Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Herramienta de gestión, cálculo y control en laboratorio del límite de residuos de medicamentos veterinarios y contaminantes en alimentos

Estudiante: David Martos Dourado
Dirección: Carlos Fernández Lozano
Dirección: Carolina Nebot García

A Coruña, setembro de 2021.

Dedicado a mi familia y amigos

Resumen

En este proyecto se trata el desarrollo de una aplicación multiplataforma que permite organizar los distintos datos obtenidos durante la validación de un método analítico, con el objetivo de poder controlar los límites de residuos y contaminantes establecidos en alimentos. Esta aplicación permite verificar que un método analítico desarrollado en un laboratorio funciona adecuadamente y cumple con la legislación vigente. Todo ello se consigue insertando en la aplicación de forma automática los datos obtenidos de diferentes muestras empleadas para validar el método. A continuación la aplicación realiza una serie de cálculos estadísticos y da cómo resultado determinados parámetros que permiten evaluar el método.

Abstract

In this project the development of a cross-platform application is studied, which allows to organize the different values obtained during the validation of an analytical method, in order to control the established limits of residue and contaminants on food. This tool allows to verify that an analytical method developed in a laboratory works correctly and comply with the law. This is accomplished by automatically inserting in the application the data obtained from different samples used to validate the method. After that, the tool performs a series of statistical calculations and obtains certain properties that can allow the method to be evaluated.

Palabras clave:

- Multiplataforma
- Aplicación de escritorio
- Base de datos relacional

Keywords:

- Cross-platform
- Desktop application
- Relational database

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1 Contexto

El uso de sustancias con actividad farmacológica en animales productores de alimentos está permitido dentro de la Unión Europea, pero siempre que se garantice la salud pública. Es por ello por lo que no todas las sustancias están permitidas, y para aquellas permitidas se ha establecido lo que se denomina el límite máximo residual (LMR). El LMR se puede definir como la concentración máxima permitida de una sustancia farmacológica en un alimento concreto (músculo, hígado, riñón, grasa, huevo, leche, miel, etc.) la cual al ser ingerida por el ser humano puede constituir un riesgo para su salud. Los LMR para los diferentes alimentos de origen animal están recogidos en el Reglamento 470/2009 [1].

Para garantizar la seguridad alimentaria, la concentración de sustancias farmacológicas, así como de contaminantes, debe de ser controlada por los diferentes laboratorios tanto públicos como privados. Los laboratorios deben de garantizar que las concentraciones de sustancias activas, fármacos o contaminantes, sean inferiores a los LRM establecidos por la legislación vigente.

Antes de determinar si la concentración de un determinado fármaco o contaminante es correcta se debe de verificar que el método empleado es adecuado y que todos los pasos que se realizan para la determinación llevan a una cuantificación adecuada. Al no haber unos métodos de análisis oficiales y estándares cada laboratorio puede emplear su propio método. Pero para ello debe de demostrar que es válido para dicho análisis. La validación del método consiste en la realización del protocolo de análisis en una serie de muestras (un mínimo de 128 muestras) las cuales tienen el o los compuestos a determinar a unas concentraciones concretas. Con el fin de establecer criterios y procedimientos únicos de validación sobre los métodos analíticos, así como sobre la interpretación de resultados generados por los distintos laboratorios dedicados al control de alimentos dentro de la UE se aprobó el Reglamento 2021/808 [2], que no solo indica cómo se deben interpretar los resultados sino también cómo debe de

llevarse a cabo la validación.

Teniendo en cuenta la concentración real de los fármacos en las muestras y el valor obtenido durante la validación, se aplica una serie de fórmulas estadísticas y se obtienen una serie de parámetros comúnmente conocidos en química analítica que indican la precisión y exactitud, así como otras variables del método empleado que nos indicará si es adecuado o no el método para dicho análisis o análisis inequívoco de la/s sustancia/s. La validación es una parte vital de un método de análisis ya que nos garantiza la fiabilidad del método, y en el caso concreto de la determinación de residuos de sustancias farmacológicas en alimentos la presencia de determinadas sustancias no autorizadas o permitidas podrían conllevar un riesgo para la salud y por tanto una sanción penal para el productor del alimento.

La validación no solo es uno de los aspectos más importantes sino también uno de los más laboriosos ya que el número de muestras que se requiere es elevado, sin embargo, y teniendo en cuenta los conocimientos de las personas que trabajan en los laboratorios de control de residuos la parte más tediosa de la validación se centra en la evaluación de los resultados de la validación, es decir, la aplicación de fórmulas matemáticas para obtener parámetros tales como repetibilidad, reproducibilidad, rendimiento, ... del método.

El Laboratorio de Higiene, Inspección y Control de Alimentos (LHICA-USC) [3] de la Universidad de Santiago de Compostela en Lugo, entre numerosas otras tareas se dedican al control de la presencia de residuos y contaminantes en alimentos. Para ello el LHICA debe disponer de numerosos métodos analíticos validados y que garanticen la identificación y cuantificación correcta de las sustancias en los alimentos. Cada día surgen nuevas alertas y nuevas técnicas que hace que el laboratorio esté habitualmente poniendo métodos analíticos nuevos a punto y realizando validaciones de dichos métodos.

Una validación adecuada se debe de realizar en un mínimo de tres días, y tal como se mencionó anteriormente con un mínimo de 128 muestras. Según la legislación vigente, la validación se debe de realizar empleando como referencia el LMR para aquellas sustancias que tengan límite máximo residual. Por otro lado, para aquellas sustancias que no tengan un LMR en un determinado alimento la validación se realizará a la menor concentración posible que se puede determinar la sustancias de forma precisa e inequívoca. Ya sea para sustancias con LMR o que no lo tengan la concentración a la cual se realiza la validación se denomina nivel de validación (NV).

En general, la validación se debe de realizar en un mínimo de tres días y a un mínimo de tres concentraciones diferentes, empleando para cada concentración un mínimo de seis muestras. Asimismo, se emplearán una serie de muestras blancos, es decir, muestras que no tienen la sustancia a determinar y muestras dopadas a 2 concentraciones superiores a las anteriores. En términos generales la legislación establece que las concentraciones que se deben de emplear para la validación del método deberán de ser las siguientes:

- Una muestras blanco
- Seis muestras de concentración 0,5 x el nivel de validación
- Seis muestras de concentración 1 x el nivel de validación
- Seis muestras de concentración 1,5 x el nivel de validación
- Una muestra de concentración 2 x el nivel de validación
- Una muestra de concentración 5 x el nivel de validación
- Seis muestras dopadas a diferentes concentraciones para cuantificar las muestras anteriores.

Para cada una de las muestras tomadas anteriormente es necesario obtener la medición de concentración que proporciona el equipo. Actualmente, este resultado se anota a mano en diversas hojas de cálculo preparadas por algunos laboratorios. Es esta parte de inserción manual de datos es donde recae bastante carga de trabajo y donde pueden surgir errores a la hora de copiar los resultados obtenidos. Además, cabe destacar que existen varias formas correctas de realizar la validación por lo que los cálculos obtenidos de una hoja de cálculo pueden ser diferentes, lo que dificulta el mantenimiento de varios métodos de cálculo en un mismo lugar.

Asimismo, hay que mencionar que la tendencia actual es el desarrollo de métodos multiresiduales, es decir, métodos que permitan determinar varias sustancias de una sola vez, lo que complica aún más el análisis de los datos de la validación. El procesado de la muestra sería el mismo para todos los analitos, pero la respuesta del equipo varía de un analito a otro, y por tanto los datos obtenidos para cada analito son diferentes. Sería como multiplicar cada una de las 121 muestras empleadas para la validación por el número de analitos que se quiera determinar.

1.2 Motivación

Dado que las herramientas actualmente disponibles no son óptimas, surge la posibilidad de crear una herramienta software para el grupo de investigación LHICA, que responda a todas las necesidades de esta serie de procesos. Tal herramienta podría hacerse cargo de gran parte del proceso de forma totalmente automática. Esto incluiría la inserción de datos de las muestras a analizar, el almacenamiento del histórico de grupos de experimentos relacionados, la implementación de distintos métodos de cálculo y la exportación de resultados del proceso a varios formatos; todo desde un mismo lugar. Además de estas funcionalidades, este software podría disponer de gráficas y elementos interactivos que permitan visualizar los resultados

de manera intuitiva y sin dificultades. En resumen, este software aliviaría la dificultad del proceso, puesto que la parte química ya trae muchas dificultades de por sí.

1.3 Objetivos

Los objetivos que se desean conseguir con este proyecto pueden separarse en dos ramas principales, la automatización de los distintos procesos realizados en la metodología de laboratorio, y el análisis de los resultados obtenidos.

Respecto a la automatización, es un objetivo muy importante a cumplir, ya que la principal ventaja que ofrece esta aplicación es facilitar el proceso manual que se realiza a día de hoy al tratar con esta serie de experimentos y muestras. Por una parte es necesario facilitar la inserción de las muestras y experimentos, tratando de evitar posibles errores humanos al realizar esta tarea. Por otra, también es importante automatizar la ordenación y agrupación de las distintas muestras ya que pueden pertenecer a distintos proyectos con objetivos diferentes.

Para conseguir esta automatización ha sido necesario desarrollar una parte del software que se encargue de analizar los datos generados por el equipamiento de laboratorio para insertarlos de forma automática en la aplicación agrupando estas muestras según donde aplique.

En cuanto al análisis de resultados, es importante desarrollar funcionalidades interactivas que permitan visualizar el comportamiento de los datos de manera intuitiva así como de monitorizar el estado de las muestras obtenidas mediante gráficas. Además, el desarrollo de herramientas de exportación de datos es muy importante para poder compartir la información extraída de los experimentos entre diferentes laboratorios y otras partes interesadas.

Por último, se desea que la aplicación pueda ser utilizada en distintos laboratorios en un futuro, sin necesidad de tener conexión a internet, por lo que se utilizarán tecnologías multiplataforma de escritorio para conseguir este objetivo.

1.4 Estructura de la memoria

Esta memoria está divida en varios capítulos. Primero se realizará un análisis del estado del arte de este tipo de herramientas de validación. A continuación se describirán las tecnologías y herramientas claves en el desarrollo de este proyecto y las razones de su elección. Después se estudiará la metodología de trabajo utilizada y cómo se adaptará para este proyecto en concreto. La parte central de la memoria consistirá en la explicación de las fases de diseño e implementación del proyecto. Por último se procederá a concluir la memoria analizando los objetivos alcanzados y el trabajo futuro a realizar.

1.5 Plan de trabajo

El desarrollo de la aplicación será dividido en varias fases. En primer lugar se realizará un análisis de los objetivos y necesidades del equipo de LHICA para escoger la tecnología y herramientas que mejor se adecúen a su situación. A continuación se crearán los diagramas de diseño pertinentes así como ”mockups” de la interfaz de usuario que faciliten la organización de los datos y flujo de la información en las fases iniciales del proyecto. Tras esta fase de diseño inicial, la implementación consistirá en varios sprints con funcionalidad mínima lanzable que se validarán con parte del equipo de LHICA como forma de seguimiento. Durante la implementación también se realizarán ciertas pruebas software mínimas, sobretodo haciendo mayor énfasis en los métodos de cálculo de resultados y en la importación de datos. Finalmente, se instalará la aplicación en LHICA en modo de pruebas para obtener feedback y observar de qué manera afecta el uso de esta herramienta en la productividad del proceso de control de sustancias.

Capítulo 2

Estado del arte

Dado que este procedimiento es muy específico, no ha habido mucho trabajo en el desarrollo de herramientas para su gestión y automatización. En este caso en concreto, el laboratorio de la USC, utiliza tres herramientas distintas para facilitar parte del proceso, todas ellas basadas en hojas de cálculo convencionales creadas ad hoc de forma manual. Además, en algunos experimentos se debe de usar más de una herramienta, dependiendo del tipo de análisis y muestra, o para verificar la corrección de los resultados.

2.1 Herramienta 1

Esta hoja de cálculo es una adaptación de otra hoja creada originalmente por un laboratorio polaco con el que colabora el laboratorio de la USC y cuenta con varias páginas preparadas para la inserción manual de datos para la obtención de los resultados de la validación y de tablas resumen. Dado que es una adaptación, existen múltiples errores e inconsistencias debidos al proceso de cambio y adaptación, dejando detrás funciones obsoletas, texto en polaco, valores de cálculo intermedios innecesarios y, en general, un desorden en la organización de los datos y fórmulas. Véase la figura 2.1 donde se observa que la hoja no está preparada para tratar más de tres días de validación, mostrando los datos de entrada del día 4 después de una tabla de resultados. Además de la existencia de varias celdas con "#VALUE!", que indica que hay errores en algunas fórmulas. Por último en la figura 2.2 se observan varias secciones sin traducir. Además de esto cabe destacar que de todos los valores y cálculos obtenidos, solo una pequeña parte es relevante.

2.2 Herramienta 2

La siguiente herramienta es conocida como ResVal© [4] y se trata de una hoja de cálculo desarrollada por un equipo de Países Bajos. Para poder utilizar esta hoja es necesario adquirir

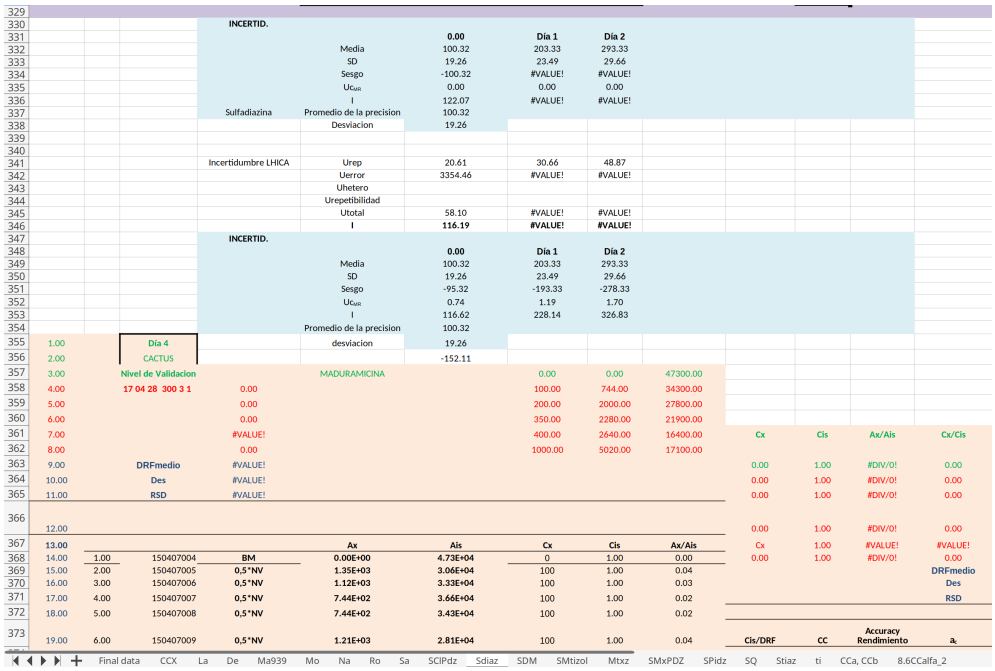
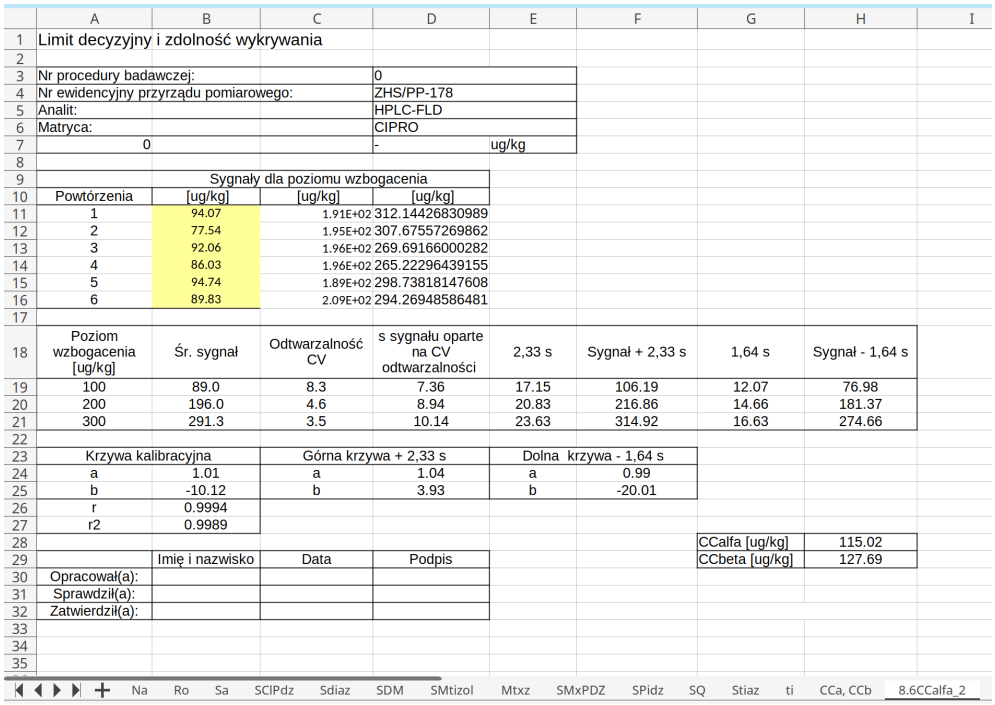
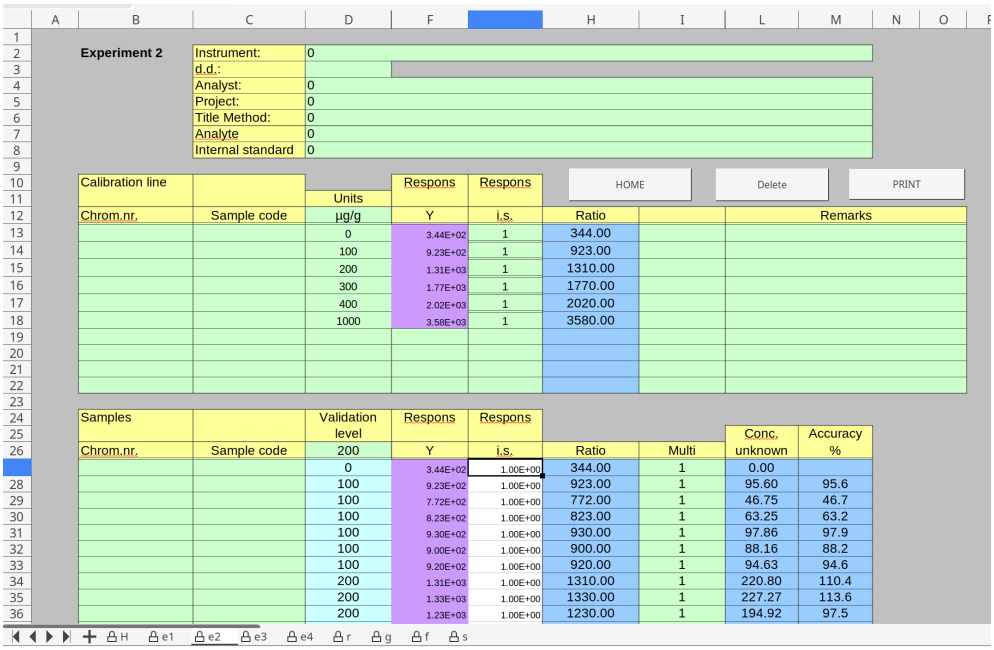


Figura 2.1: Desorden y errores



	A	B	C	D	E	F	G	H	I
1	Limit decyzyjny i zdolność wykrywania								
2									
3	Nr procedury badawczej:			0					
4	Nr ewidencyjny przyrządu pomiarowego:			ZHS/PP-178					
5	Analit:			HPLC-FLD					
6	Matryca:			CIPRO					
7	0			- ug/kg					
8									
9		Sygnały dla poziomu wzbogacenia							
10	Powtórzenia	[ug/kg]	[ug/kg]	[ug/kg]					
11	1	94.07	1.91E+02	312.14426830989					
12	2	77.54	1.95E+02	307.67557269862					
13	3	92.06	1.96E+02	269.69166000282					
14	4	86.03	1.96E+02	265.22296439155					
15	5	94.74	1.89E+02	298.73818147608					
16	6	89.83	2.09E+02	294.26948586481					
17									
18	Poziom wzbogacenia [ug/kg]	Śr. sygnał	Odtwarzalność CV	s sygnału oparte na CV odtwarzalności	2,33 s	Sygnał + 2,33 s	1,64 s	Sygnał - 1,64 s	
19	100	89.0	8.3	7.36	17.15	106.19	12.07	76.98	
20	200	196.0	4.6	8.94	20.83	216.86	14.66	181.37	
21	300	291.3	3.5	10.14	23.63	314.92	16.63	274.66	
22									
23	Krzywa kalibracyjna		Górna krzywa + 2,33 s		Dolna krzywa - 1,64 s				
24	a	1.01	a	1.04	a	0.99			
25	b	-10.12	b	3.93	b	-20.01			
26	r	0.9994							
27	r2	0.9989							
28							CCalfa [ug/kg]	115.02	
29		Imię i nazwisko	Data	Podpis			CCbeta [ug/kg]	127.69	
30	Opracował(a):								
31	Sprawdził(a):								
32	Zatwierdził(a):								
33									
34									
35									

Figura 2.2: Secciones en polaco



		Experiment 2	Instrument:	0										
			d.d.:											
			Analyst:	0										
			Project:	0										
			Title Method:	0										
			Analyte	0										
			Internal standard	0										
		Calibration line			Respons	Respons		HOME		Delete		PRINT		
				Units										
		Chrom.nr.	Sample code	µg/g	Y	i.s.	Ratio		Remarks					
				0	3.44E+02	1	344.00							
				100	9.23E+02	1	923.00							
				200	1.31E+03	1	1310.00							
				300	1.77E+03	1	1770.00							
				400	2.02E+03	1	2020.00							
				1000	3.58E+03	1	3580.00							
		Samples		Validation level	Respons	Respons								
		Chrom.nr.	Sample code	200	Y	i.s.	Ratio	Multi	Conc. unknown	Accuracy %				
				0	3.44E+02	1.00E+00	344.00	1	0.00					
				100	9.23E+02	1.00E+00	923.00	1	95.60	95.6				
				100	7.72E+02	1.00E+00	772.00	1	46.75	46.7				
				100	8.23E+02	1.00E+00	823.00	1	63.25	63.2				
				100	9.30E+02	1.00E+00	930.00	1	97.86	97.9				
				100	9.00E+02	1.00E+00	900.00	1	88.16	88.2				
				100	9.20E+02	1.00E+00	920.00	1	94.63	94.6				
				200	1.31E+03	1.00E+00	1310.00	1	220.80	110.4				
				200	1.33E+03	1.00E+00	1330.00	1	227.27	113.6				
				200	1.23E+03	1.00E+00	1230.00	1	194.92	97.5				

Figura 2.3: ResVal©

una licencia de pago a través de su página web. Actualmente esta licencia se puede obtener mediante un pago único de 100 €. Esta herramienta está más organizada y estructurada pero el principal problema es que no es editable, por lo que no es posible realizar cambios cuando sea necesario (más días de validación o método de cálculo diferente). En la figura 2.3 se muestra una captura de esta herramienta.

2.3 Herramienta 3

Por último, el equipo de LHICA ha desarrollado una hoja de cálculo simplificada. Esta herramienta solo incluye el cálculo de CCα y CCβ así como la recta de regresión de las muestras. Esta simplificación está optimizada para el método de trabajo concreto que realizan en el laboratorio de la USC. En la figura 2.4 se muestra una captura de la hoja de cálculo.

2.4 Conclusiones

A pesar de que las herramientas utilizadas aceleran parte del proceso, todas ellas comparten limitaciones fundamentales debido a que están basadas en hojas de cálculo. Para cada serie de experimentos se debe mantener un archivo distinto, y si en un futuro fuese necesario algún cambio en el procedimiento de validación, las modificaciones pertinentes en la hoja de cálculo tendrían que ser propagadas a todos los archivos. Otro problema de estas soluciones

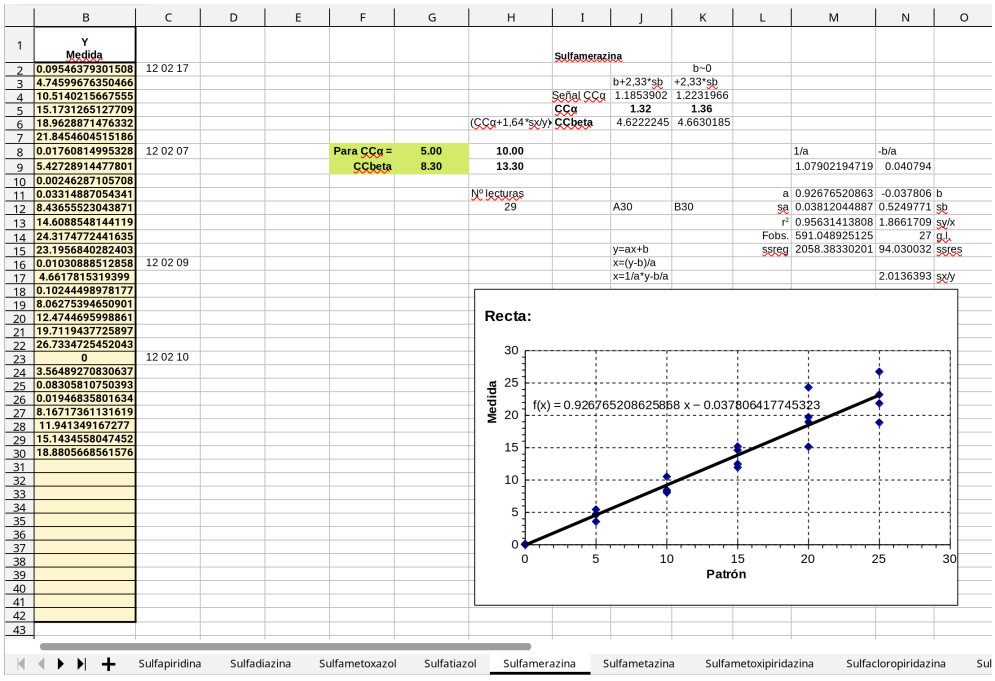


Figura 2.4: Hoja de cálculo del laboratorio

es la entrada manual de datos, muy propensa a errores. Las hojas de cálculo no dejan bien delimitadas las partes que son de entrada de datos y las que son generadas por la herramienta, por lo que el usuario podría llegar incluso a borrar elementos importantes. Otro detalle es que las hojas de cálculo distribuyen la información en una cuadrícula, lo que provoca mucho espacio en blanco y la estructura de la misma dificulta la realización de una interfaz de usuario óptima. Todos estos factores motivan la creación de una herramienta automática y optimizada para el proceso. Ésta será capaz de tratar de forma automática todos los datos a partir de ficheros generados por el método de análisis utilizado en LHICA, simplificando toda la tarea de análisis de sustancias, así como la gestión de los experimentos realizados, pudiendo acceder fácilmente a los resultados obtenidos en el pasado.

En este proyecto, la inserción automática de datos se enfocará en el análisis de ficheros generados por la instrumentación de laboratorio utilizada en LHICA, en concreto un espectrómetro de masas, máquina capaz de extraer la concentración de una sustancia en función de su masa. No obstante, estos ficheros podrían variar en función del método de medición e instrumentos utilizados, que varían de un laboratorio a otro. Para que la herramienta pueda importar de forma automática los datos de múltiples laboratorios sería necesario un formato intermedio conocido por la aplicación, en forma de CSV u hoja de cálculo. Esto implicaría un esfuerzo adicional por parte del laboratorio para generar este formato intermedio, volviendo de nuevo a la posibilidad de cometer errores humanos. Por ello, este trabajo se centrará en

la importación de los ficheros generados en LHICA para hacer el proceso lo más automático posible, dejando la posibilidad de ampliar la capacidad de importación en el futuro.

Capítulo 3

Tecnologías y herramientas

En este capítulo se procederá a describir el conjunto de herramientas y tecnologías que fueron fundamentales para el desarrollo del proyecto. Se describirán además brevemente herramientas y software de apoyo utilizados que facilitaron la experiencia de desarrollo y planificación del proyecto.

3.1 Flutter

Uno de los objetivos a alcanzar es la capacidad de poder ejecutar la aplicación final en el mayor número de dispositivos posibles y así aumentar la compatibilidad con diversidad de equipos en los laboratorios. Además, cumpliendo el requisito de que pueda ser utilizada sin conexión a internet y de forma portable, requisito necesario dado que en ocasiones se toman y analizan muestras desde distintas ubicaciones. La portabilidad de una aplicación proporciona además una forma de uso que no requiere instalaciones o despliegues complejos. Es por ello que se ha optado por una tecnología de desarrollo de aplicaciones multiplataforma de escritorio.

A día de hoy existen diferentes formas de desarrollar una aplicación multiplataforma y aunque no hay una clara separación pueden diferenciarse en las basadas en tecnologías Web y en las aplicaciones tradicionales con código nativo o compilado. Para esta aplicación se ha decidido utilizar Flutter [5], una tecnología creada por Google que en sus inicios fue desarrollada para la creación de aplicaciones móviles multiplataforma (Android e iOS), pero que a día de hoy proporciona la capacidad para compilar a código nativo (no basado en tecnologías web) para Windows, Linux y macOS.

La principal razón por la que se ha escogido esta tecnología, a pesar de estar en beta para plataformas de escritorio, es por la gran eficiencia que ofrece a la hora de desarrollar nuevas funcionalidades teniendo en cuenta todos los entornos al mismo tiempo. Esta eficiencia fue observada con anterioridad por el equipo de desarrollo en otro proyecto profesional donde se

utilizó Flutter para el desarrollo de una aplicación móvil, ManaBox [6], por lo que esta experiencia previa ha ayudado mucho en la decisión. En concreto, se utilizó Flutter para migrar una aplicación hecha para Android con tecnologías nativas (Java y Kotlin) en poco menos de tres meses, manteniendo toda la funcionalidad anterior y extendiendo el soporte a la plataforma iOS, utilizada por los iPhones y iPads de Apple, y todo esto sin provocar ningún gran cambio estructural del código fuente debido a la unificación de las plataformas.

En los siguientes apartados se procederá a describir en más detalle el lenguaje de programación utilizado en las aplicaciones Flutter (Dart) así como las características del propio framework.

3.2 Lenguaje de programación

El lenguaje con el que se programa una aplicación en Flutter es Dart [7], también diseñado por Google en 2011 cuyo principal objetivo en sus inicios era ofrecer una alternativa a Javascript en el navegador. A pesar de no haber cumplido con su objetivo, la elección por parte del equipo de Flutter de utilizar Dart como lenguaje ha hecho que éste haya aumentado en popularidad en los últimos años. Dart es un lenguaje orientado a objetos, con tipado fuerte y estático, muy similar a Java y a C en cuanto a su sintaxis. Entre las características de Dart se pueden destacar los tipos ”nullables” como ciudadanos de primer orden, que facilitan la detección de errores muy típicos al intentar utilizar variables que no tienen valor en un momento determinado. Esta capacidad está además completamente integrada en la fase de compilación, por lo que ofrece optimizaciones a nivel de código máquina, consiguiendo así binarios más pequeños y mejores tiempos de ejecución.

Este lenguaje tiene dos modos de compilación, JIT (Just in time) durante el desarrollo o en modo de pruebas, y AOT (Ahead of time) utilizado cuando se compila una aplicación preparada para ser lanzada a un entorno de producción. El modo JIT cuenta con la capacidad de compilar incrementalmente, permitiendo lo conocido como ”Hot Reload”. ”Hot reload” o ”recarga en caliente” es una característica del compilador JIT de Dart, que permite aplicar cambios en el código fuente en menos de un segundo, sin necesidad de recompilar y ejecutar desde cero la aplicación. Esto por sí solo facilita mucho la fase del desarrollo y diseño de la interfaz de usuario y es otro de los grandes motivos de la elección de la tecnología. Por otro lado, el compilador AOT es capaz de obtener código máquina para las arquitecturas x64 y ARM, con un tiempo de lanzamiento inicial optimizado.

3.3 Kit de desarrollo de la interfaz de usuario

Se ha incluido esta sección ya que Flutter no deja de ser un conjunto de herramientas que permiten pintar en un ”Canvas”. Esto lo consigue mediante dos fases claramente diferenciadas, disposicionamiento o ”layout” y renderizado. La fase de ”layout” se encarga de medir y colocar los elementos de la interfaz en pantalla utilizando las restricciones indicadas por el usuario. Por ejemplo: El botón ”OK” debe de ir debajo del texto ”Hola mundo”. Para la fase de renderizado Flutter utiliza la librería de gráficos 2D Skia [8] con código nativo. A diferencia de otras tecnologías multiplataforma que recurren a la plataforma nativa para el renderizado, el código compilado de Dart utiliza Skia de forma nativa, evitando así la carga innecesaria que conlleva comunicarse con la plataforma.

Cabe destacar que Flutter es una tecnología de código abierto y muy orientada a la comunidad. Esto facilita en gran medida la exploración del ecosistema y muchas veces ayuda a entender ciertos aspectos de bajo nivel del framework. De lo contrario, con el código cerrado, uno depende mucho del mantenimiento de la tecnología por parte del equipo que haya detrás, con posibilidad nula de contribuir y añadir mejoras por parte de la comunidad.

Otra característica de la tecnología es la gran cantidad de herramientas de desarrollo oficiales que ofrece por defecto: ”debugger”, análisis de código y plugins para editores conocidos como VSCode [9] e IntelliJ IDEA [10]. Esta serie de herramientas se integran con la característica previamente mencionada de Dart, el ”Hot reload”.

3.4 Base de datos

La necesidad de una aplicación portable y con persistencia de datos determina en gran medida los tipos de base de datos que se pueden utilizar. Para esta aplicación se ha utilizado SQLite [11], una base de datos relacional muy ligera y portable que puede ser compilada para plataformas de escritorio y móvil. De todas formas, dado que uno de los requisitos es proteger los datos de acceso no autorizado, no es suficiente con utilizar SQLite, ya que todos los datos están en claro y son visibles si se utiliza cualquier lector de bases de datos SQLite. Por tanto se ha hecho uso además de una extensión para SQLite conocida como SQLCipher [12]. Esta extensión permite interactuar con la base de datos al igual que con SQLite, de forma totalmente transparente. La diferencia es que los datos están encriptados, y para poder abrir la base de datos es necesario utilizar una clave. Esta clave solo es conocida por la aplicación por lo que así se obtiene un acceso seguro a los datos.

3.5 Software de apoyo

3.5.1 Control de versiones

Para una correcta gestión y mantenimiento del proyecto, se ha utilizado el software de control de versiones Git [13]. Esta herramienta mantiene un historial de todos los cambios realizados en el proyecto de forma local, con posibilidad de volver atrás en cualquier momento. Este histórico local puede ser enviado a un servidor o repositorio remoto en el que otras personas pueden colaborar. Para este proyecto se ha utilizado un repositorio remoto de Gitlab. A pesar de que el desarrollo ha sido realizado por una sola persona, Git sigue siendo una herramienta indispensable, ya que permite mantener múltiples versiones del proyecto al mismo tiempo mediante ”ramas”. En este proyecto para cada funcionalidad con cierta complejidad se ha creado una rama independiente. Cuando se da como completada la funcionalidad, esta rama se junta con la rama principal que puede incluir otros cambios que no guarden relación con la misma.

3.5.2 Entorno de desarrollo

Para el entorno de desarrollo no es necesaria la instalación de muchas herramientas. Para comenzar con el desarrollo solamente es necesario un editor de texto que soporte los plugins oficiales de Flutter, aunque esto no es estrictamente necesario, podría utilizarse cualquier editor de texto. Para este proyecto se ha seleccionado Visual Studio Code, un editor de código abierto desarrollado por Microsoft con integración para diversidad de lenguajes y frameworks.

Para la compilación de una aplicación de escritorio con Flutter, además de instalar el framework desde la página oficial, es necesario instalar software adicional en función de la plataforma objetivo. En el caso de Linux, la plataforma principal utilizada en el desarrollo, se necesita instalar el compilador de C y C++ ”clang” junto con las librerías de desarrollo GTK (”libgtk-3-dev” en distribuciones Linux basadas en Debian). Por otro lado, para la compilación de un ejecutable Win32 para la plataforma Windows, es suficiente con instalar Visual Studio 2019 que incluye los componentes para la compilación de código C y C++. Por último, para compilar en macOS, es necesario instalar XCode, una herramienta utilizada para desarrollar aplicaciones nativas en los sistemas operativos de Apple, macOS (escritorio) e iOS (móvil). A día de hoy Flutter no cuenta con la capacidad de realizar compilación cruzada o ”cross-compiling”, por lo que para obtener un ejecutable Windows es necesario realizar la compilación desde la propia plataforma. Lo mismo ocurre para las otras dos plataformas.

3.5.3 Planificación

Para facilitar la fase de planificación y priorización de objetivos y tareas se ha utilizado la herramienta web Notion [14]. Esta aplicación dispone de tableros o pilas donde gestionar las tareas del proyecto. Existen otras herramientas similares y muy utilizadas, como Trello, pero se escogió Notion porque el equipo de desarrollo ya tenía experiencia habiéndolo utilizado anteriormente. Esta herramienta puede adaptarse muy bien a distintas metodologías de trabajo dada la diversidad de plantillas de las que dispone, pudiendo utilizarse tanto para la gestión de proyectos software como para la toma de notas o recordatorios individuales. Por último, para la creación del diagrama de Gantt de la planificación inicial y el seguimiento de las desviaciones en las tareas se ha utilizado la herramienta de código abierto GanttProject [15].

3.5.4 Diagramación y mockups

Para la realización de los ”mockups” de la interfaz y diversos diagramas se ha utilizado la herramienta web conocida como Diagrams.net [16]. Es una herramienta de diseño gráfico gratuita y de código libre que ya había sido usada anteriormente por el equipo de desarrollo; un factor importante en su elección para evitar la formación y fase de aprendizaje en otras herramientas alternativas no libres y mucho más complejas. Cuenta con un sistema de plugins el cuál facilita la adaptación a distintos tipos de diagramas. Por ejemplo, elementos para realizar el diagrama Entidad Relación de una base de datos o elementos de interfaz de usuario como barras de búsqueda, diálogos o botones que facilitan la fase de diseño y maquetado de la aplicación.

Capítulo 4

Metodología

En este capítulo se describe la metodología que se ha usado en el proyecto.

4.1 Desarrollo ágil

A lo largo de los años las formas en las que se organiza un equipo para el desarrollo un proyecto software ha ido evolucionando de forma constante. A día de hoy una rama de estos métodos es conocida como ”Agile” o ágil. Su principal característica es mejorar la comunicación con el usuario final o cliente durante todo el proceso del desarrollo, a través del uso de ciclos de desarrollo cortos, lanzamientos utilizables continuos y una mentalidad de refactorización constante. Como resumen, facilitar el cambio frente a imprevistos o nuevas necesidades a lo largo del ciclo de vida del proyecto.

Estas características surgen como respuesta a los principales problemas que muchas veces traen consigo las metodologías tradicionales. Éstas le dan mucha importancia a la planificación del proyecto en su totalidad desde el primer momento. Esto suele ser muy difícil de estimar y para hacerse de forma correcta se requiere mucha experiencia en el dominio del proyecto y en proyectos similares. Muchas veces hay necesidades del cliente que surgen a la mitad del desarrollo, y una planificación de todo el software pueden dificultar en gran medida la adaptación a estos nuevos requisitos.

4.2 Scrum

Para este proyecto se ha utilizado la metodología de trabajo ágil conocida como Scrum. Es un framework cuyo principal objetivo es facilitar el trabajo en equipo. En sus orígenes fue diseñado para la coordinación de proyectos software pero también ha sido utilizado en otro tipo de contextos como el de la investigación, marketing o ventas.

Scrum parte de la base de que un equipo puede no estar familiarizado en el dominio del

proyecto al inicio del mismo y que existirá una evolución constante de este conocimiento. Para adaptarse a este entorno en que las necesidades varían, Scrum utiliza ciclos de lanzamientos cortos, ayudando así al equipo de desarrollo a aprender y a mejorar. Estos ciclos de lanzamiento son conocidos como Sprints, cada uno con un objetivo claro a alcanzar. Cada Sprint además se complementa con una reunión con el usuario final para obtener feedback y descubrir nuevas necesidades que puedan surgir. Nuevos requisitos van seguidos de una fase de refactorización, pero esto es normal y no se ve como una desventaja de la metodología, sino como una ventaja dada la capacidad para adaptarse fácilmente ante cambios imprevistos. Además, ciclos cortos corresponden normalmente con una fase de cambio también corta, la cuál no afecta gravemente a la planificación del proyecto.

Junto con los Sprints, otra característica de esta metodología es la utilización de dos pilas de elementos diferentes, la pila del producto (Product backlog) y la pila del Sprint (Sprint backlog). En la pila del producto es donde se listan todas las historias de usuario del proyecto ordenadas por prioridad. A lo largo del desarrollo, cualquier cambio adicional (corrección de errores o nueva historia de usuario) es plasmado como un nuevo item en esta pila. Por otro lado, en la pila del sprint se encuentran las tareas a realizar en un sprint concreto para alcanzar un producto lanzable, conocido también como incremento.

4.3 Adaptación de la metodología

A pesar de que la metodología Scrum suele utilizarse en equipos de más de una persona, involucrando distintos roles como el de ”Product Owner” o ”Scrum master”, para este proyecto se ha realizado una adaptación que se adecúe a un equipo de un solo desarrollador. En la metodología Scrum, cuando hay varias personas en el equipo de desarrollo, es común realizar reuniones periódicas (una o dos semanas) donde se discuten las tareas del Sprint backlog en las que se ha trabajado. Estas reuniones tienen como objetivo repriorizar tareas del sprint que hayan sufrido imprevistos o estén llevando más tiempo del estimado. Para este proyecto en concreto se han eliminado estas reuniones, y en su lugar se ha hecho uso de la herramienta web Notion para tener una visión general de las tareas a realizar en cada sprint. A pesar de esto, las reuniones con LHICA se han mantenido al final de cada sprint para tener un feedback inmediato y realizar los cambios necesarios lo antes posible. En la figura 4.1 se muestra la organización por tareas y sprints con esta herramienta.

4.4 Toma de requisitos

La toma de requisitos del proyecto ha sido realizada de forma remota con el equipo de LHICA. Las principal cuestión planteada en la reunión inicial ha sido la dificultad del proceso

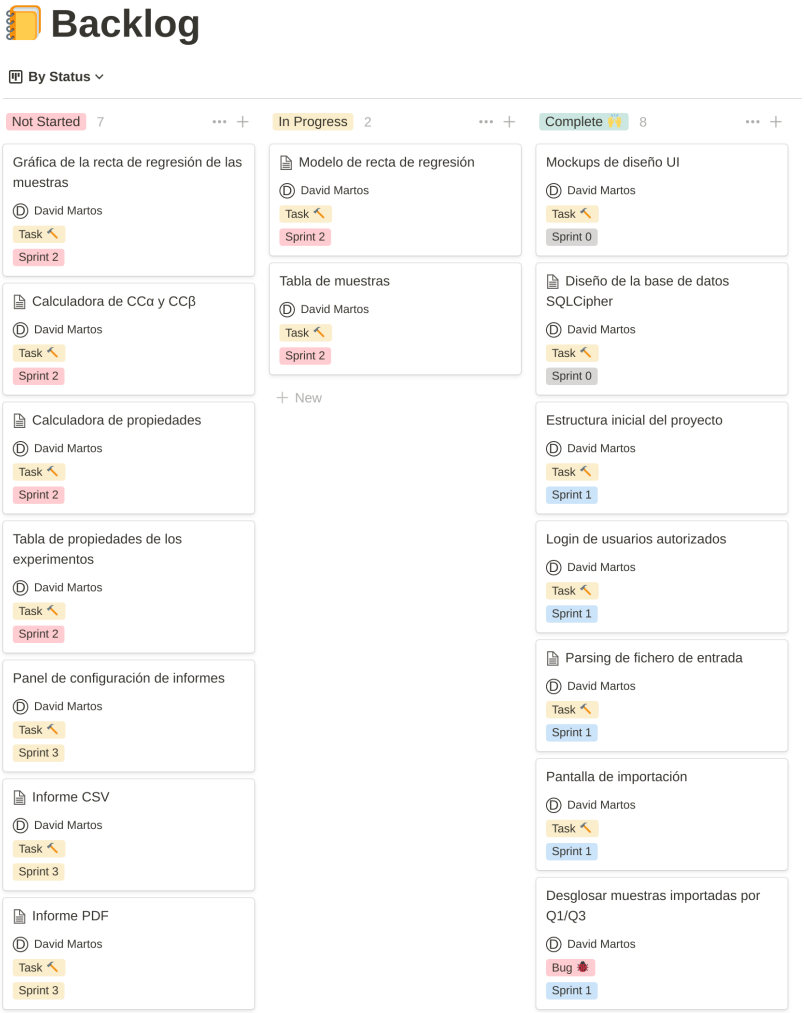


Figura 4.1: Backlog de tareas con Notion

de validación con las herramientas actuales. Además se recalcó el esfuerzo necesario para la realización de la tarea, llegando a dedicarle varios días en la entrada de datos y al análisis de resultados con estas herramientas basadas en hojas de cálculo. Otra de las necesidades del equipo es que la herramienta debería de ser lo más portable posible, sin instalaciones complejas, descartando aplicaciones web en las que es necesario tener un servidor funcionando en todo momento. Por último, la necesidad de exportar los resultados de las muestras que se persistan en la aplicación, permitiendo combinar los datos con otras herramientas de análisis más especializadas o compartir los resultados con otros laboratorios.

4.5 Historias de usuario

Tras la reunión inicial y la toma de requisitos, estas han sido las historias de usuario extraídas.

- HU-1: Como Usuario quiero que los datos de la aplicación estén protegidos de personas no autorizadas para evitar la manipulación de los mismos.
- HU-2: Como Usuario quiero poder obtener el CCα y CCβ asociado a una serie de medidas de concentración de un analito usando distintos métodos para poder comparar los resultados entre ellos y compartirlos con otros laboratorios.
- HU-3: Como Usuario quiero poder obtener la recuperación, la repetibilidad, la reproducibilidad y la incertidumbre asociados a una serie de medidas de concentración de un analito para poder determinar cuán de correctas y fiables son.
- HU-4: Como Usuario quiero poder obtener los cálculos correspondientes a una serie de experimentos con un límite máximo residual (LMR) o sin un límite determinado.
- HU-5: Como Usuario quiero poder visualizar la regresión de las medidas para cada día de validación de un analito para ver que siguen una proporción lineal.
- HU-6: Como Usuario quiero tener un histórico de los proyectos realizados para que me facilite la organización y mantenimiento de los resultados.
- HU-7: Como Usuario quiero una forma de añadir y persistir las medidas de los días de validación de un analito en un proyecto para poder acceder a ellos en cualquier momento.
- HU-8: Como Usuario quiero poder extraer la información obtenida por la aplicación en forma de documento para poder compartirlo con otros laboratorios o para utilizar estos datos en otros procesos.

- HU-9: Como Usuario quiero que la aplicación sea portable y ligera para poder utilizarla en un ordenador de un laboratorio o en un ordenador personal sin necesidad de realizar ninguna instalación.

4.6 Planificación inicial y costes

Este proyecto ha sido dividido en tres sprints.

1. **Inicio e importación de datos.**

   El objetivo es tener una aplicación donde se puedan importar las muestras a partir de ficheros obtenidos por el equipo de medición usado por LHICA. Aquí se trabajará HU-1 (Restricción de acceso), HU-6 (Organización en proyectos) con la pantalla inicial de proyectos y HU-7 (Inserción y persistencia de las muestras a partir del importador automático).

2. **Cálculo de resultados y visualización.**

   Tras este sprint se espera poder tener implementada la obtención de los resultados del análisis así como las rectas de regresión para las muestras. Por ello se trabajará en varias historias de usuario relacionadas: HU-2 y HU-3 para el cálculo de resultados asociados a las muestras importadas, HU-4 (Configuración de los resultados en base a un LMR o sin límite) y HU-5 (Recta de regresión de las muestras).

3. **Exportación de resultados.**

   Ampliar la funcionalidad de la aplicación con capacidades de generación de informes de resultados configurables por el usuario. Aquí se desarrollará HU-8 (Generación de informes), específicamente informes en PDF y CSV de los resultados obtenidos por la aplicación, tanto para un analito en concreto como para un proyecto completo.

Tras la finalización del proyecto, este ha sido el esfuerzo invertido en horas/persona (h/p).

- Captura de requisitos y análisis inicial: 10 h/p
- Elección de las tecnologías: En la decisión de las tecnologías y herramientas a utilizar frente a sus alternativas se han dedicado 10 h/p
- Diseños y mockups iniciales: 15 h/p
- Estructura inicial de la aplicación con base de datos funcionalidad de usuarios autorizados: Para la primera versión inicial de la aplicación ejecutable en las tres plataformas de escritorio se han dedicado 10 h/p

- Análisis del fichero de entrada e importación de muestras: 60 h/p
- Pantalla de información de analito y gráficas/tablas interactivas: En la pantalla principal donde se muestra toda la información y gráficas relevantes al proceso de validación se han invertido 90 h/p
- Exportación de resultados: Se invirtieron 40 h/p en las tareas de generación de informes con los datos de la aplicación.
- Estilado de la aplicación: En tareas relacionadas con el estilo y diseños de la interfaz de usuario se dedicaron 10 h/p
- Reuniones con LHICA: En la reuniones de seguimiento con el equipo se han invertido 10 h/p
- Pruebas de software: 20 h/p
- Mantenimiento de software actualizado y cambios en el framework: A lo largo del desarrollo del proyecto han surgido cambios en el framework de Flutter y en Dart que requirieron actualizar algunas partes del software. En total se han invertido 6 h/p
- Memoria Proyecto: 40 h/p

En la figura 4.2 se muestra un diagrama de Gantt con la planificación inicial de las tareas y sprints (en rojo) y la planificación resultante tras la finalización del proyecto (en azul). Las diferencias frente a las estimaciones iniciales se discutirán en el capítulo de conclusiones. Dado que el tamaño del equipo de desarrollo es de una persona, no ha sido posible paralelizar tareas, por lo que en diagrama puede observarse la linealidad en la realización de las mismas, dando lugar a desviaciones no recuperables en el camino crítico del proyecto.

El total de esfuerzo ha sido de 321 horas/persona. Considerando que el esfuerzo que se le ha podido dedicar al proyecto diariamente ha sido de 2 horas, la duración del proyecto ha sido aproximadamente 4 veces mayor a una jornada completa de 8 horas dedicadas al mismo. Este factor de la duración, el tamaño reducido del equipo de desarrollo y los bajos costes necesarios para la realización del proyecto, permiten presupuestarlo con 20€ por hora, algo más bajo que la media del coste por hora en proyectos software a medida (30€/h). Con esta cifra el coste total del proyecto resultaría en 6420€.

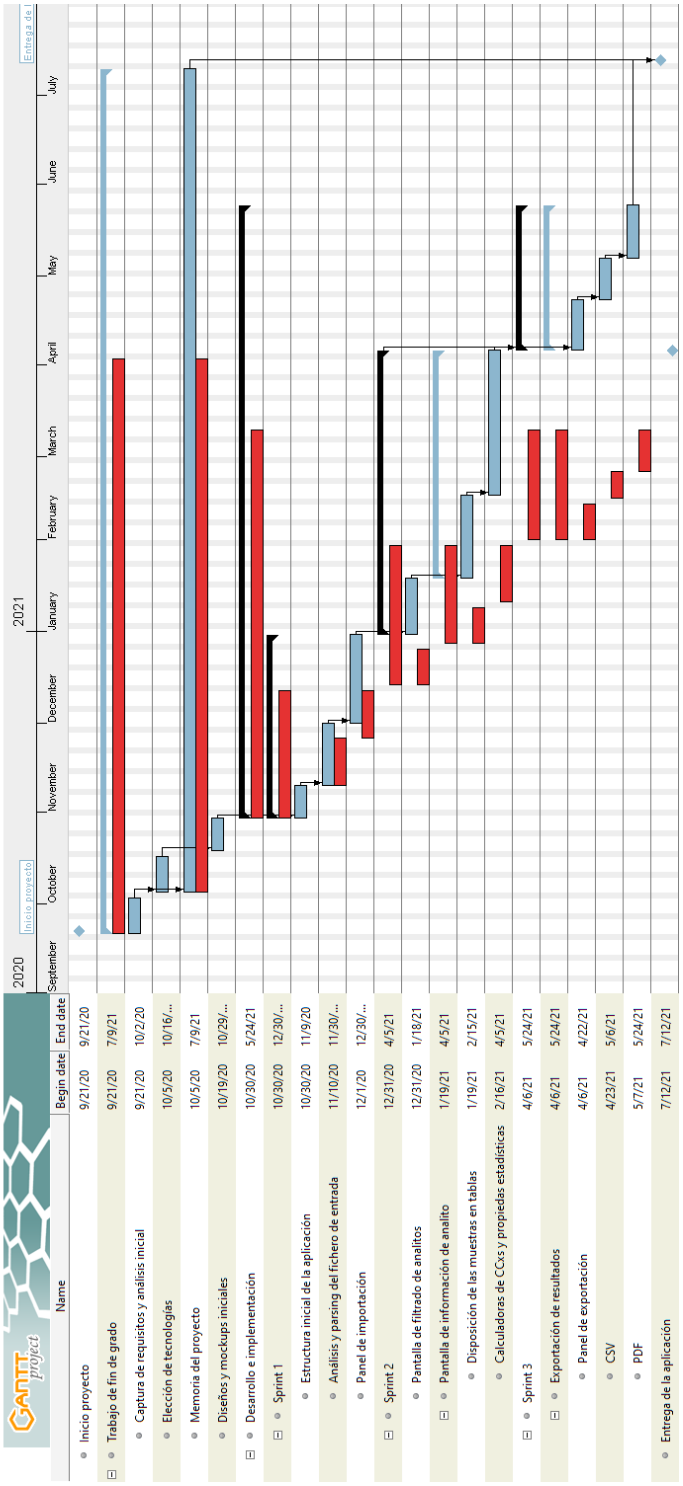


Figura 4.2: Diagrama de Gantt

Capítulo 5

Diseño y desarrollo de la aplicación

5.1 Diseño

5.1.1 Modelo de datos

En el modelado de datos con los que trata la base de datos de la aplicación se definen las siguientes entidades.

En primer lugar están los llamados ”Compuestos”. En esta entidad es donde se modela cada compuesto utilizado en cada una de las muestras. Un ejemplo podría ser el del antibiótico monensina, comúnmente utilizado en la alimentación animal.

En segundo lugar está la entidad ”Analito” que asocia un compuesto a una serie de muestras con un objetivo en común a efectos de organización. En esta entidad se describen propiedades relacionadas con el experimento como pueden ser el nivel de validación utilizado, las unidades de medida utilizadas en las concentraciones, el proyecto al que pertenece el experimento, el compuesto por el que está formado y las masas Q1/Q3, que actúan de descriptor sobre el compuesto del analito (un mismo compuesto puede tener masas Q1/Q3 distintas, por lo que es necesario distinguir la utilizada).

En tercer lugar se modela la entidad ”Muestra”, donde se incluye toda la información obtenida por el espectrómetro de masas, de entre las que se destacan el analito utilizado en la muestra, la fecha de toma, la concentración utilizada y el valor de concentración medido por el espectrómetro, también conocido como ”Área pico”.

Por último, y previamente mencionado está la entidad ”Proyecto”, cuyo objetivo es organizar múltiples analitos en un mismo lugar, y facilitar así la exportación de datos relacionados.

El diagrama entidad relación puede encontrarse en la figura 5.1

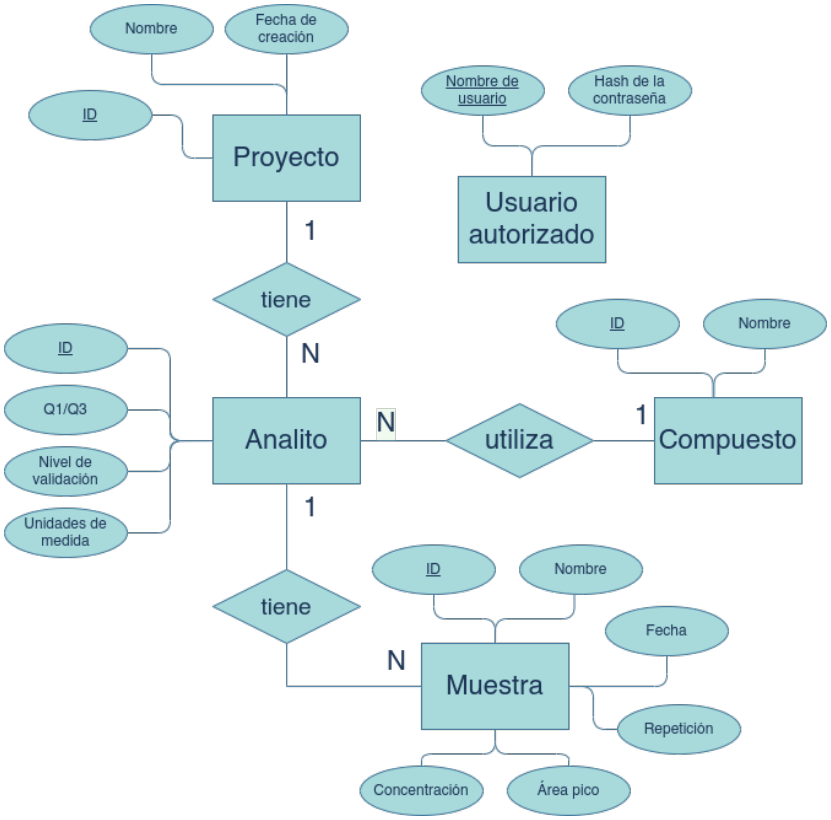


Figura 5.1: Diagrama Entidad-Relación de la base de datos

5.1.2 Interfaz de usuario

Para el diseño de la interfaz se ha decidido utilizar el lenguaje de diseño Material Design [17], recientemente renovado y adaptado para pantallas grandes de escritorio. En sus orígenes (2014) fue específicamente diseñado para interfaces móviles en Android, pero a día de hoy cuenta con componentes adaptados a escritorio, como pueden las barras de ”scroll” o menús de navegación. Además, Flutter cuenta con una gran de componentes Material incluidos por defecto por lo que facilita mucho el prototipado e implementación de la interfaz de usuario usando esta guía de diseño.

Uno de los objetivos principales del diseño de la interfaz de usuario es reducir la información mostrada en pantalla a la esencial en comparación a las hojas de cálculo ya mencionadas previamente, las cuales contienen cálculos intermedios de los resultados finales, que rara vez necesitan ser consultados y esconden los datos de mayor relevancia.

El diseño general y disposición de los elementos principales de la interfaz escogido se basa en un ”Navigation Rail” [18] para organizar en pestañas las funcionalidades esenciales de la aplicación. Éstas son la importación de datos y el acceso directo al histórico de resultados almacenados en la base de datos de la aplicación, organizado por proyectos. En la figura 5.2 se muestra esta estructura inicial.

Importar

En la sección de importación es donde se permite que la aplicación abra y analice el fichero generado por el espectrómetro de masas de LHICA para asociar sus datos a un proyecto existente o nuevo. Una vez seleccionados un proyecto y un fichero, la aplicación muestra todas las muestras encontradas en el fichero, agrupadas a su vez por analito. Si el usuario lo desea, puede desmarcar analitos que puede que estén en el fichero, pero que no le interesen en ese proyecto.

Proyectos y analitos

La vista de proyectos a su vez organiza todos los analitos analizados e incluye además una barra de búsqueda para encontrar fácilmente un analito por nombre. Véase la figura 5.3. En la vista de detalle de un analito se visualizan las muestras realizadas organizadas por día en forma de tabla. A nivel global de todos los días se muestran los resultados finales calculados por la aplicación, previamente mencionados en las historias de usuario. Dado que el CCα y CCβ pueden calcularse usando distintos métodos, en esta versión de la aplicación se incluyen dos de ellos que utilizará el laboratorio interesado. No obstante, este diseño es extendible a otros métodos de cálculo. Para cada día, además de la tabla con las muestras, un factor importante es la visualización de la proporcionalidad que siguen las muestras a distintas concentraciones.

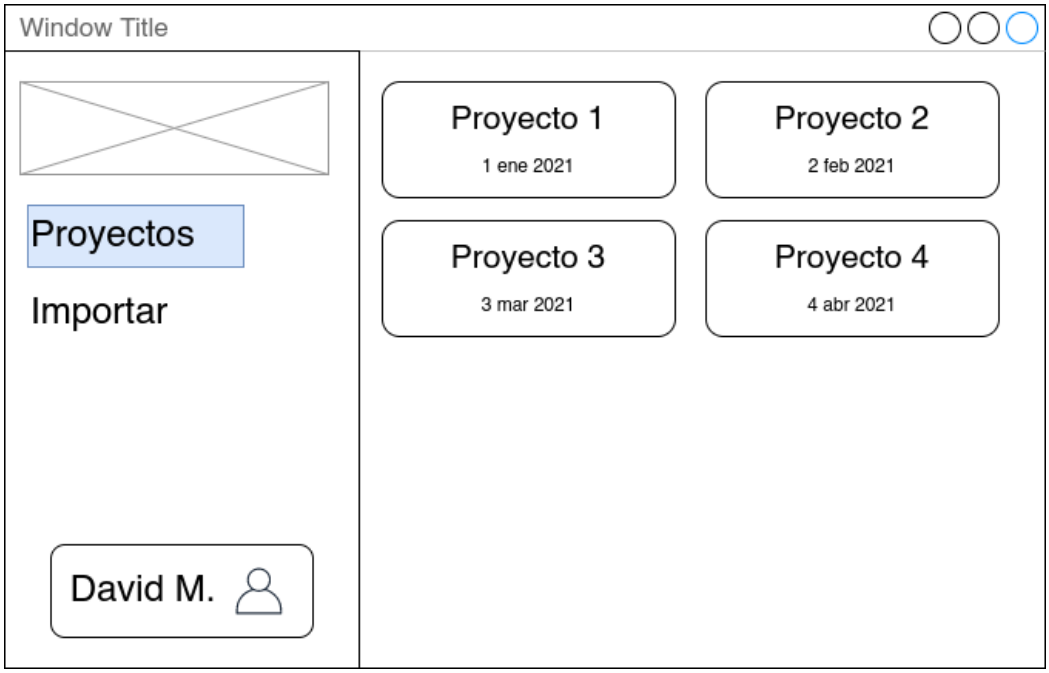


Figura 5.2: Mockup inicial de la vista de proyectos

Para ello, se incluye una recta de regresión lineal, al igual que hacen las hojas de cálculo originales. Además se incluyen los mismos resultados que se muestran de forma global, pero solo aplicados a un día en particular.

Otro aspecto importante a destacar de esta vista es que todos los resultados finales mostrados están vinculados a las muestras de la tabla, que pueden ser deshabilitadas de forma individual permitiendo así la examinación de puntos problemáticos, subconjuntos de las muestras, etc..., todo ello sin tener que modificar los datos originales. Una ventaja frente al sistema utilizado actualmente con las hojas de cálculo. Estos cambios actualizan tanto los resultados como las rectas de regresión en tiempo real. En la figura 5.4 se muestra un mockup representativo de todo lo anterior.

Exportar

Tanto desde la vista de detalle de un analito como desde la vista de un proyecto es posible exportar los CCα y CCβ obtenidos por la aplicación en base a distinto parámetros escogidos por el usuario. Estos parámetros se componen por: el formato del fichero a generar (PDF o CSV); el método de cálculo de CCα y CCβ; y los límites que se desean utilizar (nivel de validación, sin límite o límites propios).

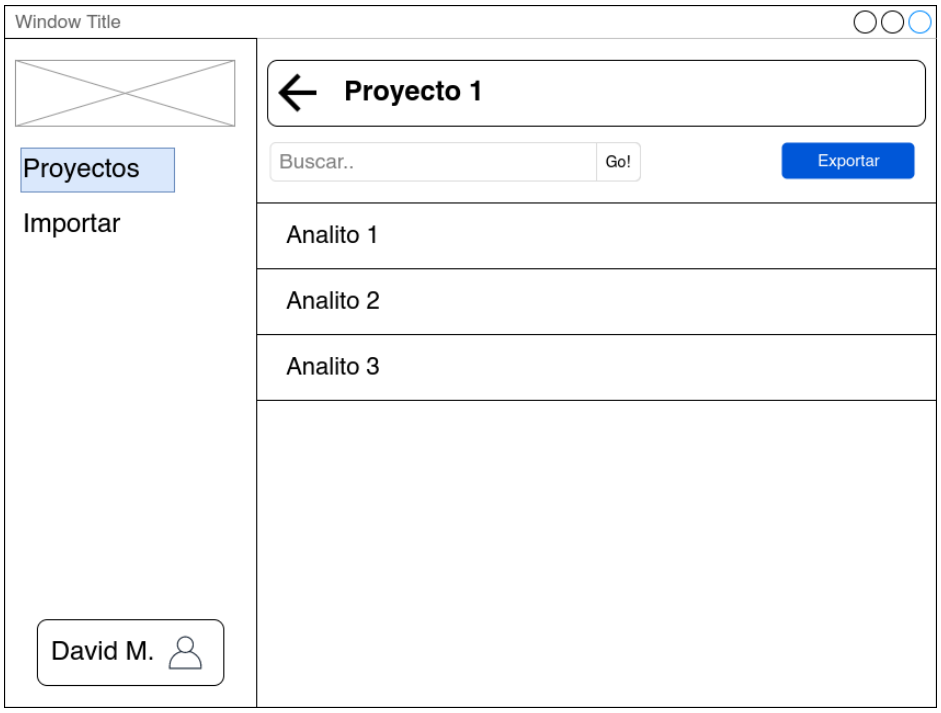


Figura 5.3: Mockup inicial de la vista de analitos

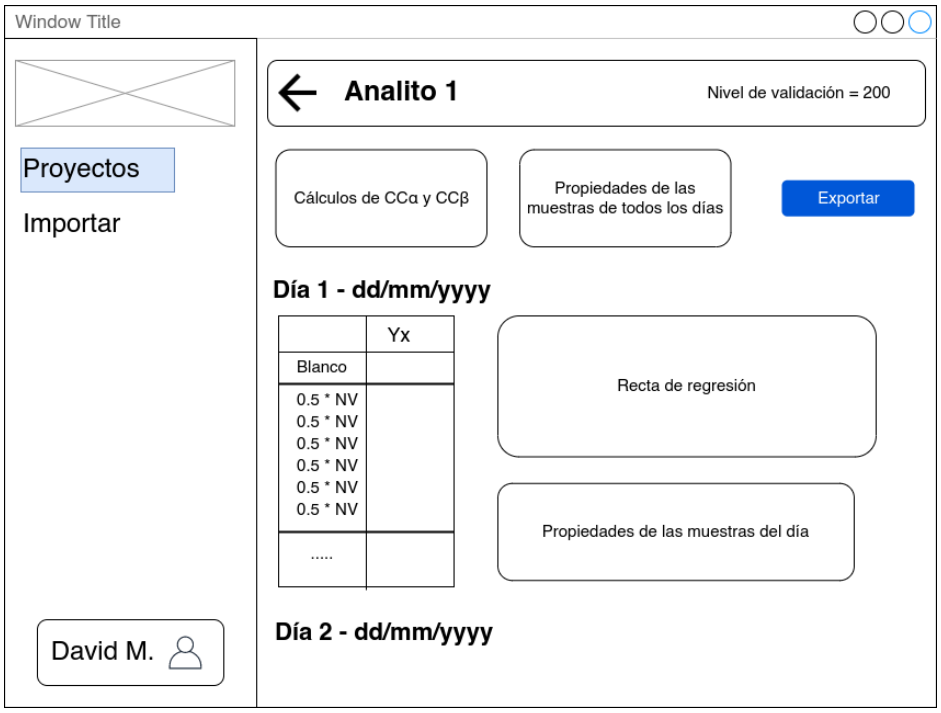


Figura 5.4: Mockup inicial de la vista de detalle de un analito

5.2 Sprint 1: Inicio e importación de datos

En este sprint se ha creado la estructura inicial del proyecto, con la pantalla de inicio, así como el mecanismo de inyección de dependencias utilizado, que facilitará la implementación y organización de funcionalidades en los siguientes sprints. Por último este sprint ha finalizado con la implementación del importador automático de muestras a partir del fichero generado por el espectrómetro de masas.

En primer lugar, para estructurar el proyecto ha sido necesario decidir el mecanismo de comunicación y envío de datos entre las distintas partes de la aplicación para que reaccionen a eventos que realice el usuario. A esto se le suele conocer como ”gestión del estado” o ”state management”. En las aplicaciones Flutter no existe ningún state management oficial ya que cada proyecto puede tener necesidades diferentes. Para este se ha utilizado la librería de Dart ”provider” [19], que proporciona una manera elegante de compartir información entre los distintos árboles de widgets/componentes de la aplicación. Esta librería se ha usado para dos grandes partes, la inyección de dependencias y la observabilidad de los datos.

La inyección de dependencias consiste en declarar aquellos elementos de los que diferentes módulos de la aplicación dependen, normalmente a través de interfaces y no implementaciones concretas. Por ejemplo, los DAOs de la base de datos son utilizados en diferentes pantallas, y se desea utilizar la misma instancia para todas. Para ello se declara un ”Provider” o ”proveedor” de la interfaz del DAO en una parte del árbol de widgets para que pueda ser utilizado en cualquiera de los descendientes. Esto facilita la creación de pruebas de software, ya que es posible inyectar un DAO ”mock” en vez de una implementación que se conecte a una base de datos real.

Para la observabilidad de los datos, se ha hecho uso de los conocidos como ”notificadores” o ”notifiers”, que son clases que guardan el estado de la interfaz y permiten emitir los cambios que hayan ocurrido en este estado para que la interfaz reaccione y se repinte con información actualizada. Este concepto de clases también suele conocerse como clases ”View Model”. Estos notifiers hacen uso del patrón de software ”Observador”, para poder notificar de cambios a múltiples partes de la aplicación (observers) al mismo tiempo. Estas clases ViewModel han sido utilizadas en todas las pantallas de la aplicación para almacenar y manipular toda la información no trivial de la interfaz de usuario, como por ejemplo las muestras cargadas de base de datos, o el estado en el que se encuentran varios ”checkboxes” de un listado.

Una vez estructurado y decidida la organización del proyecto software, se ha decidido comenzar por la tarea de autorización y acceso a la aplicación. Para esta tarea se ha implementado un mecanismo de inicio de sesión por usuario y contraseña. A pesar de estar utilizando una base de datos encriptada como la que proporciona SQLCipher, no significa que la contraseña guardada en claro sea una buena práctica. Por ello se ha utilizado adicionalmente un

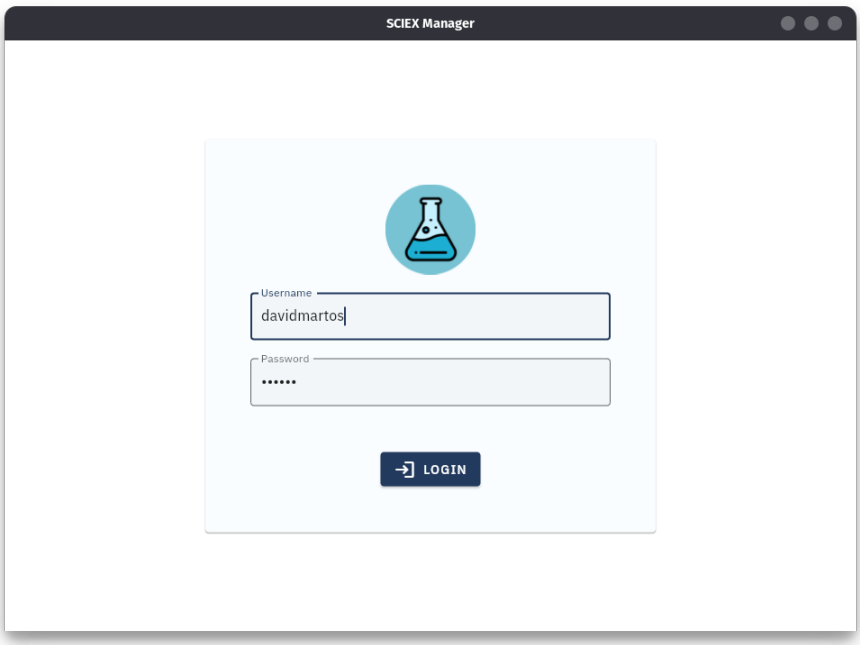
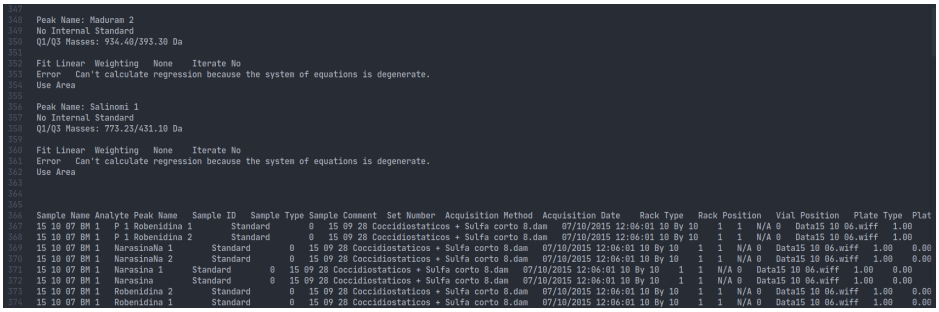


Figura 5.5: Ventana de login

algoritmo de hashing para la contraseña del usuario, en concreto ”PBKDF2”. De esta manera se persiste un hash de la contraseña en base de datos y a la hora de comprobar si un intento de contraseña es válido, se aplica la misma función de hash al intento para a continuación realizar la consulta a base de datos.

Flutter además cuenta con widgets que facilitan el trabajo con formularios como el de usuario y contraseña. Es posible configurarlos para que en casos de error como el de una contraseña mal introducida se indique correctamente un mensaje de error debajo de la entrada de texto. A cada elemento del formulario se le configura una función de validación, en el caso del campo de contraseña es la verificación de que el usuario y contraseña se encuentran en la base de datos. Esta función puede llamarse en cualquier momento y la reactividad del framework navegará automáticamente a la pantalla inicial o por lo contrario mostrará el mensaje de error correspondiente.

La siguiente tarea importante ha sido la de introducir todos los datos resultantes del fichero obtenido a partir del espectrómetro de masas en la base de datos de la aplicación. El fichero generado, a pesar de no estar marcado explícitamente con una extensión de texto (”.txt” o ”.csv”), internamente está codificado en UTF-8. Algunas partes del fichero se asemejan al formato CSV, a diferencia de otras con información necesaria para la importación, como las masas Q1/Q3. Es por ello que fue necesario desarrollar un ”parser” propio para la extracción de todos los datos. Otra peculiaridad del fichero es que todas las muestras aparecen juntas,



```
Peak Name: Maduram 2
No Internal Standard
Q1/Q3 Masses: 934.40/393.30 Da

Fit Linear  Weighting   None    Iterate No
Error   Can't calculate regression because the system of equations is degenerate.
Use Area

Peak Name: Salinomi 1
No Internal Standard
Q1/Q3 Masses: 773.23/431.10 Da

Fit Linear  Weighting   None    Iterate No
Error   Can't calculate regression because the system of equations is degenerate.
Use Area



Sample Name Analyte Peak Name   Sample ID   Sample Type Sample Comment  Set Number  Acquisition Method  Acquisition Date    Rack Type   Rack Position   Vial Position   Plate Type  Plat
15 10 07 BM 1   P 1 Robenidina 1        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00
15 10 07 BM 1   P 1 Robenidina 2        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00
15 10 07 BM 1   NarasinaNa 1        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00    0.00
15 10 07 BM 1   NarasinaNa 2        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00    0.00
15 10 07 BM 1   Narasina 1      Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00    0.00
15 10 07 BM 1   Narasina        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00    0.00
15 10 07 BM 1   Robenidina 2        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00    0.00
15 10 07 BM 1   Robenidina 1        Standard        0   15 09 28 Coccidiostaticos + Sulfa corto 8.dam   07/10/2015 12:06:01 10 By 10    1   1   N/A 0   Data15 10 06.wiff   1.00    0.00
```

Figura 5.6: Ejemplo del fichero a importar

pero para facilitar la elección y visualización de los datos antes de la importación en la interfaz se ha hecho que el ”parser” devuelva las muestras agrupadas por compuesto, las masas Q1/Q3 y la fecha de la muestra. En la figura 5.6 se muestra una de las entradas del fichero autogenerado.

Una vez los datos fueron modelados y extraídos del fichero se procedió a la implementación de la interfaz de importación. Por una parte, el menú para seleccionar un proyecto sobre el cuál guardar las muestras y para seleccionar el fichero a importar. Este proyecto puede ser uno ya existente en la aplicación o puede ser uno nuevo, con la posibilidad de crearlo desde este mismo flujo de importación. Por otra parte, una pantalla donde visualizar las muestras agrupadas por analito una vez extraídas del fichero. Para la visualización de los datos del fichero se ha optado por una disposición ”master-detail”, donde de forma general se muestran los analitos y si se selecciona uno de ellos, pueden verse las muestras asociadas al mismo en un panel lateral.

Ya con las muestras, analitos y proyectos persistidos en la base de datos, el siguiente objetivo fue la navegación ente estas distintas entidades desde la aplicación. Para esto se desarrollaron las dos pantallas necesarias, una para mostrar los proyectos (accesible directamente desde una de las pestañas principales de la aplicación) y otra donde visualizar los analitos de un proyecto en concreto. Ambas de estas tareas han sido desarrolladas de forma muy similar, dada su semejanza en la comunicación con su DAO correspondiente y en la disposición de los elementos en la interfaz. Cabe destacar que la implementación de estos dos listados (proyectos y analitos) hacen uso de una característica de la librería de comunicación con la base de datos relacional que permite reaccionar a cambios en tablas modificadas para realizar de nuevo las consultas. Esto se combina muy bien con el paradigma reactivo y declarativo de Flutter. Simplemente es necesario declarar la consulta a base de datos como ”Stream” y la librería automáticamente se encargará de recargar la información cuando alguna de las tablas que afecten a la consulta haya sido modificada, ya sea mediante un INSERT, un UPDATE o un DELETE. De esta forma, importar un fichero nuevo en la aplicación recargará automáticamente los listados de proyectos y analitos al terminar.

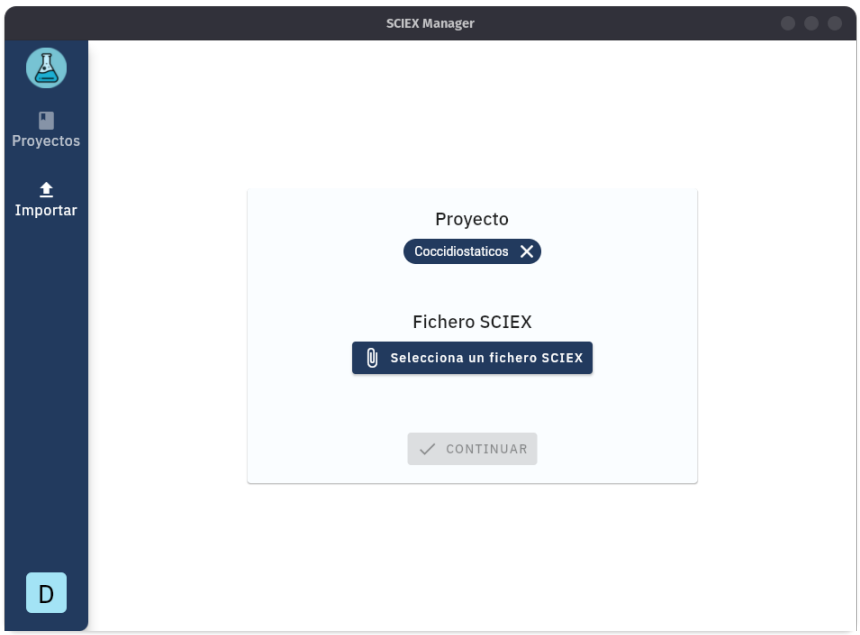


Figura 5.7: Configurador de la importación

5.2.1 Revisión del sprint

Tras tener esta base inicial de la aplicación, en una reunión con LHICA con objetivo de discutir las capacidades del importador, surgió la necesidad de agrupar los analitos de la interfaz a su vez por las masas Q1/Q3. Como ya se ha mencionado anteriormente, un compuesto puede aparecer varias veces con distintas masas Q1/Q3, por tanto se procedió a aplicar los cambios oportunos para esta nueva necesidad. Finalmente el menú de importación y pantallas de proyectos y analitos quedaron como se muestra en las figuras 5.7, 5.8 y 5.9.

5.3 Sprint 2: Cálculo de resultados y visualización

En esta fase del desarrollo se ha trabajado principalmente en la pantalla de detalle de un analito seleccionados en el listado. Aquí existen tres objetivos principales: mostrar los datos de muestras agrupadas por día de validación, mostrar la recta de regresión para cada uno de los días, y obtener los resultados finales al igual que hacen las hojas de cálculo utilizadas a día de hoy.

Para visualizar las muestras se ha optado por utilizar una tabla, donde para las columnas se utiliza un subconjunto de todas las características que nos proporciona el fichero de importación. De todas formas, los datos no incluidos en la tabla están igualmente persistidos en

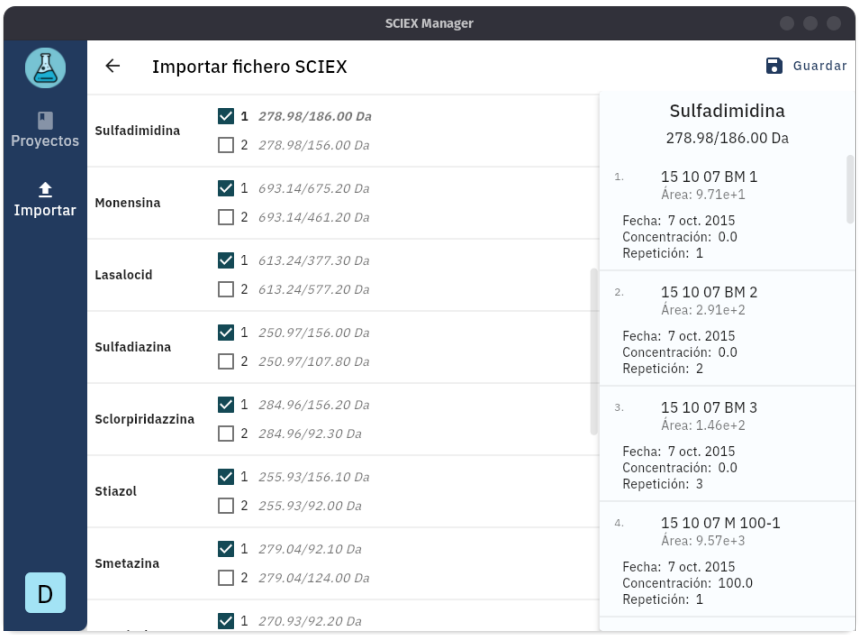


Figura 5.8: Vista de la importación de analitos y muestras

la base de datos por si en un futuro es necesario realizar una funcionalidad en la que sean relevantes. Además, para facilitar la lectura de la tabla, conociendo el dominio de los datos, se ha implementado la tabla de tal manera que las muestras para cada concentración estén visiblemente agrupadas utilizando líneas de separación más gruesas de lo normal.

La implementación de la gráfica con la recta de regresión no tuvo ningún contratiempo. Con todos los pares Concentración-Área es posible calcular la recta que mejor se aproxime a todos los puntos. En el código se ha modularizado la recta de regresión de tal forma que pueda ser reutilizada para el cálculo de los resultados finales del experimento, con funcionalidad adicional para consultar el valor aproximado en cualquiera de los dos ejes a partir del eje opuesto. Un ejemplo de este caso utilizado en la aplicación es la llamada Concentración Calculada o CC, que aproxima la concentración correspondiente a un Área arbitraria de medición. En un mundo perfecto, la concentración calculada coincidiría con la concentración utilizada en la muestra. En la figura 5.10 se muestra la gráfica para uno de los días de validación.

5.3.1 Cálculo de resultados

Por último, la implementación de los resultados finales puede dividirse en dos partes. Por un lado el cálculo del CCα y CCβ (con sus varias formas de cálculo), y por otro el cálculo de propiedades de análisis sobre las muestras tomadas (recuperación, repetibilidad, reproduci-

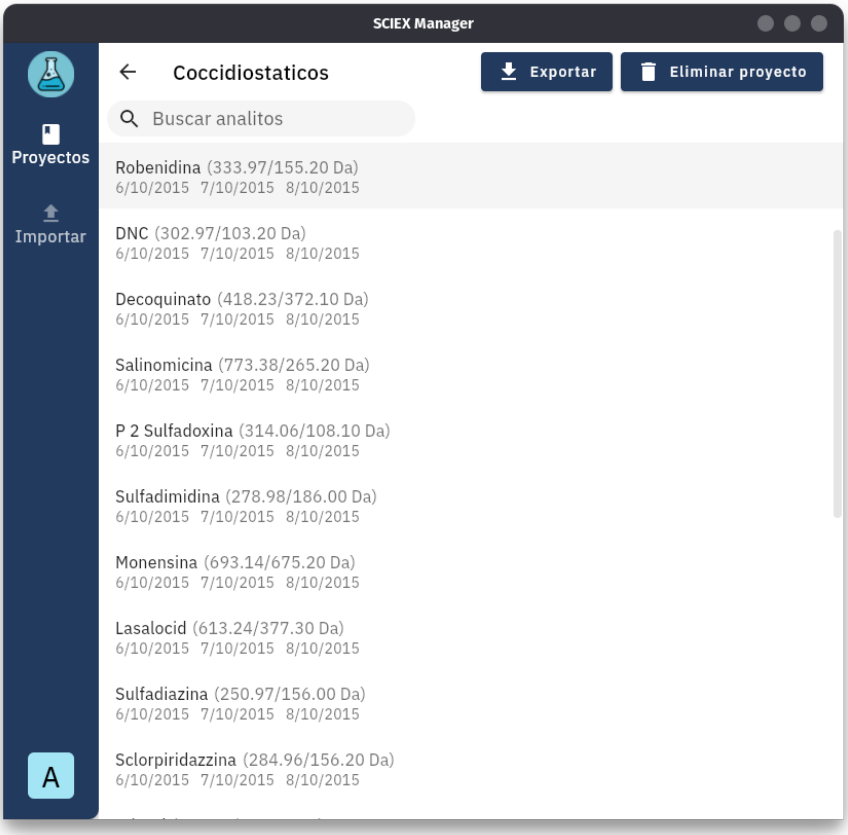


Figura 5.9: Vista de la lista de analitos

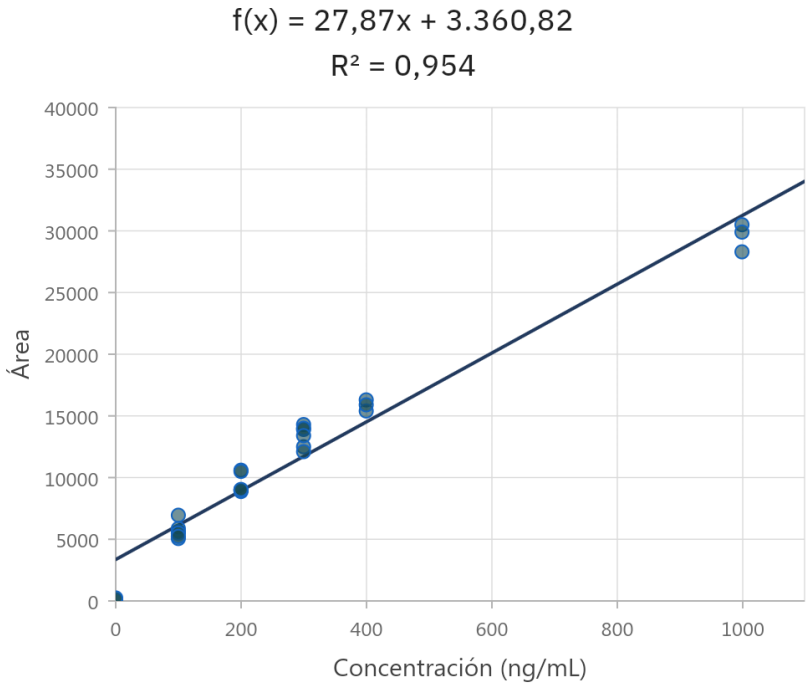


Figura 5.10: Recta de regresión de las muestras

bilidad e incertidumbre). Para ambas partes hubo que dedicar tiempo para entender el flujo de los datos y cálculos intermedios que las hojas de cálculo utilizan con objeto de extraer las fórmulas para reimplementarlas en la aplicación.

Como ya se ha mencionado, el cálculo de los CCα y CCβ varía en función del método utilizado por lo que es importante diseñar esta parte pensando en extensibilidad futura. Para ello se ha utilizado un patrón estrategia para delegar el cálculo a implementaciones concretas. El patrón estrategia es un patrón de diseño software comúnmente utilizado que permite encapsular distintos algoritmos bajo una misma interfaz, permitiendo escoger el algoritmo a ejecutar de forma dinámica y totalmente transparente al resto del sistema. En esta situación, cada método de cálculo de los CCα y CCβ es una estrategia diferente. Para esta versión de la aplicación se han creado dos implementaciones de este cálculo, que son las utilizadas por LHICA, pero en el futuro este catálogo de métodos puede ampliarse sin requerir cambios fundamentales en el diseño de la aplicación. La interfaz utilizada para describir este cálculo se muestra en la figura 5.11. Nótese que Dart no dispone de una distinción directa entre clases abstractas e interfaces pero sí de la diferencia entre herencia e implementación de interfaz.

Es necesario realizar la distinción entre un cálculo en el que existe un límite máximo de concentración y en el que no existe un límite, ya que la utilidad de un dato u otro va a depender del contexto con el que se esté trabajando. ”AnalyteAggregatedData” es una abstracción sobre

```
abstract class CCCalculator {

  CCAlphaAndBeta withLimit(double limit, AnalyteAggregatedData
    aggregatedData);

  CCAlphaAndBeta withoutLimit(AnalyteAggregatedData aggregatedData);
}
```

Figura 5.11: Interfaz de CCCalculator

las muestras utilizadas en el cálculo, pudiendo así utilizar la implementación para obtener los CCs de un día concreto o de todos los días al mismo tiempo.

El cálculo del resto de propiedades fue bastante directo, puesto que solo existe una manera de obtenerlos. De todas formas, puede destacarse que para esta serie de cálculos una de las características del lenguaje Dart ha ayudado a mejorar la expresividad de algunas fórmulas. Se trata de los ”Extension methods” o métodos de extensión, que permiten añadir funcionalidad a una librería que no está al alcance para ser modificada. Un ejemplo en este contexto son conocidas expresiones matemáticas como el cálculo de la media o desviación típica. La librería standard de Dart no implementa esta capacidad, pero con los ”Extension methods” es fácil extender esta funcionalidad sobre tipos conocidos, como el de una lista de números. En la figura 5.12 se demuestra con un ejemplo de uso de estas extensiones para el cálculo de la reproducibilidad.

5.3.2 Revisión del sprint

Tras otra reunión con el equipo de LHICA para demostrar las nuevas capacidades de la aplicación, surgió una nueva necesidad. En ocasiones es útil tener la posibilidad de descartar ciertas muestras de la ecuación porque se sabe que son erróneas o para buscar puntos clave. En base a este nuevo conocimiento se desarrolló una funcionalidad para hacer más dinámica la pantalla de detalles. Para ello todas las tablas de muestras tienen la capacidad de seleccionar sus elementos, pudiendo desmarcar algunos para visualizar cómo afecta a los resultados. A raíz de esta función de selección se ha añadido también la posibilidad de eliminar de forma permanente todas las muestras desmarcadas para que no vuelvan a aparecer. En las figuras 5.13 y 5.14 se muestra el resultado obtenido.

5.4 Sprint 3: Exportación de datos

Por último, el desarrollo de la funcionalidad de generación del informe CCα y CCβ (HU-8) puede dividirse en el panel de configuración de la exportación, en el módulo de generación de CSV y en el de generación de PDF.

```dart
double calcReproducibility(double ratio) {
    final List<double> ccs = ccMap[ratio];
    final avg = ccs.avg();
    final std = ccs.stdDeviation();
    return (std * 100) / avg;
}

extension NumListExtension<N extends num> on Iterable<N> {
  num sum() {
    return fold(0, (acu, v) => acu + v);
  }

  double avg() {
    return sum() / length;
  }

  double stdDeviation() {
    final double avgVal = avg();
    final variance = map((n) => pow(n - avgVal, 2)).avg();
    return sqrt(variance);
  }
}
```

Figura 5.12: Extension methods en Dart

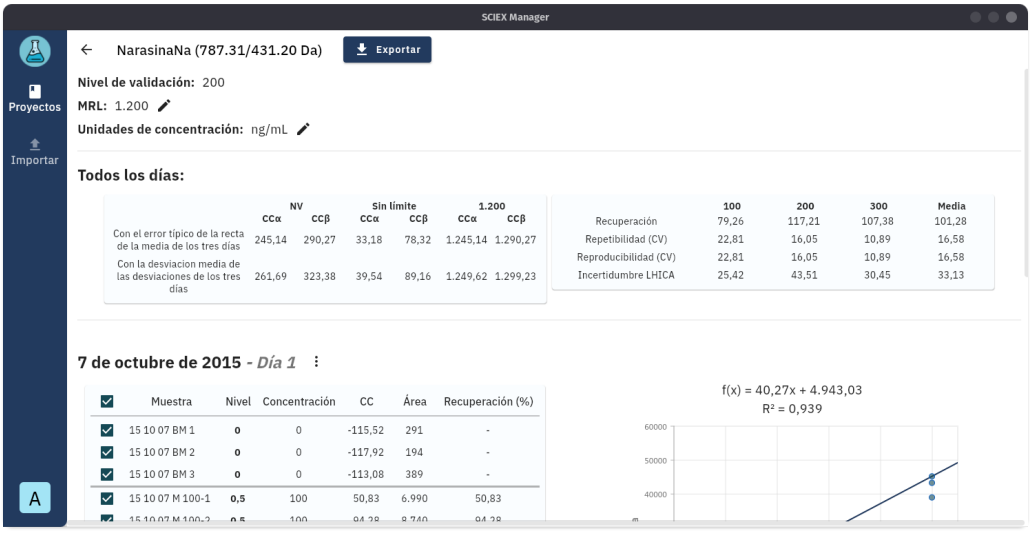


Figura 5.13: Resultados finales de la validación de un analito

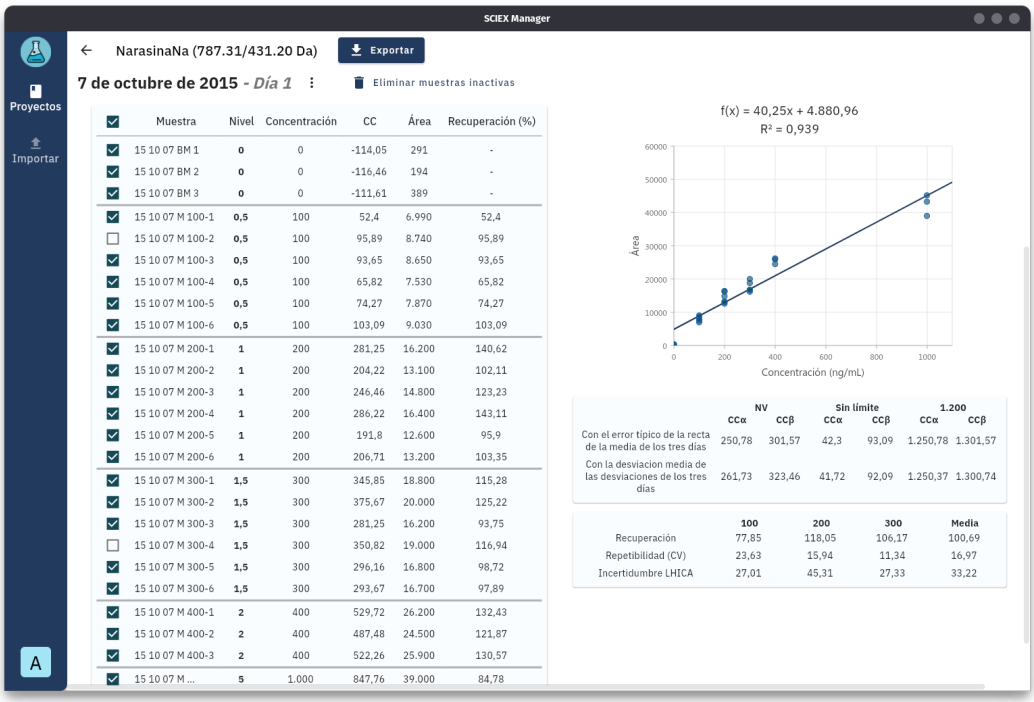


Figura 5.14: Tabla de muestras y recta de regresión

Esta capacidad de exportación se incluye tanto en la vista de detalle de un analito, utilizando sus muestras asociadas; como en la vista de detalle de un proyecto, para que se utilicen todas muestras de cada uno de los analitos del proyecto.

El panel de configuración permite seleccionar el formato de salida (PDF o CSV), el método de cálculo de los CC y los límite de residuo máximo a utilizar. Por defecto se incluyen los límites ”sin límite” y el nivel de validación del experimento, pero se añade la posibilidad de añadir múltiples límites concretos si el usuario así lo desea. Esto facilita la personalización del informe en caso de que vaya a ser compartido con otro laboratorio.

Finalmente, la generación de los ficheros de salida fue bastante directa gracias a la utilización de la previamente mencionada ”CCCalculator” con su implementación concreta en función del método de cálculo seleccionado por el usuario.

En las figuras 5.15 y 5.16 se muestra el la interfaz de configuración junto con un ejemplo de exportación a PDF.

5.4.1 Revisión del sprint

Tras este sprint final, en la reunión con el equipo, tras observar los avances y las funcionalidades de las que ya disponía la aplicación, se decidió comenzar a utilizarla en LHICA en

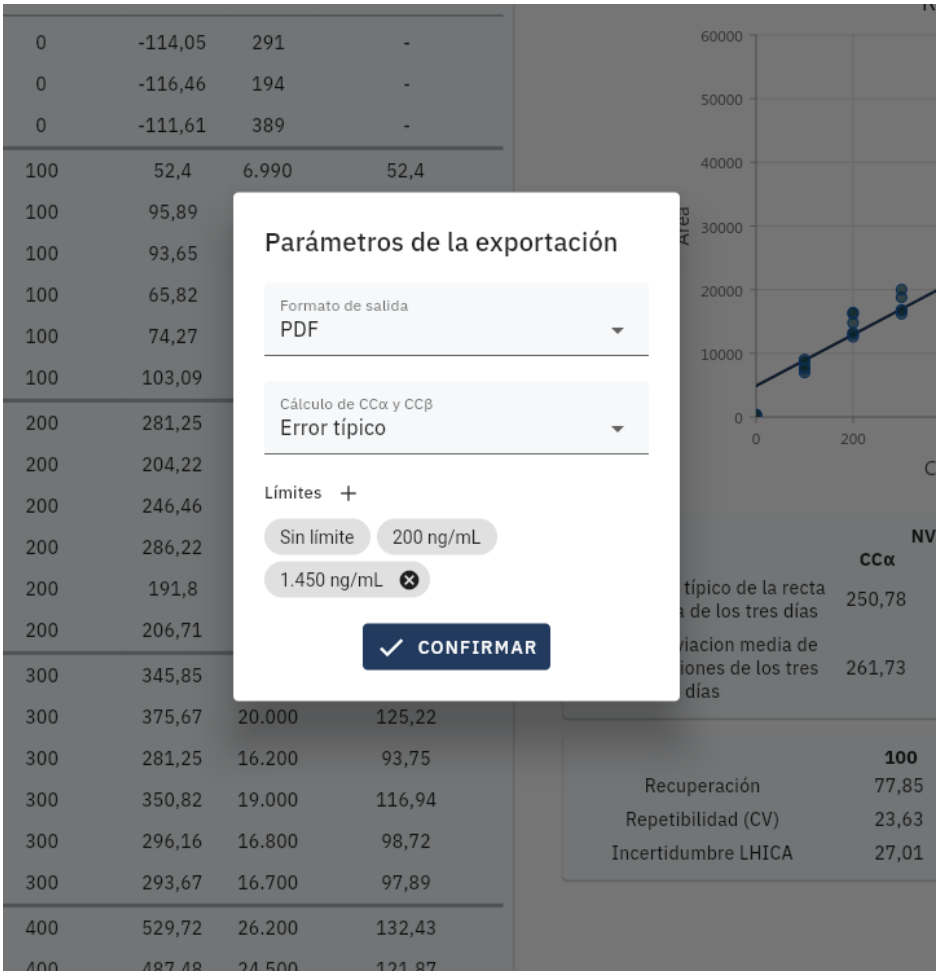


Figura 5.15: Configurador de la exportación

modo de pruebas. El principal objetivo con esta decisión es validar el funcionamiento de la aplicación en un entorno más real y así poder recibir feedback más concreto que pueda surgir a lo largo de los procesos de validación de sustancias en alimentos.

5.5 Localización

La aplicación ha sido localizada a tres idiomas: español, gallego e inglés. Existen varias formas de localizar una aplicación Flutter, y en general se dividen en herramientas que utilizan ARB o JSON para organizar las frases de la aplicación. En este proyecto se ha utilizado la herramienta ”intl_utils” [20], basada en el formato ARB, el cuál permite definir plurales, enumerados, así como parametrizar frases. ARB, un formato clave valor muy similar a JSON, es el formato oficial soportado por Flutter al localizar aplicaciones, pero para integrar las frases en el código Dart es necesario definir un gran número de variables y funciones que lo hacen una

Narasina (747.22/373.20 Da)

All days

Limit	CCα	CCβ
No limit	-23.33 ng/mL	2.82 ng/mL
200 ng/mL	226.15 ng/mL	252.29 ng/mL
1450 ng/mL	1476.15 ng/mL	1502.29 ng/mL

6/10/2015

Limit	CCα	CCβ
No limit	-9.49 ng/mL	28.76 ng/mL
200 ng/mL	238.26 ng/mL	276.52 ng/mL
1450 ng/mL	1488.26 ng/mL	1526.52 ng/mL

7/10/2015

Limit	CCα	CCβ
No limit	-51 ng/mL	-49.08 ng/mL
200 ng/mL	201.92 ng/mL	203.84 ng/mL
1450 ng/mL	1451.92 ng/mL	1453.84 ng/mL

8/10/2015

Limit	CCα	CCβ
No limit	-9.49 ng/mL	28.76 ng/mL

Figura 5.16: Exportación a PDF

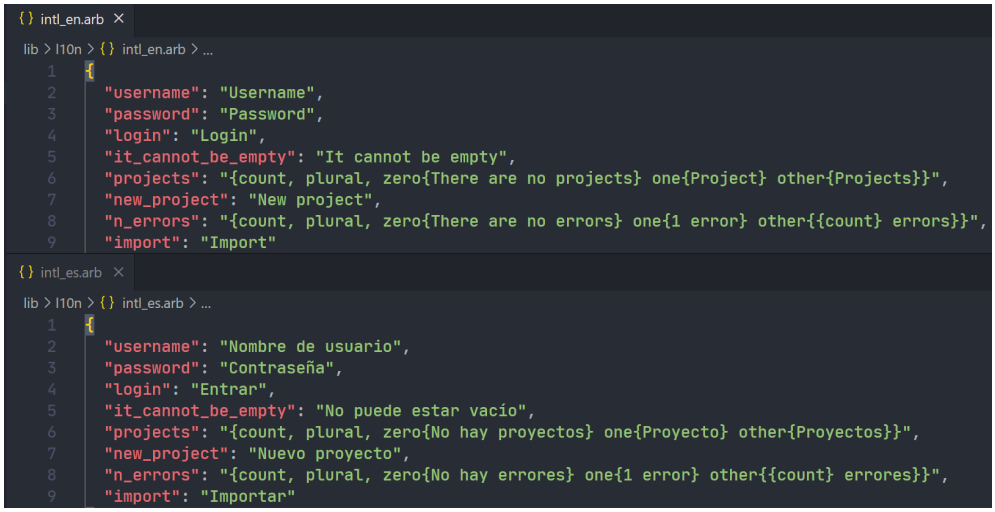


Figura 5.17: Ficheros ARB

tarea laboriosa. Por ello, ”intl_utils” utiliza generación de código para reducir este esfuerzo, proporcionando un mecanismo en el que solamente es necesario actualizar una entrada por cada frase en un archivo dedicado a cada idioma soportado. En la figura 5.17 se muestra un fragmento de los archivos ARB de español e inglés. El idioma utilizado por la aplicación viene determinado por el idioma en el que esté el sistema operativo. Además, el formato de las fechas o valores numéricos mostrados en la aplicación también se adecúa a la configuración definida en el sistema.

Capítulo 6
Pruebas

En este capítulo se describen las pruebas funcionales y no funcionales realizadas en el proyecto.

6.1 Pruebas unitarias

Las pruebas unitarias comprueban que el comportamiento o funcionamiento de una unidad de código sea el adecuado. En ocasiones, una unidad en el software depende de módulos o sistemas externos, como una base de datos o una API a la que conectarse. En estos casos, es común recurrir a una estrategia conocida como objetos simulados o ”mocking”. Con esta técnica se pueden crear y configurar objetos ”mock” que respondan con un valor concreto diseñado específicamente para la prueba que se esté realizando.

Para este proyecto se ha hecho mayor énfasis en pruebas unitarias para dos módulos en concreto, el análisis o ”parsing” del fichero de entrada y las distintas fórmulas y cálculos estadísticos involucrados en el proceso analítico. Respecto a la implementación de estas pruebas, Dart proporciona una librería de pruebas oficial integrada en el propio kit de desarrollo. Una característica de esta librería es la capacidad de agrupar pruebas relacionadas entre sí, permitiendo compartir configuraciones o ”mocks” que tengan en común, y así reducir la cantidad de código necesario, aumentando la legibilidad.

En la figura 6.1 se muestra un ejemplo de las pruebas de unidad implementadas en el proyecto para la validación de uno de los métodos de cálculo de CCα y CCβ.

```
void main() {
  final List<List<DataPoint>> samplesByDay = samplesMonensin;

  group('average std error calculator', () {
    final ccCalculator = AverageStdErrorCCCalc();
    const double validationLevel = 200;
    final aggregatedData = AnalyteAggregatedData(
        validationLevel,
        samplesByDay,
    );

    test('with limit 200', () {
      final ccAlphaBeta = ccCalculator.withLimit(200,
    aggregatedData);
      expect(ccAlphaBeta.ccAlpha, moreOrLessEquals(211.000383));
      expect(ccAlphaBeta.ccBeta, moreOrLessEquals(222.000767));
    });

    test('with limit 1250', () {
      final ccAlphaBeta = ccCalculator.withLimit(1250,
    aggregatedData);
      expect(ccAlphaBeta.ccAlpha, moreOrLessEquals(1261.000383));
      expect(ccAlphaBeta.ccBeta, moreOrLessEquals(1272.000767));
    });

    test('without limit', () {
      final ccAlphaBeta = ccCalculator.withoutLimit(aggregatedData);
      expect(ccAlphaBeta.ccAlpha, moreOrLessEquals(13.2814491));
      expect(ccAlphaBeta.ccBeta, moreOrLessEquals(24.2818327));
    });
  });
}
```

Figura 6.1: Prueba unitaria demostrativa

6.2 Pruebas de interfaz de usuario

Además de pruebas unitarias, Flutter proporciona herramientas adicionales para validar el comportamiento de los componentes de la interfaz de usuario. A grandes rasgos podría decirse que la interfaz de usuario es el resultado de aplicar una función (la lógica del programa) a un estado de la aplicación en un momento concreto. Con esta filosofía es posible realizar este tipo de pruebas mediante una serie de pasos secuenciales que simulen la interacción con la interfaz (entrada de texto, botones...) seguidos de aserciones que se aseguren que el estado en cada momento coincida con el esperado. En la figura 6.3 se muestra un ejemplo de este tipo de pruebas para la funcionalidad de filtrado por nombre en el listado de analitos.

Debe ser mencionado que este tipo de pruebas, a pesar de proporcionar mecanismos para interactuar con la interfaz, no requieren que la aplicación sea lanzada, por lo que los tiempos de

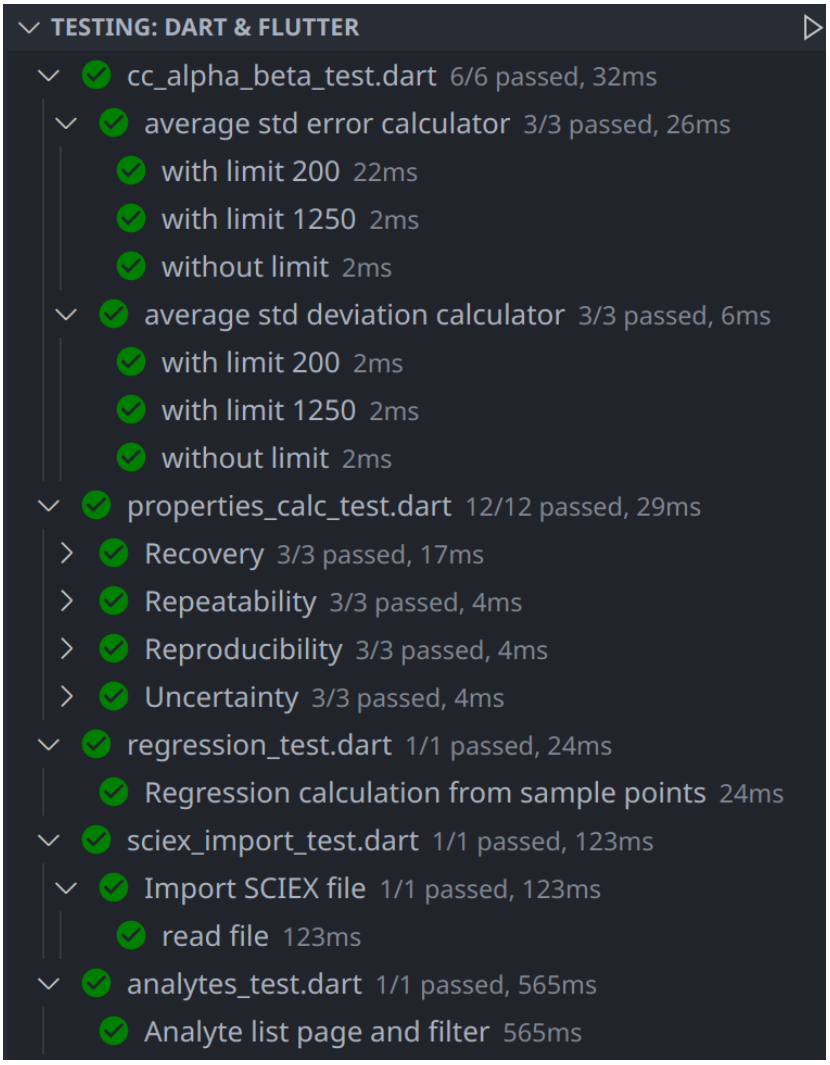


Figura 6.2: Resumen de pruebas unitarias y de interfaz de usuario

ejecución son mucho mejores. No obstante, si se necesita probar que el estilo de la aplicación sea fiel a los diseños originales, este tipo de pruebas no son adecuadas. En estos casos es preferible usar pruebas conocidas como ”golden” donde se compara la interfaz de usuario obtenida en la prueba con una imagen objetivo. Si estas imágenes difieren en un margen significativo, se determina que la prueba ha sido fallida.

En la figura 6.2 se muestra un resumen de las pruebas unitarias y de interfaz de usuario obtenido por el plugin de desarrollo Dart en Visual Studio Code, con varias agrupaciones en aquellas pruebas relacionadas entre sí, como se ha mencionado anteriormente.

```dart
void main() {
  final project = Project(
    id: 123,
    name: "Test project",
    creationDate: DateTime.now(),
  );

  final db = MockDatabase();
  final analytesDaoMock = MockAnalytesDao();
  when(() => db.analytesDao).thenReturn(analytesDaoMock);

  testWidgets('Analyte list page and filter', (tester) async {
    when(() => analytesDaoMock
        .allAnalytes(projectId: project.id))
        .thenAnswer((_) async => [
            testAnalyte(name: "ABC"),
            testAnalyte(name: "XYZ"),
          ],
    );

    await tester.pumpWidget(
        widgetWrapper(AnalytesListPage(project)),
    );

    // Initial loading
    final finder = find.byType(CircularProgressIndicator);
    expect(finder, findsOneWidget);

    // Show all analytes
    await tester.pump();
    // 2 results in the list are expected
    expect(find.byType(ListTile), findsNWidgets(2));

    // Filter analytes by compound name
    await tester.enterText(find.byType(TextField), "ab");
    await tester.pump();

    // Only one analyte expected
    expect(find.byType(ListTile), findsOneWidget);
    expect(find.text("ABC"), findsOneWidget);
  });
}
```

Figura 6.3: Prueba de interfaz de usuario demostrativa

6.3 Pruebas no funcionales

El objetivo de las pruebas no funcionales es asegurarse que características del software no directamente relacionadas con la funcionalidad del sistema (rendimiento, escalabilidad o fiabilidad) son correctas. Respecto al rendimiento de la aplicación, un punto crítico es la base de datos. Para integrar una base de datos como SQLCipher en la aplicación existen dos técnicas, el uso de canales de comunicación con la plataforma nativa y el uso de FFI (Foreign function interface), que permite ejecutar código nativo en C/C++ desde Dart. Para este proyecto se ha optado por usar FFI ya que no requiere reimplementar la comunicación con la librería nativa de SQLCipher para cada una de las tres plataformas, aumentando así la mantenibilidad y escalabilidad. Una desventaja de esta aproximación es que por defecto las operaciones a base de datos se realizarán en el hilo principal de la aplicación, pudiendo llegar a observar pérdida de fotogramas por segundo en operaciones complejas. Es por esto que adicionalmente se ha hecho uso de un hilo dedicado para la ejecución de consultas a base de datos.

Para mantener una aplicación lo más escalable posible, en todo momento se han utilizado patrones de software conocidos en aquellos módulos con alta probabilidad de ser extendidos en el futuro, como la inclusión de nuevos métodos de cálculo de CCα y CCβ al proyecto. Además se ha incluido la posibilidad de importar muestras tomadas en un número indeterminado de días sin sacrificar la interfaz de usuario. Por lo que es posible generalizar los cálculos a N días si es necesario, a diferencia de las herramientas basadas en hojas de cálculo.

Capítulo 7

Conclusiones

En este capítulo se presentan las conclusiones del proyecto, su puesta en producción y el posible trabajo futuro.

7.1 Objetivos alcanzados

El principal objetivo que este trabajo de fin de grado trataba de conseguir era la simplificación de la tarea de análisis de residuos y contaminantes en alimentos, un proceso muy costoso a día de hoy por el esfuerzo que trae la inserción de cientos de muestras a mano en diversas hojas de cálculo. Tras una última reunión con el equipo de LHICA y con la aplicación en su estado actual, la herramienta ha sido puesta en funcionamiento a modo de prueba en el ordenador de usuarios expertos. Y de acuerdo con los resultados tangibles de la fase de pruebas, la aplicación está ahorrando mucho tiempo en este proceso. Tras un tiempo de evaluación es posible que se encuentren necesidades más concretas que puedan ampliar la funcionalidad de la aplicación como trabajo futuro.

7.2 Seguimiento de la planificación

Como ya ha sido mencionado, al final de cada sprint se ha realizado una reunión con el equipo de LHICA para demostrar el funcionamiento que ha ido adquiriendo la aplicación a lo largo del desarrollo. Esto ha sido esencial para la captura de nuevas necesidades, ya que es mucho más sencillo sugerir ideas y nuevas funcionalidades cuando ya existe una base que cuando no hay nada. Los cambios más grandes realizados y que han desviado en parte la planificación y diseños iniciales han sido dos, en el primer y segundo sprint. Esto puede observarse en el diagrama de Gantt de la figura 4.2 presentado al inicio de esta memoria.

Estas desviaciones surgen principalmente por el desconocimiento del dominio por parte del equipo de desarrollo, por lo que se ha requerido más tiempo del esperado comprender en

detalle cada uno de los datos resultantes en el campo químico y estadístico. Esto fue observado en el primer sprint para poder analizar el fichero del espectrómetro de masas de forma correcta, y sobretodo, en el desarrollo del segundo sprint, ya que este requirió implementar varias fórmulas a partir de varias hojas de cálculo.

Además se ha subestimado el esfuerzo necesario para diseñar la interfaz de usuario de las pantallas principales que trabajan con mucha información al mismo tiempo. Por un lado, la pantalla de importación, que requiere integrar cientos de muestras de laboratorio sin complicar la interfaz de usuario. Por otro lado, la pantalla de detalles de un analito para el segundo sprint, que necesita un diseño óptimo para la representación de todos los resultados de los cálculos en una sola pantalla.

7.3 Lecciones aprendidas

El uso de Flutter como framework de interfaz de usuario durante el desarrollo ha facilitado la fase de diseño y prototipado de la aplicación gracias a su arquitectura basada en widgets/-componentes reutilizables. Al igual que el desarrollo para móvil en base a la experiencia anterior del equipo de desarrollo, utilizar Flutter para una aplicación de escritorio multiplataforma también ha resultado exitosamente. De nuevo, un factor muy importante para la observación de estos resultados es el sistema de Hot Reload que el framework ofrece, que a la larga ahorra mucho tiempo de compilación y facilita el desarrollo cuando se están realizan cambios en el código, pudiendo ver el resultado al instante.

La utilización de Flutter en estado beta para el desarrollo de aplicaciones de escritorio no ha provocado ningún inconveniente en el desarrollo, dado que el tipo de aplicación desarrollada no necesita una integración profunda con el sistema operativo (notificaciones del sistema, acceso al micrófono o webcam…). En cualquier caso, si este tipo de integración es necesaria en el futuro, Flutter proporciona mecanismos para generar plugins nativos que utilizan código propio de cada plataforma para comunicarse con el código Dart.

Haber utilizado Dart como lenguaje también ha resultado exitoso. El conocimiento del lenguaje no ha supuesto ningún inconveniente y ha llegado a aumentar la expresividad de algunas partes del código de la aplicación. Si se necesitara realizar otra aplicación de escritorio multiplataforma y las condiciones y requisitos lo permiten, en base a la experiencia con este trabajo, el equipo volvería a escoger Flutter y Dart para el desarrollo. Una desventaja que podría destacarse de la utilización de Flutter es que es un framework relativamente nuevo, con su primera versión estable lanzada el 4 de diciembre de 2018, ofreciendo inicialmente solo la posibilidad de crear aplicaciones móviles multiplataforma. No fue hasta 2020 cuando se lanzó la capacidad para desarrollar aplicaciones de escritorio. Este factor puede repercutir a la hora de encontrar un equipo desarrollo con experiencia a diferencia de otras tecnologías

más estandarizadas que utilizan Javascript o Typescript (React, Angular, Vue...).

La elección de la tecnología de base de datos, SQLCipher (esencialmente SQLite), tampoco ha supuesto ningún problema y ha resultado efectiva, sin ninguna observación negativa en cuanto al rendimiento. La portabilidad que ofrece también ha simplificado el proceso de implementación en los ordenadores del laboratorio, no requiriendo ningún tipo de instalación de servidor externo o dependencias adicionales.

7.4 Trabajo futuro

Como complemento al trabajo realizado, existen una serie de funcionalidades que podrían llevarse a cabo para abrir la aplicación a otros campos.

En primer lugar, la utilización de Flutter ofrece la posibilidad de generar una aplicación para dispositivos portátiles como tablets o teléfonos móvil. Esta podría ser una funcionalidad interesante sobretodo para la visualización y análisis de resultados desde cualquier lugar. Esta posible funcionalidad requeriría ciertos cambios en la arquitectura del proyecto, ya que para compartir las muestras persistidas en la aplicación de escritorio sería necesario desarrollar un sistema de sincronización junto con un servidor al cuál acceder desde el dispositivo móvil.

Otra posible rama de trabajo es ampliar la capacidad del sistema actual para poder importar muestras que estén en otro formato, ya que para este proyecto se ha utilizado el fichero autogenerado por los equipos usados en LHICA. Esto podría abrir las puertas a otros laboratorios que estén interesados en los resultados finales de los experimentos obtenidos por la aplicación y así validar sus propios métodos de análisis.

Por último, una funcionalidad adicional que podría desarrollarse sería darle capacidad colaborativa a la aplicación. Esto es, hacer posible la comunicación de resultados a otros usuarios desde la propia aplicación. Uno de los principales usos de los resultados finales es poder compartirlos con diferentes laboratorios para contrastar los distintos métodos de medición utilizados. Al igual que en la primera funcionalidad adicional sugerida, este cambio también necesitaría combinarse con un servidor adicional a través del cuál compartir y sincronizar los datos de la aplicación.

Apéndices

Apéndice A

Material adicional

A.1 Instrucciones de ejecución y compilación de producción

Para ejecutar la aplicación en modo desarrollo es necesario instalar la versión 2.2.3 de Flutter, que incluye la versión 2.13 de Dart.

A.1.1 Ejecución

- Obtener las dependencias del proyecto

  `flutter pub get`

- Ejecución de la aplicación en modo desarrollo

  `flutter run -d {plataforma}`, donde la plataforma es ”linux”, ”windows” o ”macos”.

A.1.2 Compilación

Para compilar el proyecto es necesario disponer de las herramientas mencionadas en 3.5.2.

`flutter build {tipo_salida}`, donde el tipo de salida en este caso coincide con los nombres de las plataformas: ”linux”, ”windows” y ”macos”, aunque una misma plataforma puede soportar varios tipo de compilación, por ejemplo UWP (Universal Windows Platform). En estos momentos el equipo de Flutter está trabajando para dar soporte a compilar aplicaciones para el ecosistema UWP, pero para este proyecto no se ha estudiado esta capacidad dado que no es un formato pensado para aplicaciones portables.

A.1.3 Distribución

El empaquetado y distribución de la aplicación consiste en la generación y compilación de la aplicación con el apartado anterior y a continuación comprimir el resultado en un formato

conocido como ZIP, que pueda ser descomprimido desde la máquina del usuario donde éste prefiera (En el disco duro, memoria USB…). Finalmente solo es necesario ejecutar la aplicación, sin necesidad de instalaciones.

Lista de acrónimos

CC *Concentración calculada.*

LMR *Límite máximo residual / Maximum Residue Limit.*

NV *Nivel de validación.*

Glosario

CCα Límite de decisión para confirmación: límite en el cual y a partir del cual se puede concluir con una probabilidad de error de α que una muestra no es conforme, y el valor 1 − α significa certeza estadística en porcentaje de que se ha superado el límite permitido.

CCβ Capacidad de detección del cribado: contenido mínimo de analito que puede ser detectado o cuantificado en una muestra con una probabilidad de error de β:

a) En el caso de sustancias farmacológicamente activas prohibidas o no autorizadas, la CCβ es la concentración más baja en la que un método es capaz de detectar o cuantificar, con una certeza estadística de 1 − β, muestras que contengan residuos de sustancias prohibidas o no autorizadas.

b) En el caso de sustancias autorizadas, la CCβ es la concentración en la que el método es capaz de detectar concentraciones por debajo del límite permitido con una certeza estadística de 1 − β.

Analito Sustancia que debe ser detectada, identificada y/o cuantificada y los derivados de la misma que se formen durante el análisis.

Espectrómetro de masas Dispositivo capaz de determinar la distribución de las moléculas de una sustancia en función de su masa. Se utiliza para identificar los diferentes elementos químicos que forman un compuesto.

Bibliografía

[1] "Reglamento (ce) nº 470/2009 del parlamento europeo y del consejo." [En línea]. Disponible en: https://www.boe.es/doue/2009/152/L00011-00022.pdf

[2] "Reglamento de ejecución (ue) 2021/808." [En línea]. Disponible en: https://www.boe.es/doue/2021/180/L00084-00109.pdf

[3] "Laboratorio de higiene, inspección y control de alimentos." [En línea]. Disponible en: https://lhica.org

[4] "Resval©." [En línea]. Disponible en: https://www.wur.nl/en/Research-Results/Research-Institutes/food-safety-research/Laboratory-Quality-Service-1/Reference-materials-in-webshop/Software-ResVal.htm

[5] "Flutter." [En línea]. Disponible en: https://flutter.dev

[6] "Manabox." [En línea]. Disponible en: https://manabox.app

[7] "Dart." [En línea]. Disponible en: https://dart.dev

[8] "Skia." [En línea]. Disponible en: https://skia.org/

[9] "Visual studio code." [En línea]. Disponible en: https://code.visualstudio.com/

[10] "Intellij idea." [En línea]. Disponible en: https://www.jetbrains.com/idea/

[11] "Sqlite." [En línea]. Disponible en: https://www.sqlite.org

[12] "Sqlcipher." [En línea]. Disponible en: https://www.zetetic.net/sqlcipher

[13] "Git." [En línea]. Disponible en: https://git-scm.com/

[14] "Notion." [En línea]. Disponible en: https://www.notion.so/product

[15] "Ganttproject." [En línea]. Disponible en: https://www.ganttproject.biz/

[16] "Diagrams.net." [En línea]. Disponible en: https://app.diagrams.net/

[17] "Material design." [En línea]. Disponible en: https://material.io/design

[18] "Navigation rail." [En línea]. Disponible en: https://material.io/components/navigation-rail

[19] "Provider." [En línea]. Disponible en: https://pub.dev/packages/provider

[20] "Intl utils." [En línea]. Disponible en: https://pub.dev/packages/intl_utils