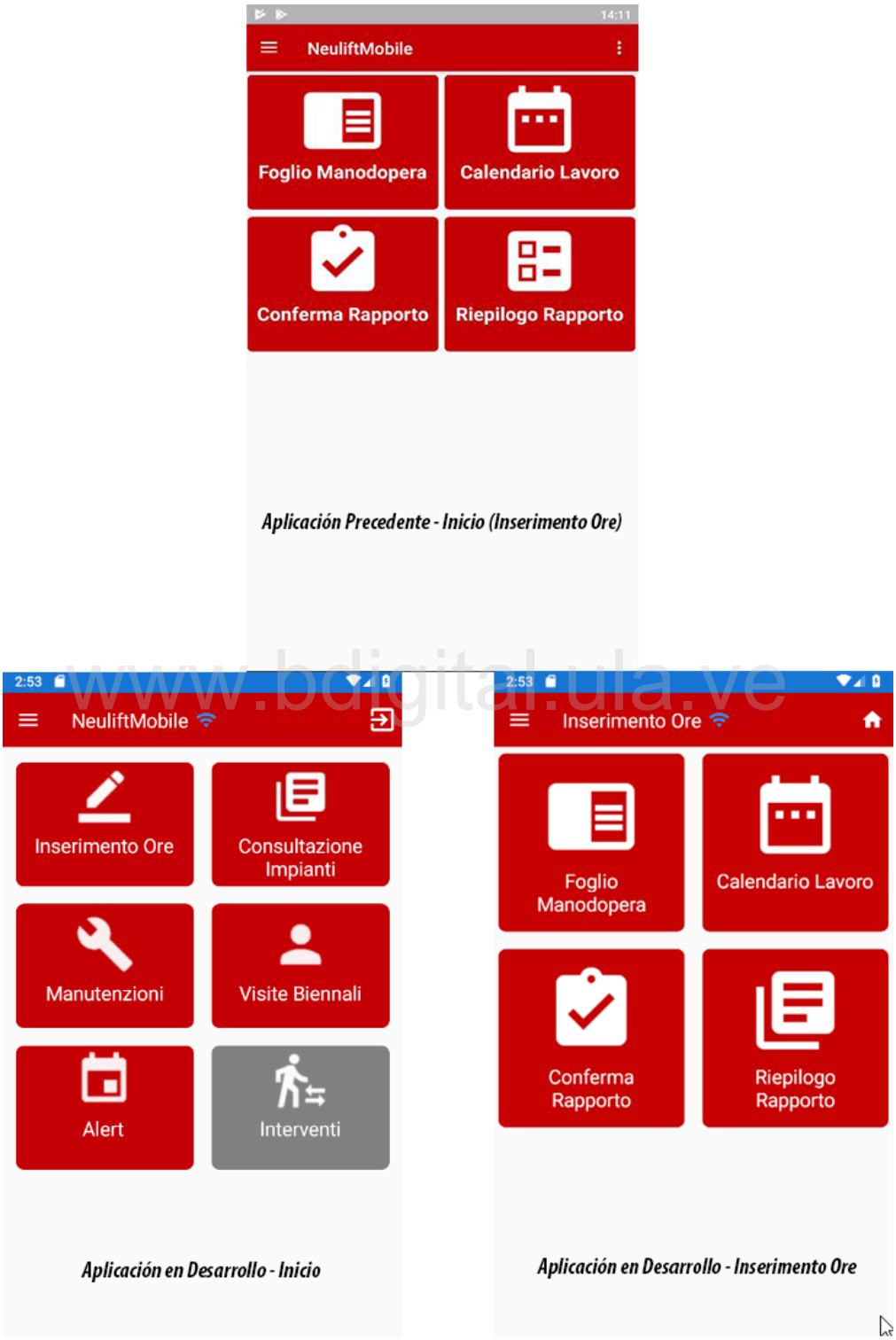


Figura 17: Pantalla Principal y cambios en la aplicación en desarrollo.

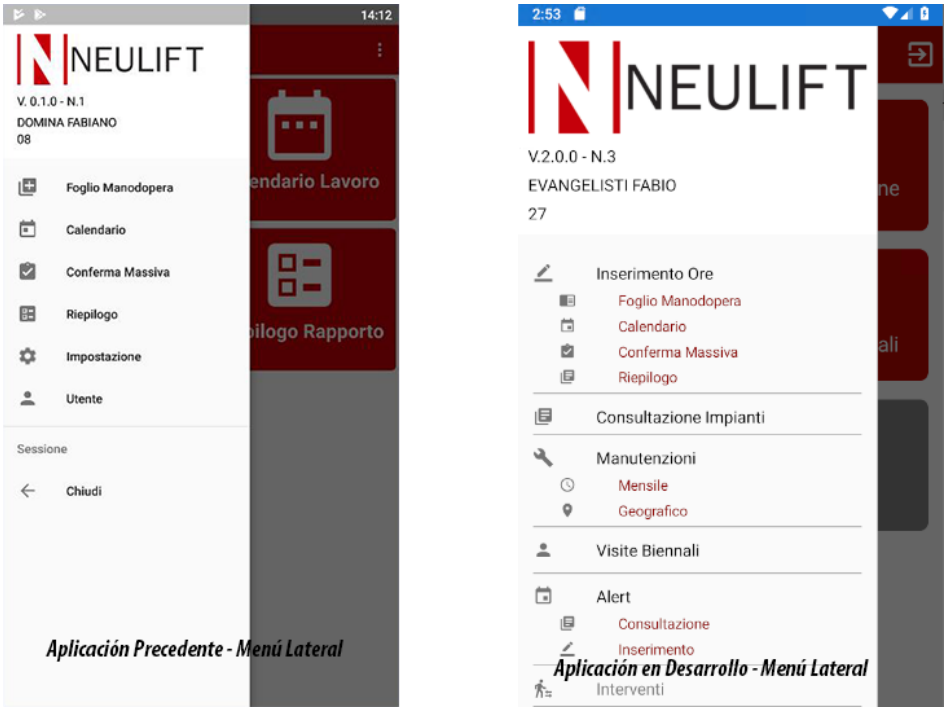


Figura 18: Menú Lateral.

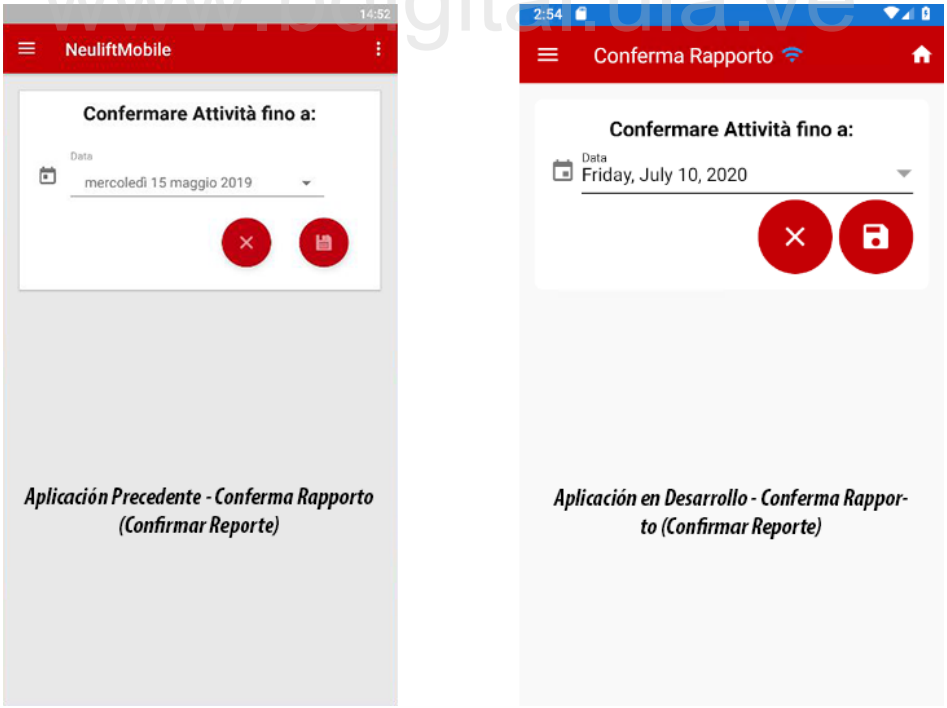


Figura 19: Conferma Rapporto (Confirmar Reporte).

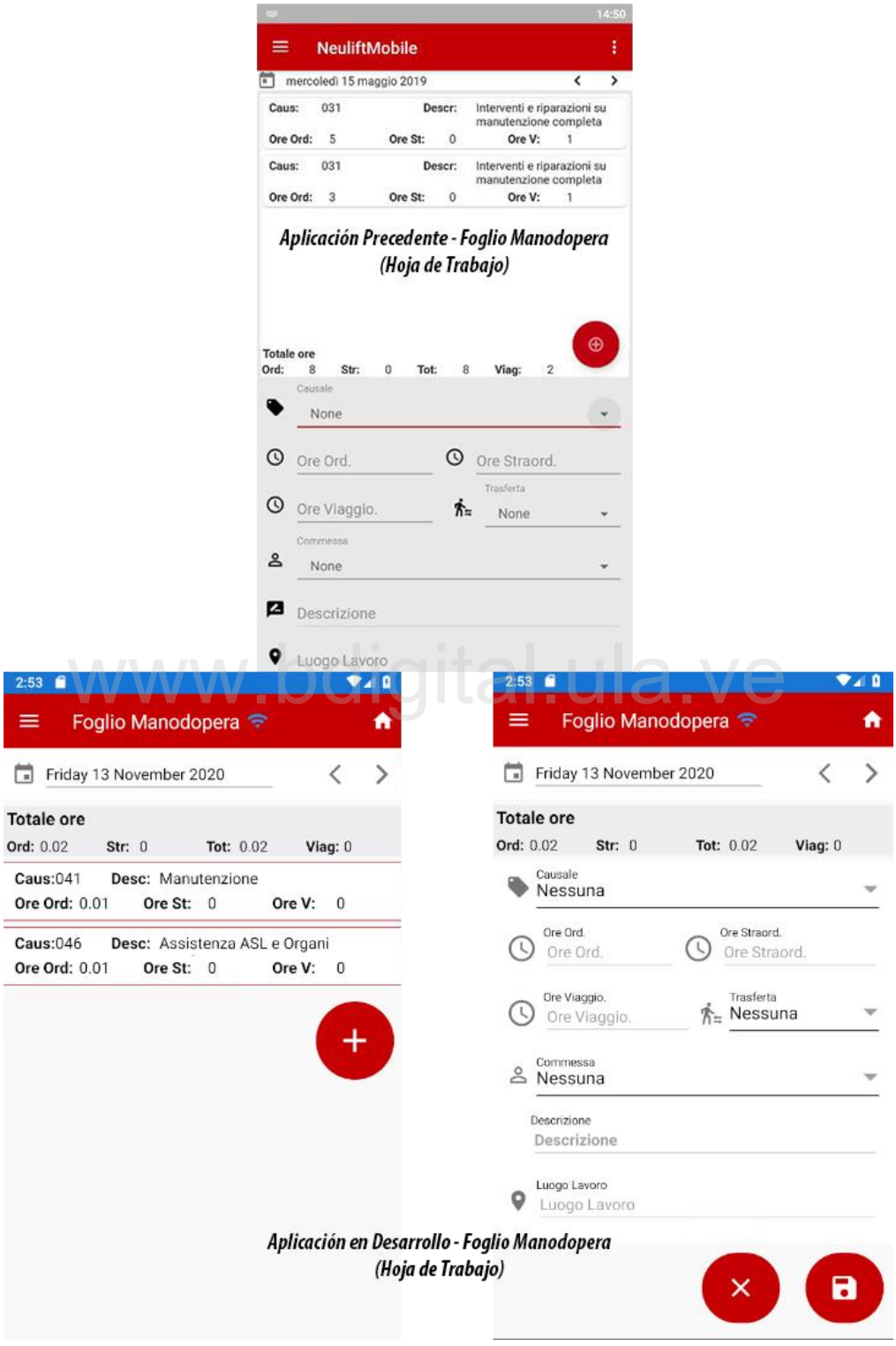


Figura 20: Foglio Manodopera (Hoja de Trabajo).

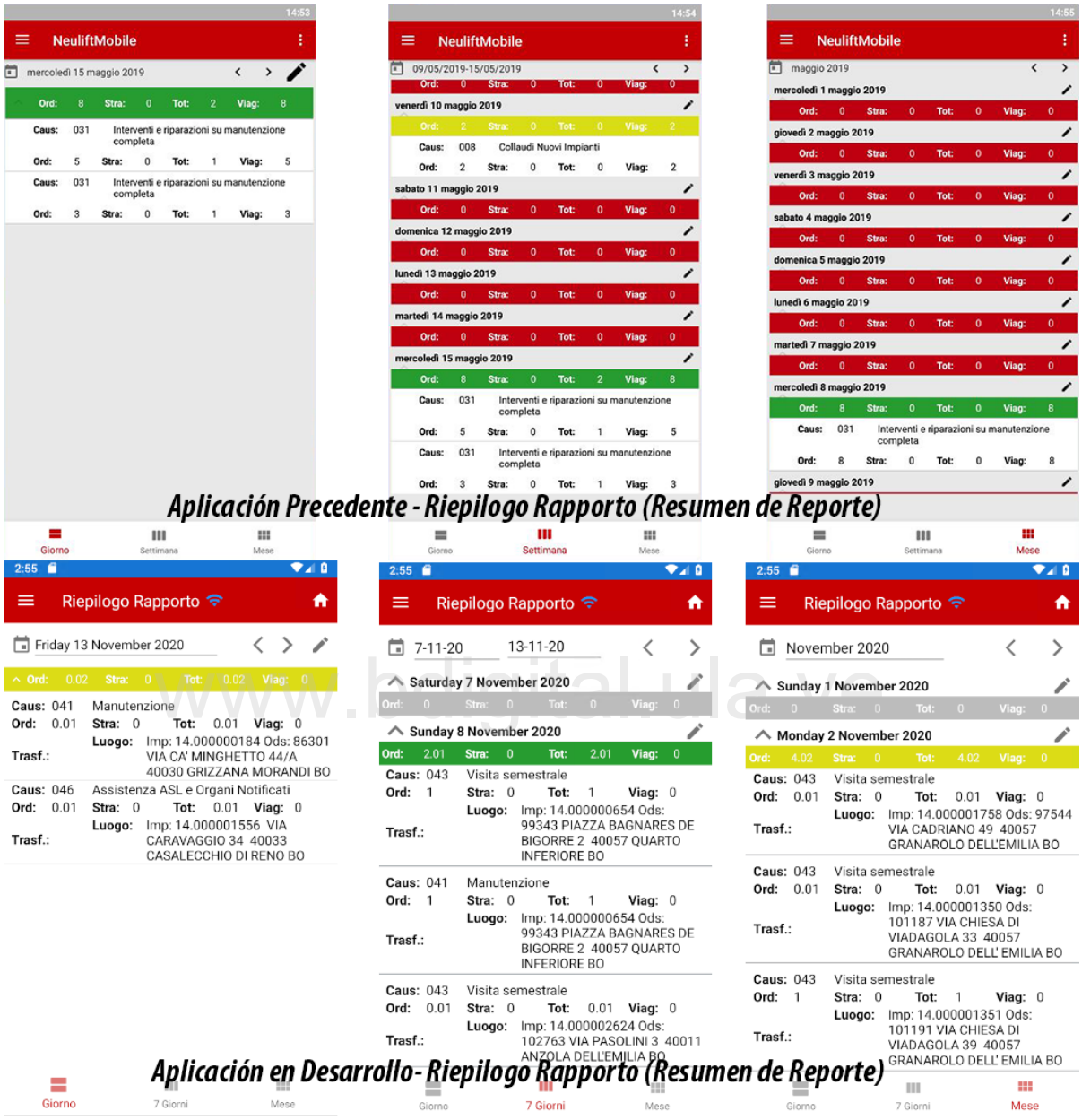


Figura 21: Riepilogo Rapporto (Resumen de Reporte).

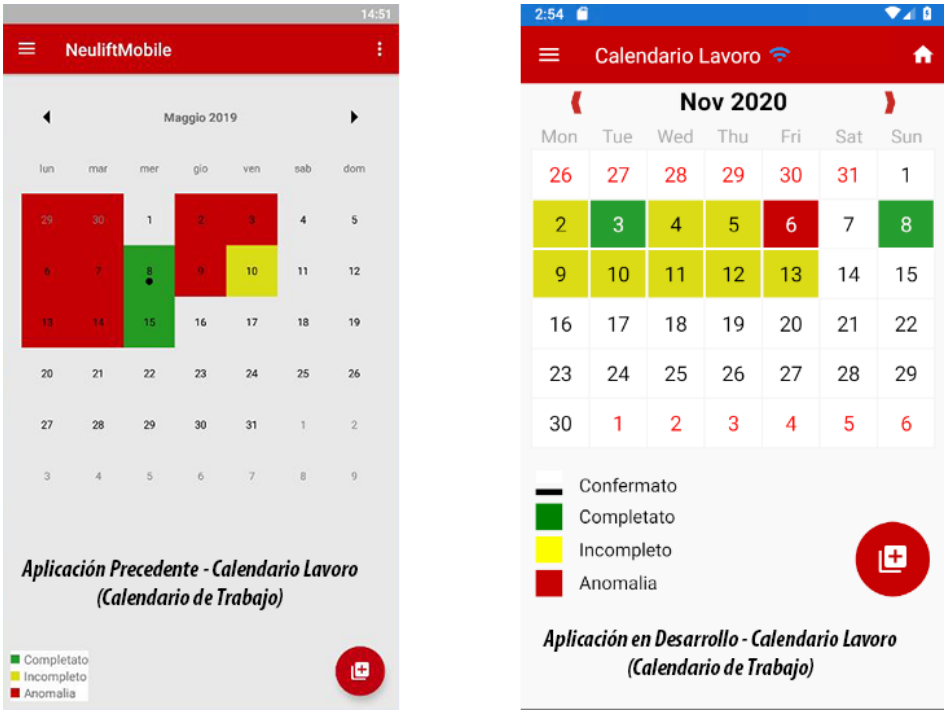


Figura 22: Calendario Lavoro (Calendario de Trabajo).

Figura 23: Menú de Manutenzioni (Mantenimientos).

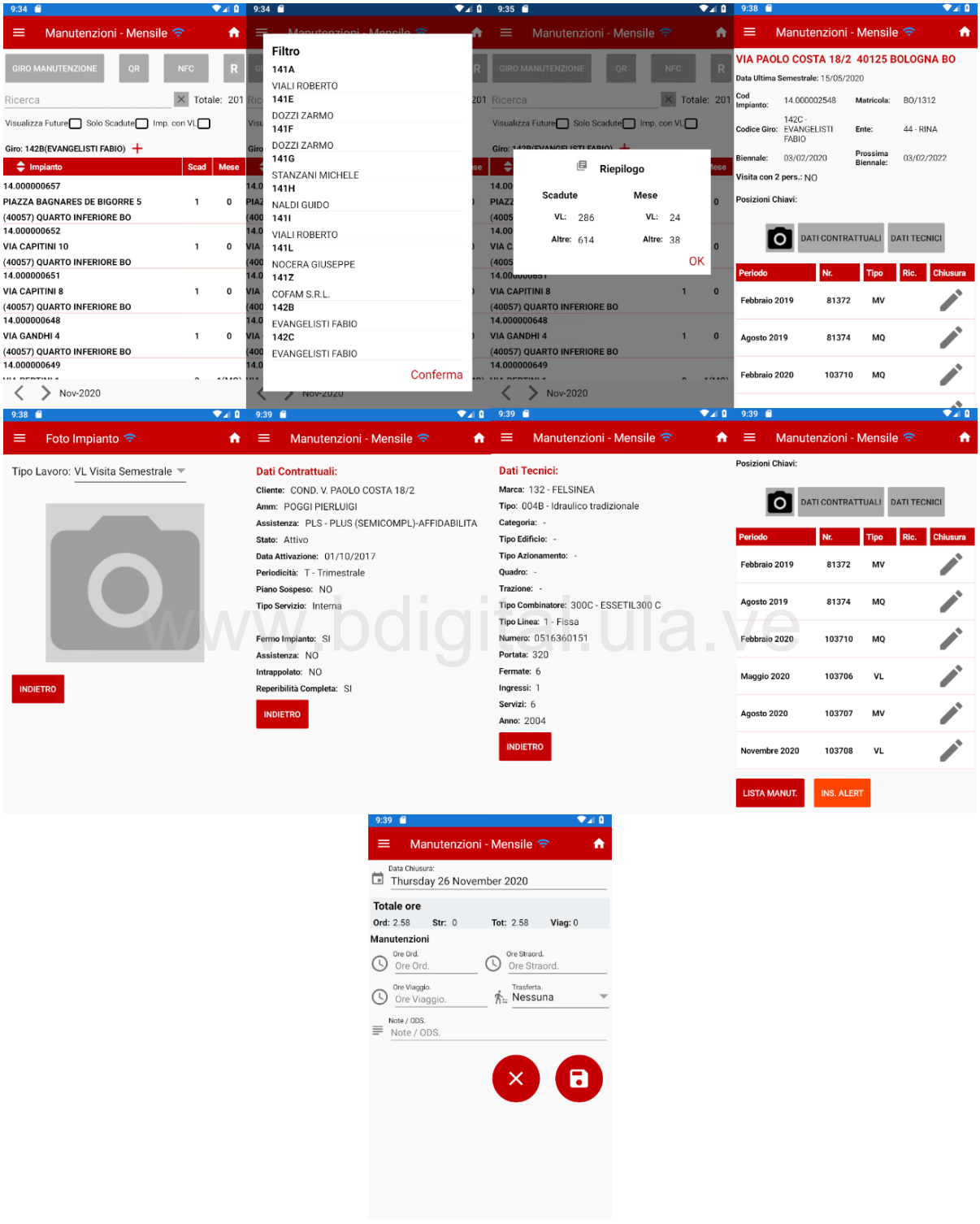


Figura 24: Manutenzioni - Mensile (Mantenimientos - Mensual).

Figura 25: Manutenzioni - Geografico (Mantenimientos -Por Zona Geográfica).

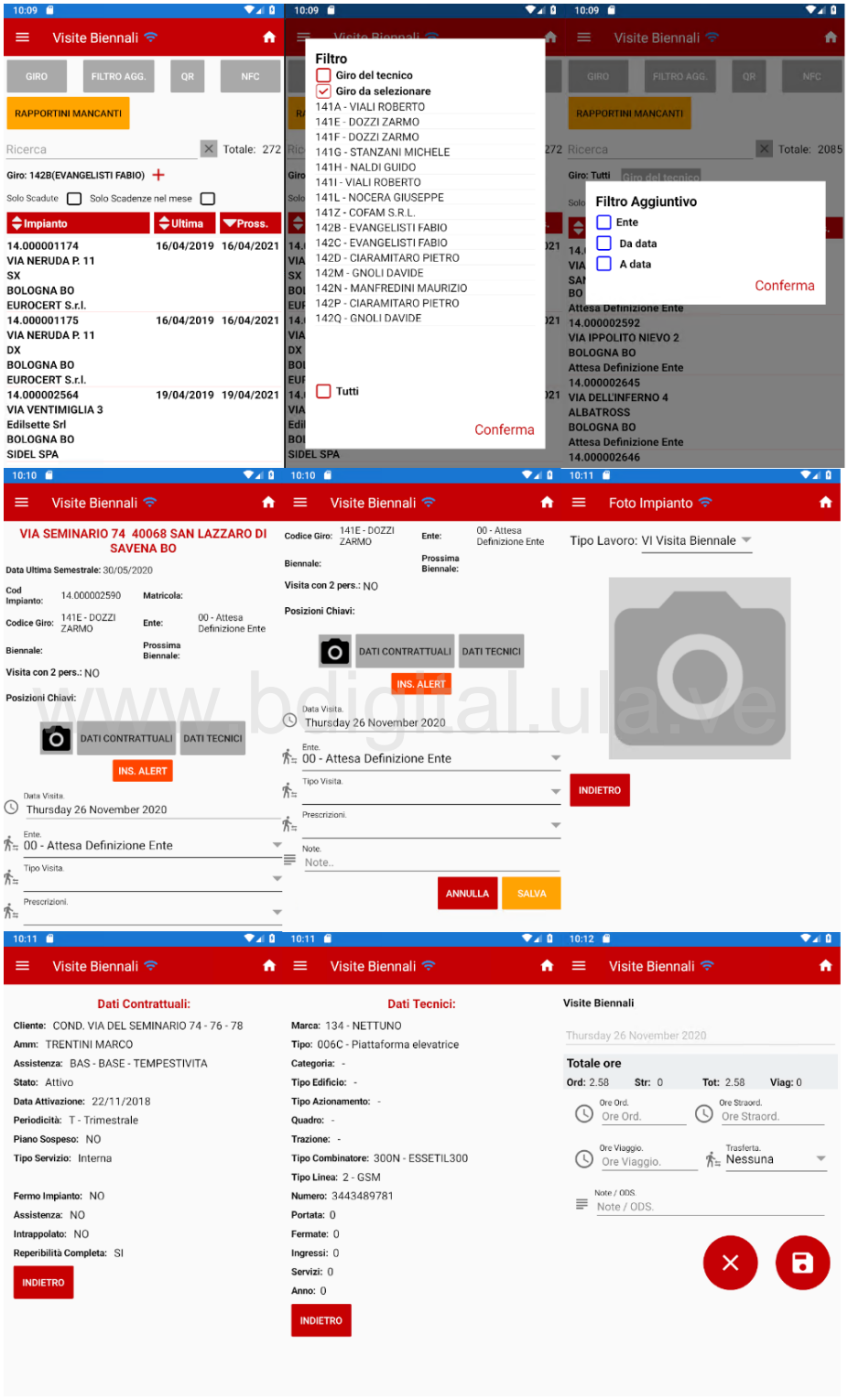


Figura 26: Visite Binnale (Visita Bianual).

Figura 27: Menú de Alert (Alerta).

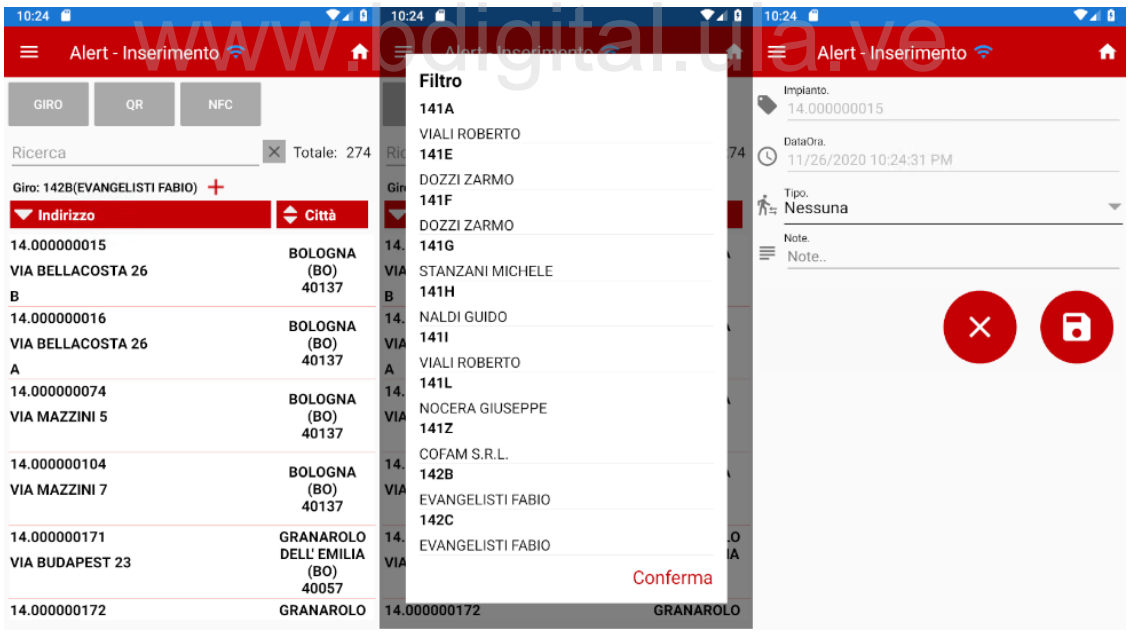


Figura 28: Alert - Inserimento (Alerta Ingresar).

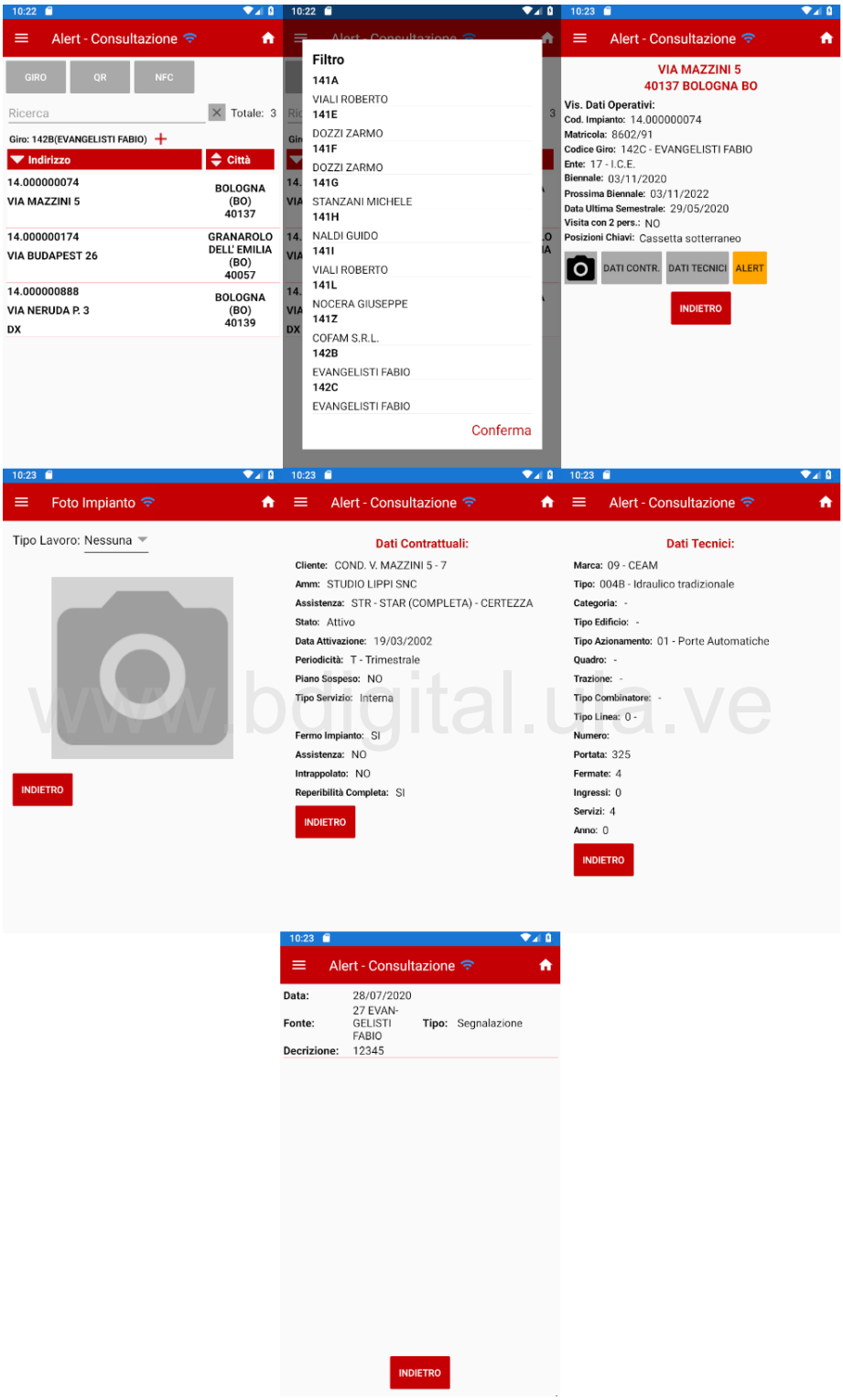


Figura 29: Alert - Consultazione (Alerta - Consultar).

Teniendo en cuanta que Datalog Italia Srl no estableció un diseño para seguir como base, para esto concedió libertad en los diseños, tomándolos colores del logo e integrándolos de manera que se en la aplicación de manera que fuese más atractiva para los clientes, además, cada que se culminaba el diseño de un módulos era enviado para su aprobación para proseguir con la implementación funcional de mismo.

4.2.2 Diseño de la estructura funcional.

Tomando en cuento el enfoque de la aplicación que sea multi-plataforma, Xamarin Forms es ideal para esto, es decir, el producto final tiene que funcionar tanto en Android y en iOS, esta decisión fue tomada por Datalog Italia Srl dada las necesidades que se necesitaba satisfacer.

Teniendo esto en cuenta la estructura del sistema a desarrollar es:

1. Desarrollar la aplicación usando el Framework Xamarin Forms en C# usando el patrón de arquitectura de software MVVC.
2. Definir los modelos que se van a usar, para la abstracción de los elementos.
3. Integración con la base de datos existente que se encuentran en los servidores de King.
4. Búsqueda de nuevos paquetes (gestor de paquetes Nuget de Visual Studio) para complementar los nuevos diseños.
5. Desarrollo de elementos necesarios para el diseño, que no se encuentra en el gestor de paquetes de Nuget.
6. Desarrollo de clases de tipo Interfaz o clases no abstractas, para después ser implementada en los proyectos de Android, iOS y UWP.

4.3 Fase de implementación.

Para la fase de implementación se inicia con tomando en cuenta todos los requerimientos, el tiempo propuesto para el diseño de la interfaz de usuario, para logar alcanzar los objetivos propuestos en el inicio, para esto se van a realizar las actividades siguiendo:

1. Preparación de la organización: el equipo de Datalog Italia Srl se encargar que buscar los recursos para necesaria para que el desarrollo se lleve a cabo de manera eficiente y sin problemas.

2. Planificación de la aplicación: los miembros del equipo se encargan de realizar el análisis y planificación de las actividades que se debe cumplir en el orden propuesto, para optimizar el tiempo de desarrollo.
3. Comunicación e información: intercambio de información entre el grupo de análisis, el encargado y el desarrollador, con el fin de aclarar las preguntas del análisis o para informar de algún cambio que se debe realizar, además, para aprobar una interfaz de usuario que se diseñó, para proseguir con el desarrollo de la misma, con esto se lleva un seguimiento del estado de la aplicación.
4. Elaboración de la documentación: el grupo de análisis se encarga de generar el documento de análisis, después e interpretar las necesidades del cliente y correcciones de las mismas, para que sea factible la elaboración de la aplicación.
5. Implementación: proceso donde se realizó el diseño de la interfaz de usuario y la estructura funcional de la aplicación.
6. Pruebas: etapa donde se encarga el grupo de pruebas en realizar rigurosas pruebas, con el fin de encontrar posibles errores de funcionalidad o anomalías visuales, al ser una aplicación móvil, estas pruebas son realizadas en varios móviles de diferentes marcas y modelos, versión del Android y resolución de la pantalla, además, de verificar que la funcionalidad de las misma sea correcta.

4.3.1 Implementación de la interfaz de usuario.

La interfaz de usuario es la conexión entre la aplicación y el usuario final, por esta razón el diseño debe ser agradable e intuitivo haciendo su uso lo menos complicado posible, de esta manera existe áreas dedicadas a este tema, por esta razón se decidió realizar un cambio en el diseño de la aplicación manteniendo su esencia inicial de la aplicación precedente a la cual se está desarrollando, para realizar los cambio se decidió colocar los colores del logo de Neulift (rojo, blanco y negro) en toda la aplicación de manera que resaltara y estilizara la misma.

Con la gran variedad de dispositivos móviles, esto conlleva a diferentes tipos de resolución, por lo mismo se decidió optar por un diseño simplista, que se adaptara a cada pantalla, manteniendo el aspectos de los componentes visuales, las tablas donde es visualizadas las listas de diferentes módulos se

decidió en algunos caso optar por colocar un scroll horizontal, debido a la gran cantidad de datos que esta debería ser visualizadas, teniendo en cuenta que se realizó un estudio con antelación para corroborar cuanto datos deben ser visualizados, como la cantidad de columnas que cada una debe poseer, de la misma manera se verifico la posición de los botones, etiquetas, etiquetas editables, checkbox. En algunos casos estos componentes se tuvieron que desarrollar desde cero, con el fin de cumplir con las características que se deseaban obtener en un inicio, como eran los popup, los cuales a la necesidad se modificaban y se les agregaban otros componentes visuales.

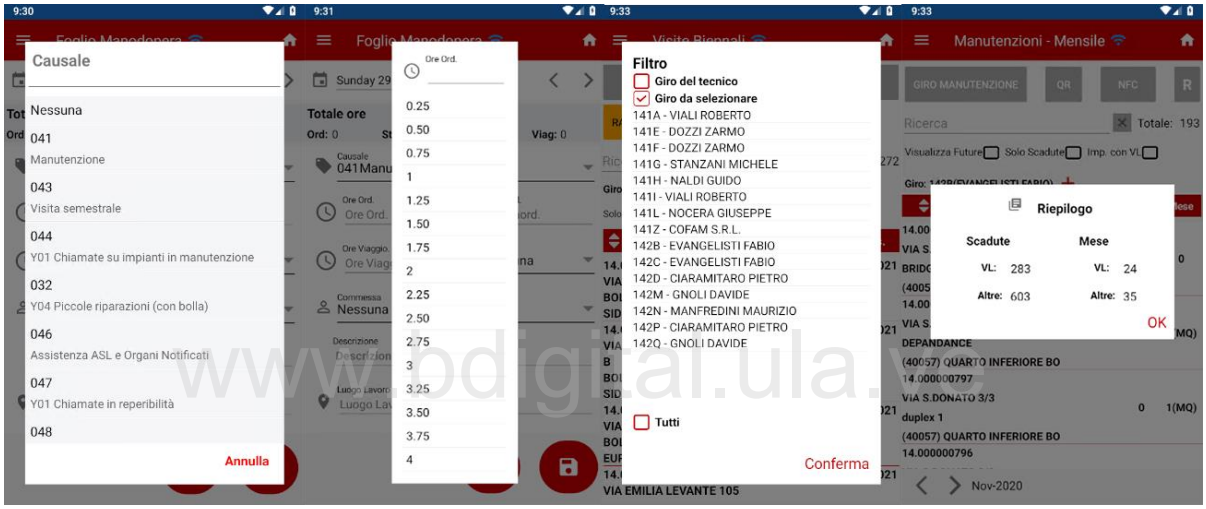


Figura 30: Diferentes tipos de PopUp's.

Para este desarrollo fue decidido usar Xamarin Forms un framework con el patrón de diseño de software MVVM en .NET C# y Forms, complementando el diseño visual de la aplicación, sin necesidad de colocar código dependiente de cada plataforma (Android, iOS o UWP) en la misma, dejando esto a Xamarin que realizara las acciones para cada plataforma final.

Teniendo esto en cuenta las validaciones de algunos campos como por ejemplo: el de ingreso de horas, se cambió su comportamiento estándar por uno específico para estos campos, como el símbolo del separador decimal y excluir cualquier carácter alfabético o carácter especial, en este caso dependiendo del idioma del dispositivo móvil, el separador decimal cambiaba entre el punto (**.**) o la coma (**,**), y el campo aunque fuese numérico aceptaba la letra **e**, lo cual no era correcto, de esta manera se cambiaba el comportamiento estándar de varios componentes para solucionar el problema, para ello se usaron **clases no abstractas** (clases interfaz), la cual se declaraba en el proyecto principal de Xamarin Forms y se

implementaban en los proyectos de Xamarin Android, Xamarin iOS o el proyecto de UWP, declarando su nuevo comportamiento.

```
namespace Neulift.Controls
{
    public class CustomEntry:Entry
    {
        public static readonly BindableProperty MaxLengthPtoperty = BindableProperty.Create("MaxLength",typeof(int), typeof(CustomEntry),int.MaxValue);
    }
}
```

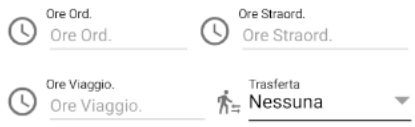
Definición de la clase no abstracta CustomEntry en el proyecto principal de Xamarin Forms Neulift.

```
[assembly: ExportRenderer(typeof(CustomEntry), typeof(CustomEntryRender))]
namespace Neulift.Droid
{
    public class CustomEntryRender: EntryRenderer
    {
        public CustomEntryRender(Context context) : base(context)
        {
            AutoPackage = false;
        }

        protected override void OnElementChanged(ElementChangedEventArgs<Entry> e)
        {
            base.OnElementChanged(e);
            if (this.Control == null) return;
            this.Control.KeyListener = DigitsKeyListener.GetInstance("1234567890,.");
        }

        private void SetMaxLength(CustomEntry view)
        {
            Control.SetFilters(new IInputFilter[]{ new InputFilterLengthFilter(view.MaxLength)});
        }
    }
}
```


Implementación de la clase no abstracta CustomEntry en el proyecto de Xamarin Android Neulift.



Componentes gráficos para las etiquetas editables de las horas.

Figura 31: Definición, implementación de la clase no abstracta y visualización de su componente.

4.3.2 Implementación de la estructura funcional.

Con la implementación de la estructura funcional de la aplicación se debe tener en cuenta la metodología elegida, llevando a cabo cada proceso para obtener el producto final, teniendo en cuenta que la distribución de las actividades fue pensada en función a la metodología, con esto en mente se procede al inicio del desarrollo de la aplicación.

Gran parte de la funcionalidad de Foglio Manodopera (hoja de trabajo), Calendario Lavoro (calendario de trabajo), Conferma Rapporto (confirmar reporte) y Riepilogo Rapporto (resumen de hora de trabajo) ya estaba desarrollada, por lo cual solo se optó por traducir dicha funcionalidad que se

encontraba en el lenguaje de programación Java y C# a solo lenguaje de programación C#, lo cual facilito la implementación funcional de estos módulos, en algunos casos, cuando el código tenía instrucciones propias de Android, se tuvo que buscar la formar que replicar dicha instrucciones en C# del proyecto principal de Xamarin Forms, sin tener que caer en el uso de instrucciones propias Android.

Como se puede observar en las siguientes imágenes del módulo de Calendario Lavoro (calendario de trabajo) como se realiza en la aplicación precedente Xamarin Android y en la aplicación en desarrollo Xamarin Forms:

```
public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{
    var view = inflater.Inflate(Resource.Layout.content_main, container, false);

    calendar = view.FindViewById<MaterialCalendarView>(Resource.Id.calendarView);


    calendar.SetOnDateChangedListener(this);
    ////calendar.SetOnDateChangedListener = this;
    calendar.AddDecorators(new GreenConditionViewDecorator(), new YellowConditionViewDecorator(), new RedConditionViewDecorator());

    Fab = view.FindViewById<FloatingActionButton>(Resource.Id.fab);
    Fab.Click += FabOnClick;

    return view;
}

class GreenConditionViewDecorator : Java.Lang.Object, IDayViewDecorator
{

    public bool ShouldDecorate(CalendarDay p0)
    {
        if (p0.Day == 4 || p0.Day == 10 || p0.Day == 20)
            return true;
        else
            return false;
    }

    public void Decorate(DayViewFacade p0)
    {
        p0.SetBackgroundDrawable(new ColorDrawable(Android.Graphics.Color.Rgb(37, 155, 36)));
    }
}

class YellowConditionViewDecorator ...

class RedConditionViewDecorator ...
```


Figura 32: Carga de datos para Calendario Lavoro (Calendario de trabajo) - Aplicación Precedente.

Para la cual el Calendario Lavoro (calendario de trabajo) se visualizaba de la siguiente manera en la aplicación precedente en Xamarin Android:

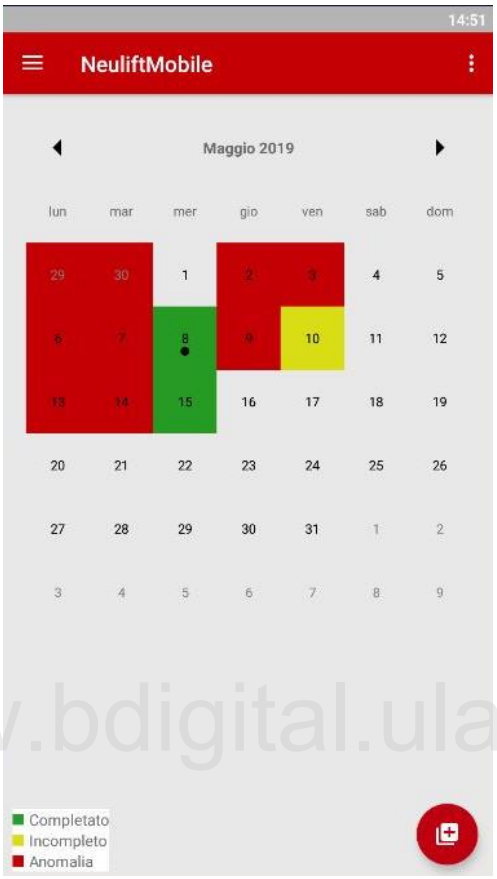


Figura 33: Calendario Lavoro (Calendario de Trabajo) - Aplicación Precedente.

En este caso se tuvo que buscar un paquete de calendario el cual permita colorear los días de un color especifico, para esto se tuvo que realizar un investigación de los posibles paquetes que cumplieran con esta especificación, el cual fue el paquete ***XamForms.Controls.Calendar de RebeccaXam***, este paquete se encuentra en el gestor de paquetes Nuget, en la siguiente imagen observaremos como se usó el paquete, para llegar a la solución:

```
calendarioLavoro.SpecialDates.Clear();
calendarioLavoro = new XamForms.Controls.Calendar();

List<SpecialDate> AllList = new List<SpecialDate>();
List<DataCountRapOre> ListConfRef = calendarioLavoroServ.ConfirmedRapOreCount;

AllList.AddRange(SpecialDays(calendarioLavoroServ.RedRapOreCount, calendarioLavoroServ.NonRedRapOreCount, calendarioLavoroServ.festivo, ref ListConfRef, Color.FromHex(Resource.COLORRED)));

AllList.AddRange(SpecialDays(calendarioLavoroServ.YellowRapOreCount, ref ListConfRef, Color.FromHex(Resource.COLORYELLOW)));
AllList.AddRange(SpecialDays(calendarioLavoroServ.GreenRapOreCount, ref ListConfRef, Color.FromHex(Resource.COLORGREEN)));
foreach (SpecialDate date in AllList)
{
    calendarioLavoro.SpecialDates.Add(date);
}
calendarioLavoro.RaiseSpecialDatesChanged();
calendarioLavoro.ForceRedraw();

Device.StartTimer(TimeSpan.FromSeconds(0.5), () =>
{
    leggenda.IsVisible = true;
    return false;
});
```

Figura 34: Carga de los datos para visualizar en Calendario Lavoro (calendario de trabajo) - Aplicación en Desarrollo.

Como se puede observar en la imagen anterior (***ver figura 34***), resulto ser más sencillo llegar a la carga de los datos, obteniendo como resultado:

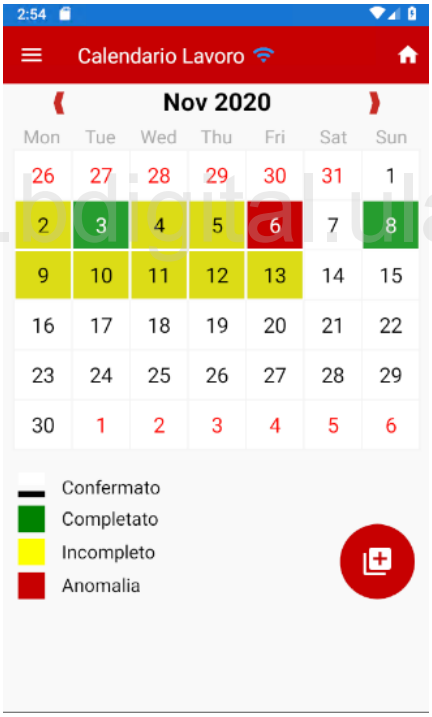


Figura 35: Calendario Lavoro (Calendario de trabajo) - Aplicación en Desarrollo.

Obteniendo un mejor resultado en Xamarin Forms que en la que se visualizaba en Xamarin Android. De la misma manera como se realizó en Calendario Lavoro (calendario de trabajo) hubo varios casos similares, como el Snakcbar que este componente no existe en Xamarin Forms, pero se realizó desde cero este componente, en este caso dependía de la plataforma (Android):

```
Snackbar.Make(btn, "Offline Login", Snackbar.LengthLong)
.SetAction("Action", (Android.Views.View.IOnClickListener)null).Show();
```

Figura 36: Snackbar - Aplicacion Precedente.

Como observamos en la siguiente imagen parte del código y su implementación:

```
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class SnackBar : TemplatedView
{
    public static readonly BindableProperty ButtonTextColorProperty = BindableProperty.Create("ButtonTextColor", typeof(Color), typeof(SnackBar), default(Color));
    public Color ButtonTextColor
    {
        get { return (Color)GetValue(ButtonTextColorProperty); }
        set { SetValue(ButtonTextColorProperty, value); }
    }

    public static readonly BindableProperty MessageProperty = BindableProperty.Create("Message", typeof(string), typeof(SnackBar), default(string));
    public string Message
    {
        get { return (string)GetValue(MessageProperty); }
        set { SetValue(MessageProperty, value); }
    }

    public static readonly BindableProperty CloseButtonTextProperty = BindableProperty.Create("CloseButtonText", typeof(string), typeof(SnackBar), "Close");
    public string CloseButtonText
    {
        get { return (string)GetValue(CloseButtonTextProperty); }
        set { SetValue(CloseButtonTextProperty, value); }
    }

    public static readonly BindableProperty FontSizeProperty = BindableProperty.Create("FontSize", typeof(float), typeof(SnackBar), default(float));
    public float FontSize
    {
        get { return (float)GetValue(FontSizeProperty); }
        set { SetValue(FontSizeProperty, value); }
    }

    public static readonly BindableProperty TextColorProperty = BindableProperty.Create("TextColor", typeof(Color), typeof(SnackBar), Color.White);
    public Color TextColor
    {
        get { return (Color)GetValue(TextColorProperty); }
        set { SetValue(TextColorProperty, value); }
    }
```


Figura 37: Parte del código del Snackbar en Xamarin Forms - Aplicación en Desarrollo.

```
SnackB.Message = ex.Message;
SnackB.IsOpen = !SnackB.IsOpen;
```

Figura 38: Implementación del Snackbar - Aplicación en Desarrollo.

Para los módulos de Manutenzioni (mantenimientos), Visite Biennali (visita bianual) y Alert (alerta), se realizó el estudio de los documentos de análisis para cada módulo, para comprender cuál debería ser su comportamiento, como la carga de datos a través de API, al guardado y la modificación de los mismos, sin embargo cabe acotar, que para el diseño se tuvo libre albedrio para desarrollarlo, pero esto deben ser aprobados antes de seguir con algún otro modulo.

Resaltado que la comunicación con el servidor se realiza a través de una API ya desarrollada, que solo se agregan nuevas características para ser utilizados en los nuevos módulos.

Por motivos de acuerdos de confidencialidad el código no puede ser mostrado, solo aquel que la empresa Datalog Italia Srl considere irrelevante que sea expuesto en el presente proyecto de grado.

Capítulo 5

Conclusiones y Recomendaciones.

Una vez culminado el desarrollo de la aplicación en Xamarin Forms Neulift de la empresa Datalog Italia Srl se procede a exportar la aplicación a la plataforma deseada en este caso para Android, con el fin de presentar el producto final al cliente, con el fin que la revise y apruebe la aplicación, para este momentos la aplicación se encuentra en la versión 2 donde se agregaron otros módulos en la que se encuentra desarrollando, además a continuación se detallara la conclusión de los módulos desarrollado para presentar el proyecto de grado, así como el aporte y las recomendaciones del mismo.

5.1 Conclusiones.

La empresa Datalog Italia Srl decidió realizar el desarrollo de una aplicación móvil para la empresa Neulift, de un módulo de KING, que se encarga del registro de horas de trabajo, resúmenes de reportes, confirmación de reportes, mantenimientos, alertas, entre otros; con el fin de satisfacer la necesidad del cliente de realizar estas operaciones desde la comodidad de un dispositivo móvil, ya sea Android o iOS, teniendo esto en cuenta, Datalog Italia Srl realizo un primera aplicación en Xamarin Android, con funciones limitadas, vista que esta cumplía con parte del objetivo, se decidió realiza el desarrollo en otro framework de desarrollo multi plataforma como lo es Xamarin Forms, cumpliendo con el objetivo de poder realizar el desarrollo solo una vez y luego se exportado en Android y iOS.

Con esto en mente se decidió comenzar con el desarrollo, colocando los módulos ya existentes y agregando otros, que se requerían por pedido del cliente, de esta manera se alcanzaron los objetivos propuestos y en el tiempo acordó por la empresa.

La culminación de la aplicación en el tiempo acordado y con una interfaz de usuario fácil e intuitivo, complació al cliente lo cual solicito a la empresa Datalog Italia Srl, que se agregaran el resto lo los módulos a la aplicación, para esto se llegó a un acuerdo cuales módulos tenían prioridad sobre otros.

Cabe resalta que todos los módulos ya se encuentran desarrollado y en completo funcionamiento en el software KING (software administrativo contable) que es usado por la empresa Neulift. Al poseer la aplicación móvil, independiente del Sistema Operativo que se encuentre (Android o iOS), el usuario puede ingresar los datos de manera cómoda y estos ser reflejados en la aplicación de escritorio (KING).

5.2 Aportes.

En el desarrollo de la aplicación se obtuvo dos grandes aportes: laboral y personal. Cuando se refiere a aporte personal:

- En el transcurso que se desarrollaba la aplicación, se obtuvo conocimiento para resolver los problemas que iban saliendo a luz, referente al ámbito de la programación.
- Adquisición de nuevos conocimiento por parte de los encargados de proyecto, en específico, como abordar el problema, como resolverlos de manera eficiente.
- Colocar en práctica los conocimientos que se obtuvieron a lo largo de la carrera, como la creación de diferentes tipos de documentación, diagramas entre otros, con el fin de resolver los objetivos que se planteaban.
- Reconocimiento por parte de la empresa cuando se culminada cada fase del proyecto y en el nivel de profesionalismo que es inculcado en las enseñanza de nuestra universidad.

Se puede enlistar una serie de aportes laborales, que hacen énfasis el desarrollo de aplicación y la empresa:

- El desarrollo de la aplicación facilito a los usuarios el ingreso de los datos de manera más sencilla de lo que se venía haciendo.
- El desarrollo de una aplicación que tiene los módulos de KING, hace ver que se puede realizar el desarrollo de diferentes aplicaciones móviles, de diferentes módulos, para clientes específicos de Datalog Italia Srl.

5.3 Recomendaciones.

Después del desarrollo de la aplicación se observaron una seria de problemas, que ya al final no fueron graves, se pudieron abordar de diferente manera obteniendo un resultado mucho más sencillo, se enlista una serie de recomendaciones , que se pueden aplicar para los próximos trabajos o para el próximo desarrollo de la aplicación de la misma:

- Limpieza del código del proyecto, de manera que se puede disminuir el tamaño de la aplicación cuando es exportada a cualquiera de las plataformas.
- Eliminar los archivos que no son usados, como son las imágenes o en algunos casos bajar la resolución de las mismas.
- Dedicar un grupo para la realización de las pruebas de la aplicación, de manera que se pueda obtener la retroalimentación de los errores en mucho menor tiempo.
- Dedicar un grupo al desarrollo de los documentos de los análisis para facilitar el desarrollo de la misma, sin crear algún retraso en la fecha de entregas de los sprints.

Bibliografía

Proyestosagiles.org, *Qué es SCRUM* [en línea], 2008.

Disponible en < https://proyectosagiles.org/que-es-scrum/ >

[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *Microsoft Visual Studio 2017* [en línea], 2019.

Disponible en < https://visualstudio.microsoft.com/vs/ >

[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *IntelliSense* [en línea],

Disponible en < https://es.wikipedia.org/wiki/IntelliSense >

[Consulta: 10 Noviembre 2019]

C Sharp (C#) [en línea], 2000.

Disponible en < https://es.wikipedia.org/wiki/C_Sharp >

[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *C# Documentation* [en línea],

Disponible en < https://docs.microsoft.com/en-us/dotnet/csharp/ >

[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *Overview of .NET Framework* [en línea],

Disponible en < https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview >

[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *Microsoft .NET* [en línea], 2002
Disponible en < https://es.wikipedia.org/wiki/Microsoft_.NET >
[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *Introduction to the C# language and the .NET Framework* [en línea], 2015
Disponible en < https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework >
[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *Xamarin* [en línea], 2015
Disponible en < https://es.wikipedia.org/wiki/Xamarin >
[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *Xamarin* [en línea], 2015
Disponible en < https://dotnet.microsoft.com/apps/xamarin >
[Consulta: 10 Noviembre 2019]

Microsoft Corporation, *What is Xamarin?* [En línea], 2020
Disponible en < https://dotnet.microsoft.com/apps/xamarin >
[Consulta: 07 Junio 2020]

Programación orientada a objetos [En línea], 2015
Disponible en < https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos >
[Consulta: 07 Junio 2020]

Servicio de InformáticaASP.NET MVC 3 Framework, *Modelo Vista Controlador (MVC)* [En línea], 2015
Disponible en < https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html >
[Consulta: 07 Junio 2020]

ECURED, *Caso de uso* [En línea], 2019
Disponible en < https://www.ecured.cu/Caso_de_uso >
[Consulta: 07 Junio 2020]

rebeccaXam, *Calendar Control Plugin for Xamarin.Forms* [En línea], 2015
Disponible en < https://github.com/rebeccaXam/XamForms.Controls.Calendar >
[Consulta: 07 Junio 2020]