ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE MÓDULOS EN APLICATIVOS MÓVILES DE SISTEMA ERP UTILIZANDO PRÁCTICAS DEVOPS SEGURAS PARA MANTICORE LABS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

NAULA LOMAS CHRISTIAN ALEJANDRO
christian.naula@epn.edu.ec

DIRECTOR: Ing. Adrián Egüez, MSc.
adrian.eguez@epn.edu.ec

DMQ,

CERTIFICACIONES

Yo, Christian Alejandro Naula Lomas declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Christian Alejandro Naula Lomas

Certifico que el presente trabajo de integración curricular fue desarrollado por Christian Alejandro Naula Lomas, bajo mi supervisión.

MSc. Vicente Adrián Eguez Sarzosa

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Christian Alejandro Naula Lomas
Vicente Adrián Egüez Sarzosa

DEDICATORIA

Dedico este trabajo a mi madre, padre, hermana, familia, novia, compañeros y amigos, de quienes he recibido el apoyo, las palabras, los consejos y el cariño necesario para nunca rendirme y buscar cumplir mis metas y objetivos. De mi para ellos este trabajo con mucho cariño.

AGRADECIMIENTO

Quiero agradecer en primeramente a mi madre, quien siempre ha sabido guiarme en todo mi crecimiento, no solo profesional sino también personal, enseñándome la importancia de ser una persona con principios y valores. A mi padre, quien ya no se encuentra conmigo en estos momentos, pero del que siempre recibí su apoyo incondicional y palabras de aliento en aquellos momentos difícil de mi crecimiento.

A mi hermana, por ser quien siempre me ha impulsado a conseguir todas las metas propuestas.

A mis docentes, de los cuales he aprendido un sinfín de conocimientos, quienes han sabido guiarme y formarme como profesional, y que siempre han inculcado valores que caracterizan a una persona de bien.

A mi tutor, del cual he recibido su guía y sus conocimientos a lo largo de este último paso necesario para poder culminar mis estudios universitarios.

A mi novia, compañeros y amigos, quienes han sido una parte fundamental en mi vida durante todo este proceso, aquellos con los que he disfrutado risas y llantos, buenos y malos momentos, y que siempre han sabido estar ahí para darme una mano en todo cuando más me hizo falta.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La gestión del inventario de una empresa es una labor ardua y en ocasiones compleja, sin embargo, esta nos ayuda a mantener un control sobre los productos que posee una organización a lo largo de toda su cadena de producción, convirtiéndola así en una pieza fundamental en la gestión de recursos empresariales. El presente trabajo de integración curricular propone una aplicación móvil, para el manejo del Inventario de una empresa como un módulo de un sistema ERP, desarrollada bajo una metodología Scrum y con prácticas DevOps que optimicen recursos en la elaboración de tareas específicas en la codificación y despliegue del aplicativo, tales como la construcción del ejecutable para dispositivos Android (archivo .apk) y la carga de versiones estables para su uso una vez terminado el desarrollo. El aplicativo será construido bajo un entorno de desarrollo web con tecnologías HTML, CSS, TypesCript y el Framework de desarrollo móvil Ionic.

PALABRAS CLAVE: ERP, Módulo de Inventario, DevOps, Aplicación Móvil.

ABSTRACT

The management of a company's inventory is an arduous and sometimes complex task; however, it helps us maintain control over the products that an organization has throughout its entire production chain, thus making it a fundamental piece in business resource management.

The present curricular integration work proposes a mobile application, for the management of the Inventory of a company as a module of an ERP system, developed under a Scrum methodology and with DevOps practices that optimize resources in the elaboration of specific tasks in coding and deployment. of the application, such as the construction of the executable for Android devices (.apk file) and the loading of stable versions for use once development is finished. The application will be built under a web development environment with HTML, CSS, TypeScript technologies and the Ionic mobile development framework.

KEYWORDS: ERP, Inventory Module, DevOps, Mobile Application.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Dentro del mundo empresarial existe un gran número de tecnologías que asisten al correcto funcionamiento de empresas y negocios. Cuando hablamos de dichas tecnologías englobamos un sin número de artefactos tecnológicos en donde claramente entra el software. Uno de los productos software indispensables en el manejo y asistencia empresarial es el ERP (Enterprise Resource Planning), el cual ayuda a automatizar aquellas tareas relacionadas con aspectos tanto operativos como productivos de las organizaciones.

Si bien un sistema ERP engloba un conjunto de aplicaciones diferentes, una de sus aplicaciones indispensables es el poder manejar el inventario de una organización. Teniendo en cuenta eso, se desarrollará el módulo de Inventario de un ERP, donde este módulo nos permite controlar de manera exhaustiva la mercancía existente dentro de una empresa a lo largo de todo el proceso productivo y comercial que esta desarrolla, desde la entrada hasta su respectiva salida.

El objetivo de la implementación de este módulo dentro de una empresa es en sí automatizar todos los procedimientos que comúnmente son realizados de manera manual, englobando aspectos tales como la administración, gestión, notificación, de la mercancía de cada organización, evitando así errores manuales comunes que conllevan a una pérdida económica para la empresa.

1.1 Objetivo general

Desarrollar e implementar un módulo de manejo de inventario de un ERP para la empresa Maticore-Labs haciendo uso de una metodología Scrum acompaña de prácticas de DevOps.

1.2 Objetivos específicos

1. Definir cada uno de los requerimientos iniciales de aplicativo a construir y representarlos en historias de usuario.

2. Establecer un diseño de las interfaces y modelamiento de base de datos que sirva como guía durante la construcción del aplicativo.
3. Hacer uso de tecnologías cómo Ionic para el desarrollo del módulo de inventario de un ERP.

1.3 Alcance

En el presente proyecto se realizará una implementación de un módulo de inventarios para un sistema ERP en un ambiente de desarrollo móvil, y haciendo uso de una metodología Scrum con prácticas de DevOps. Como etapa inicial del proyecto se abordará el diseño del módulo de inventario, obteniendo así una definición de los requisitos funcionales con sus respectivas historias de usuario, además de definir un diseño de la arquitectura del aplicativo, un diseño de interfaces y diagramas tanto de navegación cómo de base de datos.

Como fase siguiente tenemos la configuración y levantamiento de las tecnologías necesarias para el desarrollo tales como IDES, configuración de bases de datos y plataforma destinada para la ejecución de las prácticas de DevOps. Una vez configurado todo lo referente al ambiente, se procederá con el desarrollo e implementación como tal, el cual será dividido a lo largo de cuatro sprints o periodos de trabajo definidos con tareas específicas.

Finalmente se evaluarán los resultados del aplicativo en base a criterios de aceptación previamente definidos para cada una de las tareas y funcionalidades construidas, esto con el fin de garantizar un porcentaje alto en la satisfacción de los implicados.

1.4 Conceptos Técnicos

1.4.1 ERP (Enterprise Resource Planning)

ERP hace referencia a un tipo de software utilizado por organizaciones con el fin de cubrir y administrar diversas actividades comerciales realizados de manera cotidiana. En adición un ERP admite y controla aspectos organizacionales como la gestión de finanza, manejo del recurso humano, gestión de cadena de suministros, entre otros [1].

1.4.2 Aplicación Móvil

Una aplicación móvil es un software, el cual ha sido diseñado para resolver un problema específico y para ser ejecutado en un llamado “dispositivo móvil”, especialmente en teléfonos o celulares inteligentes “smartphones” o tabletas “tablets” [2].

1.4.3 Aplicaciones Híbridas

Son llamadas híbridas ya que combinan el uso del hardware y sistema operativo propio de las aplicaciones nativas, y el uso de diferentes elementos de las aplicaciones web en el desarrollo de su interfaz. Este tipo de aplicaciones son desarrolladas por herramientas comúnmente utilizadas en el desarrollo web, las cuales son HTML, CSS y un lenguaje de programación como JavaScript o TypeScript, haciendo uso también de distintos frameworks que ayudan a simplificar su proceso de desarrollo de aplicaciones multiplataforma, como podría ser Ionic [3].

1.4.4 Base de Datos no relacional

Son bases de datos que no cuentan con un esquema fijo, brindando así una mayor flexibilidad al momento de incluir campos de manera libre, sin la necesidad de redefinir toda su estructura [4].
Las bases de datos no relacionales pueden ser categorizadas en base a su modelo de datos, donde tenemos las siguientes [4]:

- Bases de datos claves – valor: Riak
- Bases de datos orientadas a documentos: MongoDB
- Bases de datos basadas en columnas: Cassandra
- Bases de datos de grafos: Infinite Graph.

1.5 Herramientas de Desarrollo

1.5.1 Lenguajes de Programación

El proyecto desarrollado hace uso de TypeScript como lenguaje de programación, acompañado del lenguaje de marcado de hipertexto HTML y del lenguaje de hoja de estilo en cascada CSS.

- TypeScript: Es un lenguaje de programación de código abierto que fue desarrollado por Microsoft. Está basado en el lenguaje de programación JavaScript, sin embargo, está pensado para el desarrollo de aplicaciones a gran escala. Una de las principales diferencias con su hermano JavaScript es que hace uso de un tipado estática, agregando así distintas reglas y especificando como deben ser usados diferentes tipos de valores [5].

- HTML5: El lenguaje HiperTextual de Etiquetas también conocido como HTML es un lenguaje estandarizado que permite asignar una estructura a una página web, sin embargo, no permite la asignación tanto de funcionalidad como de estilo [6].

- CSS: También conocido como Lenguaje de Hojas en Cascada, CSS es un lenguaje utilizado para dar estilo a una estructura previamente construida con HTML [7].

1.5.2 Framework de Desarrollo

- Ionic: es un SDK o "kit de desarrollo de software" de código abierto de frontend, utilizado en la construcción y desarrollo de aplicaciones híbridas, haciendo uso de lenguajes propios de la web como HTML, CSS y TypeScript. Este Framework permite que a partir de un único código se puedan generar aplicaciones para sistemas IOS, Android y la propia web [8].

- Firebase: Es una plataforma creada por Google diseñada con el fin de facilitar el desarrollo y creación de aplicaciones móviles. Esta plataforma se encuentra alojada en la nube, lo que facilita su integración con aplicaciones diseñadas para sistemas IOS, Android y web. Las herramientas que nos proporciona esta plataforma se encuentran agrupados en 4 categorías que son: Análisis, Desarrollo, Crecimiento y Monetización [9].

1.5.3 Base de datos

La base de datos utilizada para la elaboración del presente proyecto es Cloud Firestore, la cual es una de las varias herramientas que nos proporciona la plataforma Firebase. Esta es una base de datos NoSQL flexible utilizada en el desarrollo de aplicaciones

móviles y web. Esta base de datos se encuentra alojada en la nube de Google y maneja un tipo de almacenamiento de documentos organizados por colecciones, los cuales pueden anidarse entre ellos y admiten el albergue de sub-colecciones dentro de ellos [10].

1.5.4 Entorno de Desarrollo Integrado

Es un software utilizado en el diseño y codificación en una única interfaz gráfica. Este sistema cuenta con un editor de código fuente, automatización de compilaciones locales y un depurador [11].

Estos sistemas, además ayudan a los desarrolladores a encontrar soluciones a sus problemas, debido a que presentan un analizador de código, permitiendo así identificar los errores humanos en tiempo real [11].

Para la elaboración del presente proyecto es WebStorm, el cual es un entorno de desarrollo integrado utilizado para JavaScript y sus tecnologías relacionadas [12], donde entra el lenguaje de programación TypeScript que fue utilizado en la codificación del proyecto.

1.5.5 DevOps

El término proviene de la combinación de desarrollo (Development) y operaciones (Operations) [13]. Un modelo de DevOps consiste en la integración de los departamentos de desarrollo y operaciones, donde estos trabajan en conjunto a lo largo de todo el ciclo de vida de la aplicación, es decir desde la fase de desarrollo hasta la propia implementación [14].

1.5.6 Git

Es un sistema de control de versiones el cual trabaja de forma distribuida, donde los miembros del equipo confirmar el trabajo realizado en un ámbito local para proceder con una sincronización de su copia con aquella que se origina a raíz del repositorio albergado en un servidor [15].

Git trabaja bajo un sistema que únicamente añade información, estableciendo así un mecanismo que evita se elimine información, haciendo más sencilla la recuperación de datos [16].

1.5.7 GitLab

Gitlab es un software libre que consiste en un repositorio construido sobre Git, el cual está destinado a la gestión de proyectos. Mediante el uso de Gitlab podemos realizar un continuo seguimiento a los proyectos en curso, así como su estado actual e histórico [17].
Una de las mayores ventajas que tiene Gitlab frente a sus competidores directos es que al ser un software libre puede instalarse y utilizarse en cualquier tipo de servidor, además de que ofrece la opción de albergar su repositorio directamente en su plataforma bajo un costo [18].

1.6 Metodologías Ágiles

1.6.1 SCRUM

“Scrum es un marco de trabajo para la creación y mantenimiento de productos complejos [19]”. Se caracteriza por implementar un enfoque adaptativo mediante iteraciones con el fin de reducir el riesgo presentado con el cambio, enfatizando en prácticas definidas para la gestión de proyectos.
Bajo el uso de Scrum se priorizan eventos que ya están previamente definidos, los cuales corresponden a bloques de tiempo que cuentan con una duración máxima [19].

Sprint

Un Sprint es un evento que tiene duración menor a un mes, tiempo en el cual se crea un incremente del producto que se encuentra en desarrollo. Este incremente debe ser totalmente utilizable y poseer características suficientes para ser desplegado. Cada Sprint contiene las siguientes etapas [19]:

- Planificación del sprint (Sprint Planning)
- Revisión del sprint (Sprint Review)
- Retrospectiva del sprint (Sprint Retrospective)

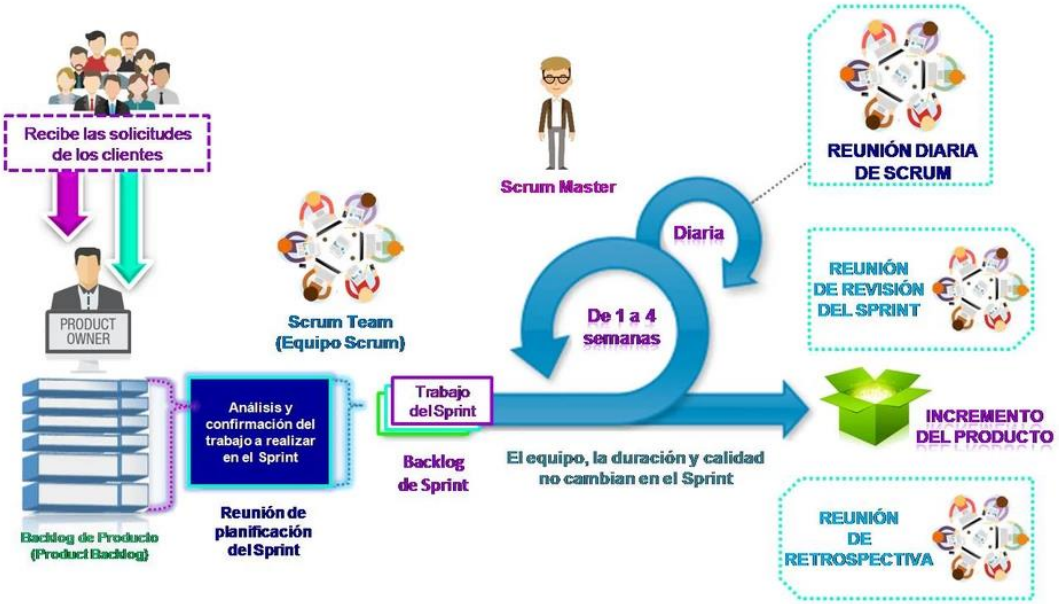

Figura 1. Marco de trabajo Scrum

2 METODOLOGÍA

Durante la elaboración del proyecto se utilizó SCRUM como metodología, tomando como duración de cada Sprint una semana con 6 horas de trabajo diarias. Adicionalmente, se establecieron flujos de DevOps que acompañarán todo el ciclo de vida del aplicativo.

2.1 Equipo Scrum

El equipo de Scrum está conformado por un dueño de producto (Product Owner), el cual es el encargado de asegurar que el equipo aporte valor auténtico al proyecto. Adicionalmente tenemos el equipo de desarrollo y un Scrum Master.
El equipo de Scrum está conformado de la siguiente manera.

- Product Owner: Msc. Adrián Eguez
- Scrum Maste: Msc. Adrián Eguez
- Equipo de Desarrollo: Christian Naula

2.2 Implementación DevOps

El flujo de ejecución de DevOps entre las distintas herramientas se conforma por cuatro etapas principales, tal y como se puede visualizar en la Figura 2. Estas etapas se detallan a continuación

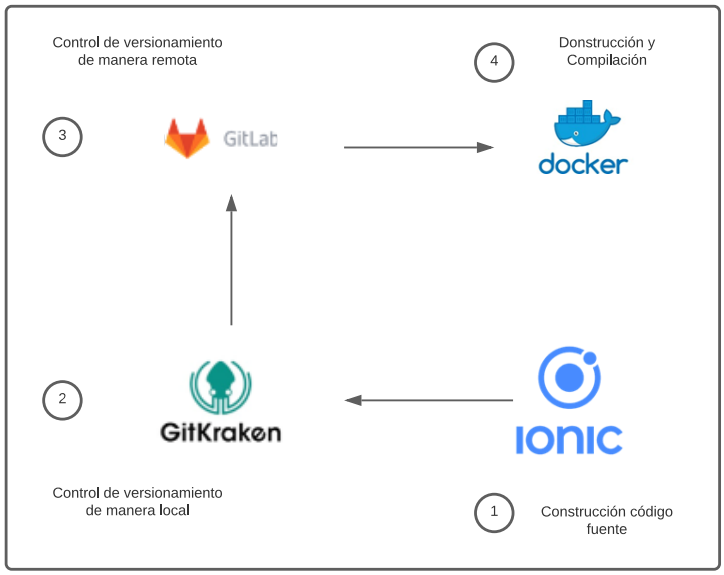

Figura 2. Flujo de ejecución de DevOps

Etapa 1

En la primera etapa se realiza la construcción del código fuente perteneciente a un incremento en el entorno de desarrollo defino para el aplicativo. Para el presente proyecto se hizo uso del Framework de Ionic, el cual hace uso de tecnologías y lenguajes orientados al desarrollo web, tales como HTML, CSS y TypeScript.

Dentro del proyecto de Ionic se hizo uso de servicios proporcionados por la plataforma Firebase, los cuales son: Cloud Firestore para el manejo de base de datos, Athentication para el control de ingreso a la plataforma mediante usuarios y contraseñas, y finalmente App Distribution para el despliegue continuo de la aplicación mediante su archivo .apk.

Etapa 2

La segunda etapa consiste en el control de las versiones generadas a raíz del código fuente de manera local. Este proceso se realiza mediante el uso del software GitKraken, el cual trabaja sobre la tecnología Git. Desde este software se realizará el envío al repositorio local a un repositorio remoto.

Etapa 3

La tercere etapa corresponde al manejo de versiones de manera remota a través del uso de la plataforma online GitLab. En esta etapa se realizan las integraciones respectivas entre lo almacenado de manera remota y los cambios que llegan desde el repositorio local. Esta herramienta también nos permite realizar las configuraciones de las prácticas de DevOps para la integración continua.

Este repositorio remoto se integra de 3 ramas, donde destacan las ramas “Dev” y “testing” utilizadas para la ejecución de los flujos de DevOps utilizados. En la rama Dev, se ejecutan las tareas correspondientes a la fase de construcción del aplicativo, generando así el archivo .apk que corresponde al archivo ejecutable utilizado para instalar el aplicativo en dispositivos móviles.
Por otro lado, a rama “testing” se encarga de ejecutar las tareas que corresponden al despliegue del aplicativo y facilitar su distribución.

Etapa 4

En la cuarta y última etapa entra en juego un contenedor de Docker, que es donde se levantará la imagen que se encargará de ejecutar todos los comandos e instrucciones necesarias para la ejecución de la integración continua que fue configurada en la etapa 3.
Para este proceso el contenedor hace uso de una imagen prediseñada que alberga algunas dependencias por defecto, como puede ser el propio SDK de Android, que es indispensable para la construcción del archivo APK.

2.2.1 Configuración de DevOps

Previo a configurar el flujo de DevOps es necesario se disponga de un repositorio creado en la plataforma de GitLab, donde almacenaremos todos los archivos del proyecto de Ionic creados para el desarrollo del aplicativo y cada incremento. En la Figura 3 se muestra el repositorio creado en GitLab, el cual puede ser encontrado en el Anexo I.

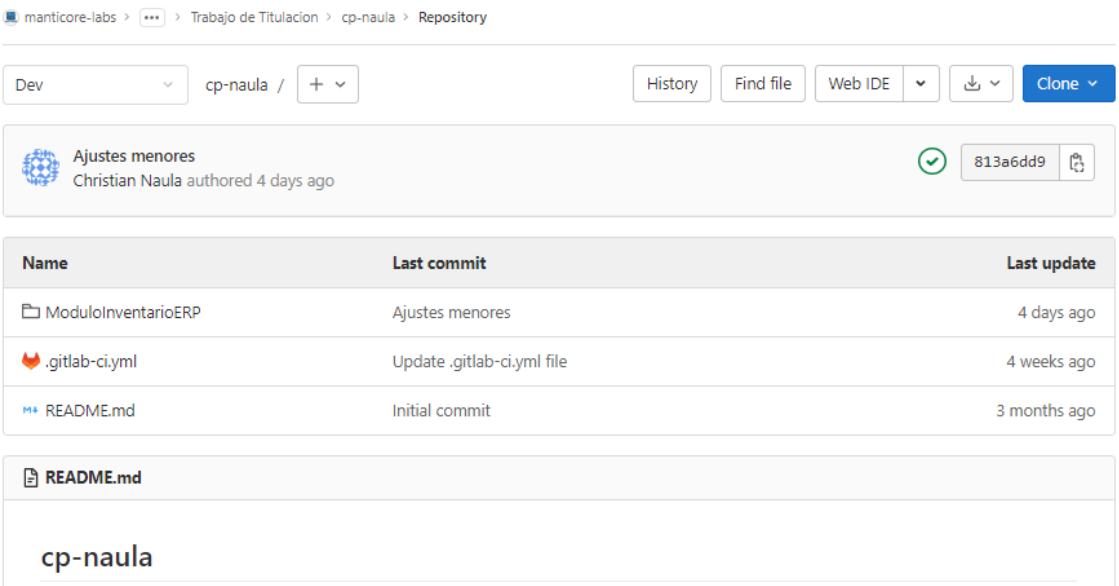

Figura 3. Repositorio remoto en GitLab

Una vez creado el repositorio remoto es necesario vincularlo de alguna manera a un repositorio local, para lo cual se creó una carpeta contenedora del repositorio dentro de un directorio del equipo utilizado para el desarrollo. En la Figura 4 se muestra el directorio que albergará el repositorio local. Una vez creado el directorio, abrimos el software GitKraken y lo usaremos para clonar el repositorio remoto en la carpeta “Repositorio”. En la Figura 5 se muestra el proceso de clonación realizado en GitKraken.

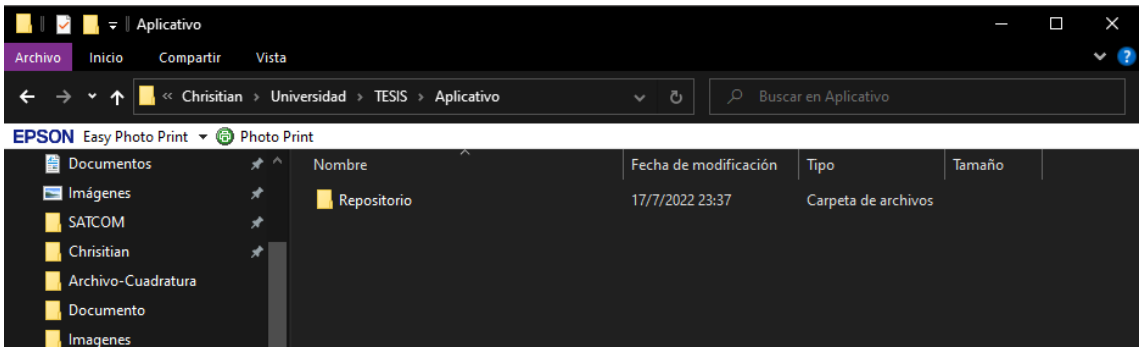

Figura 4. Carpeta contenedora para el repositorio local.

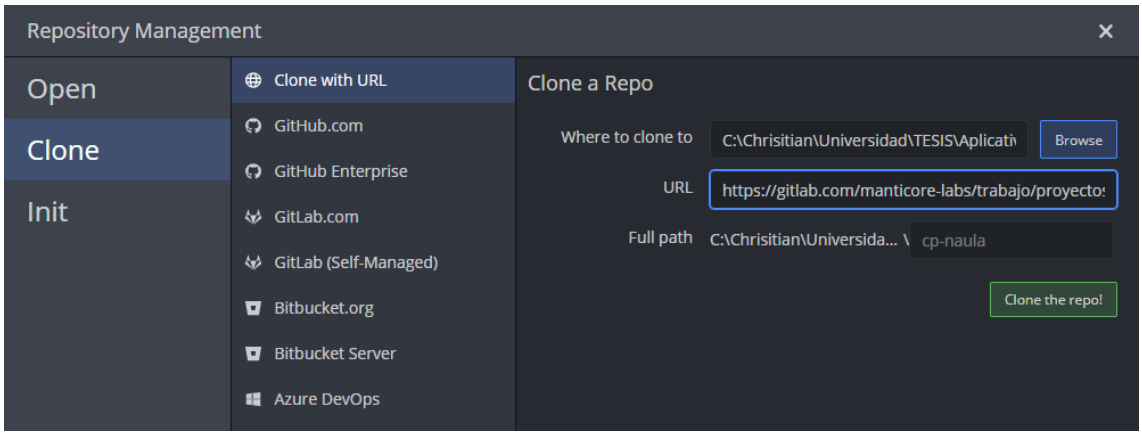

Figura 5. Clonación de repositorio remoto en el directorio local

Una vez tenemos vinculado el repositorio remoto con el local es necesario realizar las configuraciones de GitLab CI/CD, que es la herramienta dentro de GitLab que nos permite utilizar las prácticas de DevOps. Para este caso, haremos uso de un contenedor propio mediante el uso de Docker. Como primer paso debemos deshabilitar la opción para usar contenedores compartidos propios de GitLab, en la Figura 6 se muestra la opción mencionada deshabilitada.

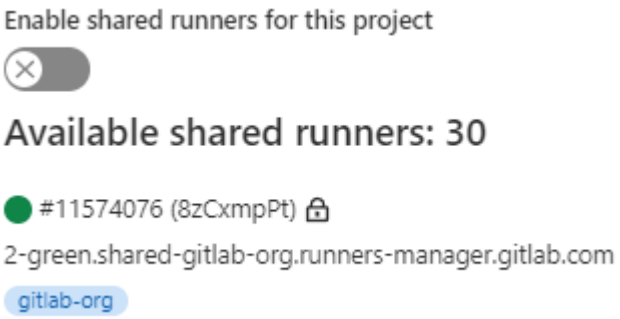

Figura 6. Uso de contenedores compartidos en GitLab deshabilitado.

Como siguiente paso debemos descargar e instalar el software GitLab Runner, abrir una consola de comandos como administrador y ejecutar el archivo .exe que viene en el paquete descargado junto con el comando “register” con el fin de registrar un nuevo Runner. Aquí se nos solicitará pegar una URL y un Token para continuar con el registro, misma información que se nos proporciona en la sección de configuración de CI/CD de Gitlab tal y como se muestra en la Figura 7. Continuando con el proceso se nos solicita una descripción del Runner y etiquetas y demás datos opcionales. En la Figura 8 se muestra la consola de comandos con el proceso de registro correcto.

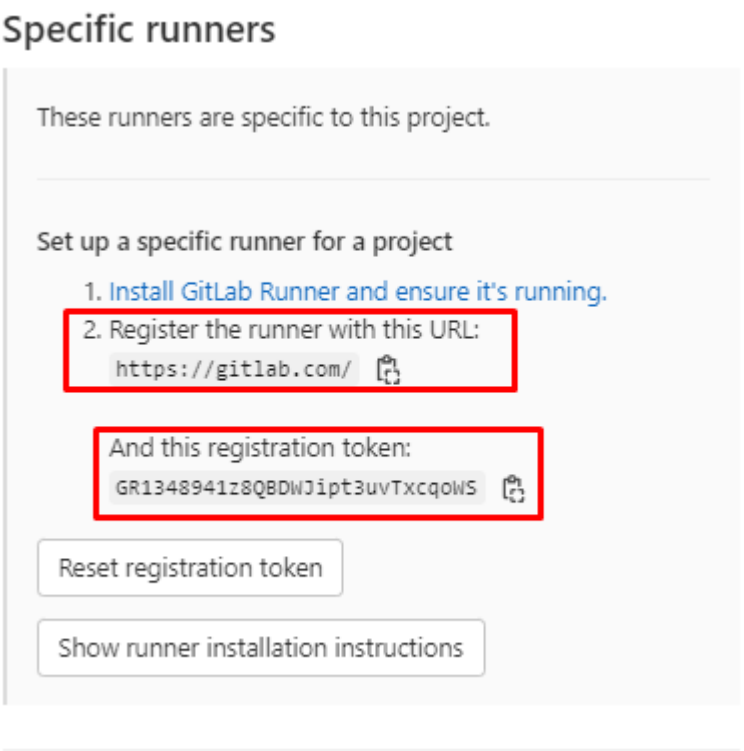

Figura 7. URL y Token de registro de un nuevo Runner en GitLab.

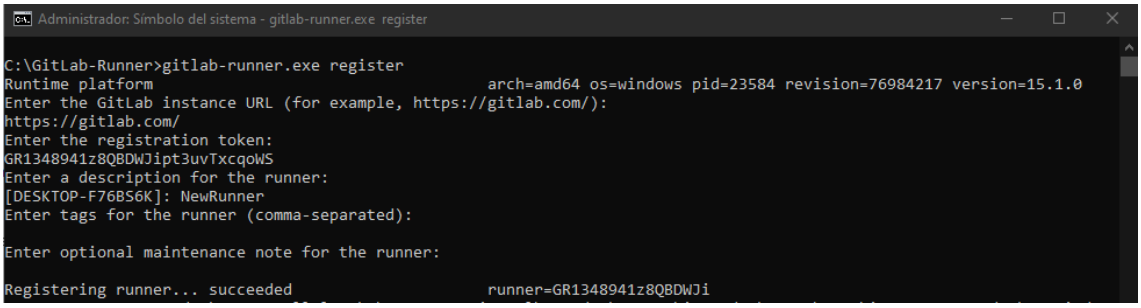

Figura 8. Registro de nuevo Runner

Con el Runner registrado de manerea satisfactoria debemos configurar un ejecutor para este, para lo cual seleccionaremos la opción de Docker e ingresaremos la imagen que se utilizará para ejecutarlo. En este caso elegimos la imagen circleci/android en su versión 28-node, la cual viene con dependencias de Android preinstaladas como por ejemplo el SDK de Android, el cual es necesario para el correcto funcionamiento de las prácticas de DevOps. En la Figura 9 se visualiza la configuración del ejecutor del Runner con Docker.

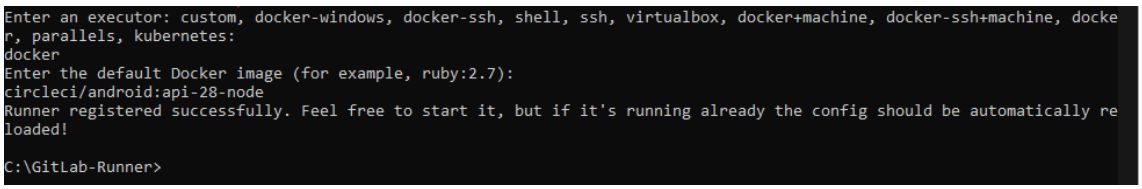

Figura 9. Configuración de Docker como ejecutor del Runner registrado.

Con el proceso de registro del Runner local y el ejecutor con Docker completado, debemos levantar el Runner con el comando “start” para verificar que el contenedor se encuentra correctamente vinculado a GitLab. En la Figura 10 se muestra el comando para levantar el contenedor y en la Figura 11 se muestra el Runner ya disponible en GitLab.

Figura 10. Comando para levantar el contenedor del Runner.

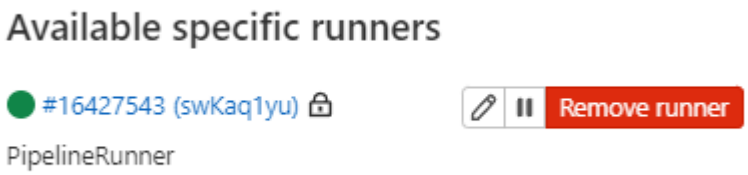

Figura 11. Runner disponible en GitLab

Finalizada las configuración del contenedor que ejecuta el runner local, se debe configurar un archivo con nombre ".gitlab-ci.yml" el cual alberga todas las configuraciones y tareas necesarias para la ejecución de las actividades de DevOps. En la Figura 12 se muestra el archivo creado en el repositorio remoto en GitLab.

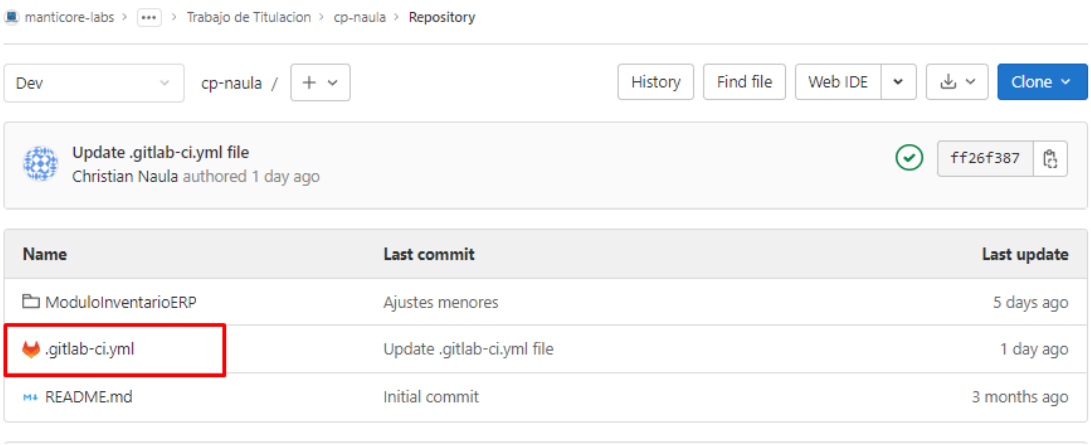

Figura 12. Archivo ".gitlab-ci.yml" en el repositorio

Como primer punto de configuración en el archivo YML es el campo "image", que corresponde a la imagen base que se ejecutará en el contenedor de Docker, en este caso se eligió la imagen circleci/android en su versión 28-node. Esta imagen tiene la particularidad de tener instalado varias dependencias de Android, incluido su SDK. Se puede encontrar en enlace directo a la imagen en el Anexo VI. En la Figura 13 se muestra la configuración del campo "image".

```
1
2   image: circleci/android:api-28-node
3
```

Figura 13. Configuración del campo "image"

El segundo campo por configurar es “before_script”, donde se deben colocar los comandos por defecto necesarios para que la imagen esté lista para utilizarse. En este apartado se suele realizar tareas como la instalación de dependencias y librerías que no vienen instaladas por defecto en la imagen seleccionada. En la Figura 14 se muestra la configuración del campo en el archivo.

```
4   before_script:
5     - sudo apt-get update
6     - sudo apt-get install nodejs
7     - npm --version
8     - npm install
9     - sudo npm install -g @angular/cli
10    - sudo npm i -g @ionic/cli
11    - sudo npm install -g firebase-tools@10.9.2
12
```

Figura 14. configuración del campo "before_script".

Siguiendo con la configuración pasamos al campo “stages”, donde se indica el número y las etapas que serán ejecutadas durante la ejecución del pipeline resultante. En este caso se definieron dos stages, la primera es la etapa “build” que se ejecutará únicamente al realizar un push en la rama “Dev”, y la segunda es la etapa “deploy” que se ejecutará al realizar un merge en la rama “testing”. En la Figura 15 se muestra la definición de las dos etapas mencionadas.

```
14  stages:
15    - build
16    - deploy
17
```

Figura 15. configuración del campo "stages".

Como siguiente etapa en la configuración es el “build-job” corresponde a las tareas que se ejecutan en cada uno de los “stages” previamente definidos. En este paso se definieron 3 características que son: 1) “stage” – la etapa a la que corresponde el job, 2) “script” – las tareas a ejecutar durante el job, y 3) “only” – la rama en la que se ejecutará el job.

En la Figura 16 se muestra la configuración del “build-job” para la etapa “build”, el cual se ejecutará únicamente al indicar un cambio en la rama “Dev” del repositorio remoto.

Aquí se ejecutan todos los comandos necesarios para la construcción del archivo ejecutable del aplicativo o APK.

```
build-job:
  stage: build
  script:
    # Construcción APK
    - dir
    - nodejs -v
    - cd ModuloInventarioERP
    - npm install
    - ionic build
    - ionic capacitor add android
    - ionic capacitor sync
    - cd android
    - sudo chmod +x gradlew
    - ./gradlew assembleDebug
    - cd app/build/outputs/apk/debug
    - dir
  only:
    - Dev
```

Figura 16. Configuración del campo "build-job" en la rama “Dev”.

En la Figura 17Figura 16 se muestra la configuración del “build-job” para la etapa “deploy”, el cual se ejecutará únicamente el realizar un merge desde la cualquier rama hacia la rama “testing” del repositorio remoto. Aquí se ejecutan todos los comandos necesarios para la construcción del archivo ejecutable del aplicativo o APK, y luego se procederá a cargar el archivo construido en el servicio de Firebase App Distribution.

```
deploy-job:
  stage: deploy
  script:
    - nodejs -v
    - cd ModuloInventarioERP
    - npm install
    - ionic build
    - ionic capacitor add android
    - ionic capacitor sync
    - cd android
    - sudo chmod +x gradlew
    - ./gradlew assembleDebug
    - cd app/build/outputs/apk/debug
    - cd /builds/manticore-labs/trabajo/proyectos/trabajo-de-titulacion/cp-naula/ModuloInventarioERP/android/app/build/outputs/apk/debug
    #- cambiamos el nombre a pp-debug + dia + mes + año + minutos
    - sudo mv app-debug.apk  app-debug-$(date +%d-%m-%y-%M).apk
    #- creacmos variable global con el mismo nombre que tendrá el app
    - export nombreapp=app/build/outputs/apk/debug/app-debug-$(date +%d-%m-%y-%M).apk
    - echo $nombreapp
    #- volvemos a la carpeta android dentro del proyecto de ionic
    - cd /builds/manticore-labs/trabajo/proyectos/trabajo-de-titulacion/cp-naula/ModuloInventarioERP/android/
    - if [ -f $nombreapp ]; then firebase appdistribution:distribute $nombreapp --app 1:30196687884:android:77226124e2b22590833a29 --token "1//
0dLaENiUUoDD3CgYIARAAGA0SNwF-L9IrNCo83MJRBfEeTh1wxHRIiCSIpiUN2KCLPHfdHTyyT9XZXyGl7nhlxCHrqD8_lzg37_A"; fi
  only:
    - testing
```

Figura 17. Configuración del campo "build-job" en la rama “testing”.

2.2.2 Ejecución de DevOps

Una vez configurado el entorno y los artefactos necesarios para una correcta ejecución de las prácticas de DevOps, podemos ya probar el flujo completo. Para esto se debe debemos realizar el envío de datos desde el repositorio local hacia el remoto. Esta actividad se realiza cuando se ha añadido un incremento importante de funcionalidades en el aplicativo o cuando se obtiene una versión estable que puede ser desplegada sin problemas.

El primer “stage” a probar es el de “build”, donde como primer paso, se debe realizar un commit de los cambios realizados en el repositorio local en la rama Dev tal como se muestra en la Figura 18. Posteriormente procedemos a realizar un push de la rama Dev en el repositorio local a la rama Dev del repositorio remoto, generando que se ejecute un pipeline con el “stage build”. En la se muestra el push realizado desde GitKraken.

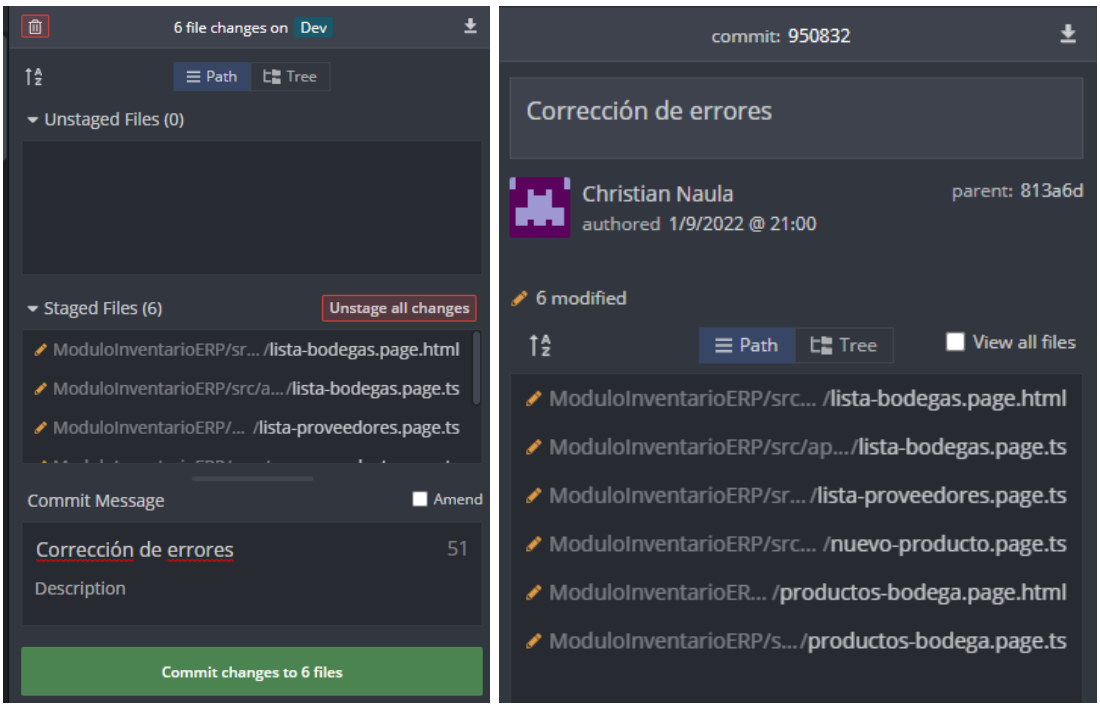

Figura 18. Commit a la rama Dev local.

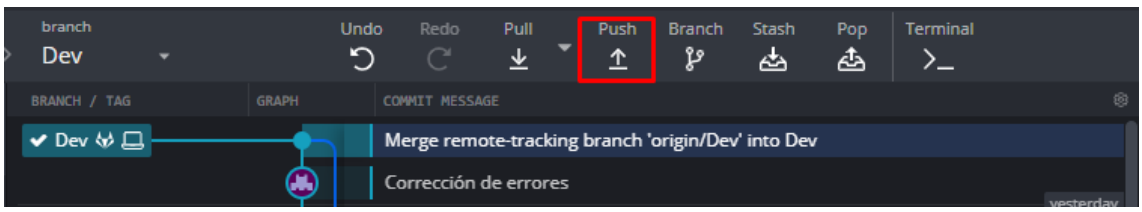

Figura 19. Push a la rama Dev remota.

Con el envío de datos a la rama “Dev” del repositorio remoto comienza un flujo de DevOps, generando un nuevo pipeline para la etapa “build”, siguiendo las instrucciones indicadas previamente en la Figura 16. La ejecución de las instrucciones se realiza en un contenedor de Docker, el cual necesita estarse ejecutando previamente para evitar errores en la ejecución del pipeline. En la Figura 20 se muestra el contener ejecutando el pipeline dentro de Docker y en Figura 21 se muestra el pipeline en ejecución desde GitLab.

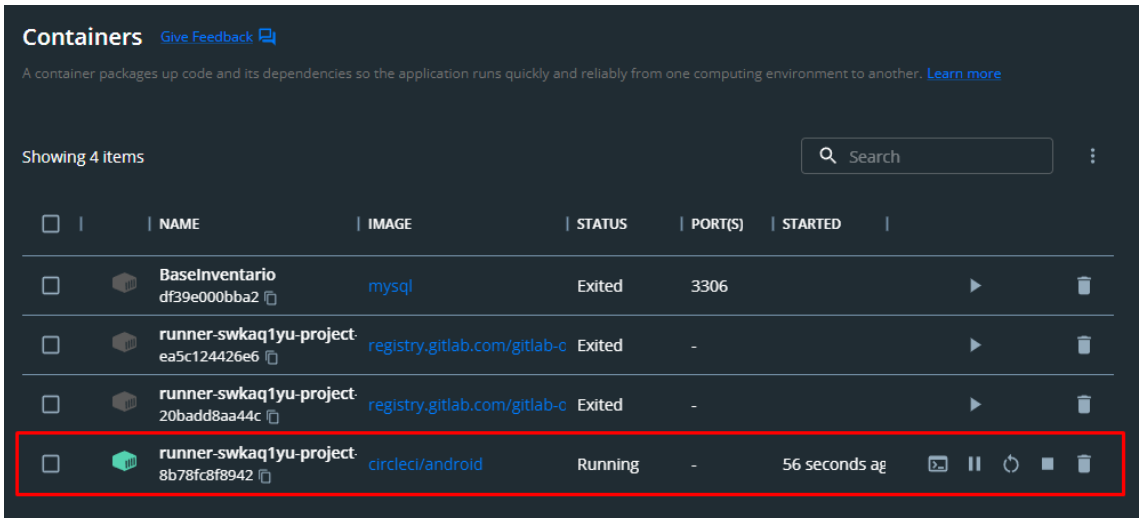

Figura 20. Contenedor ejecutándose dentro de Docker

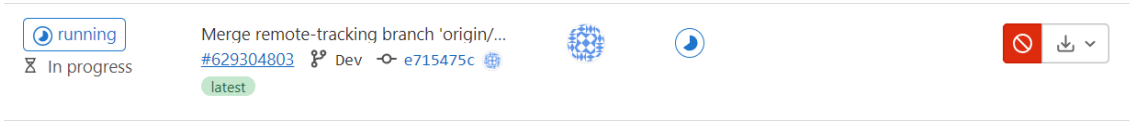

Figura 21. Pipeline de la rama Dev en ejecución.

Cuando se termine la ejecución de todas las instrucciones configuradas en el archivo “.gitlab-ci.yml” para el “satege build” se genera un archivo APK y GitLab devuelve el mensaje indicando que el pipeline terminó su ejecución con éxito. En la Figura 22 se muestra el pipeline completado en GitLab y en la Figura 23 se muestra el resultado de la ejecución de las instrucciones configuradas en el archivo YML.

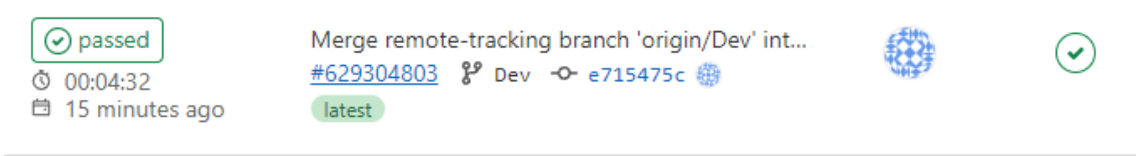

Figura 22. Pipeline de la rama Dev completado

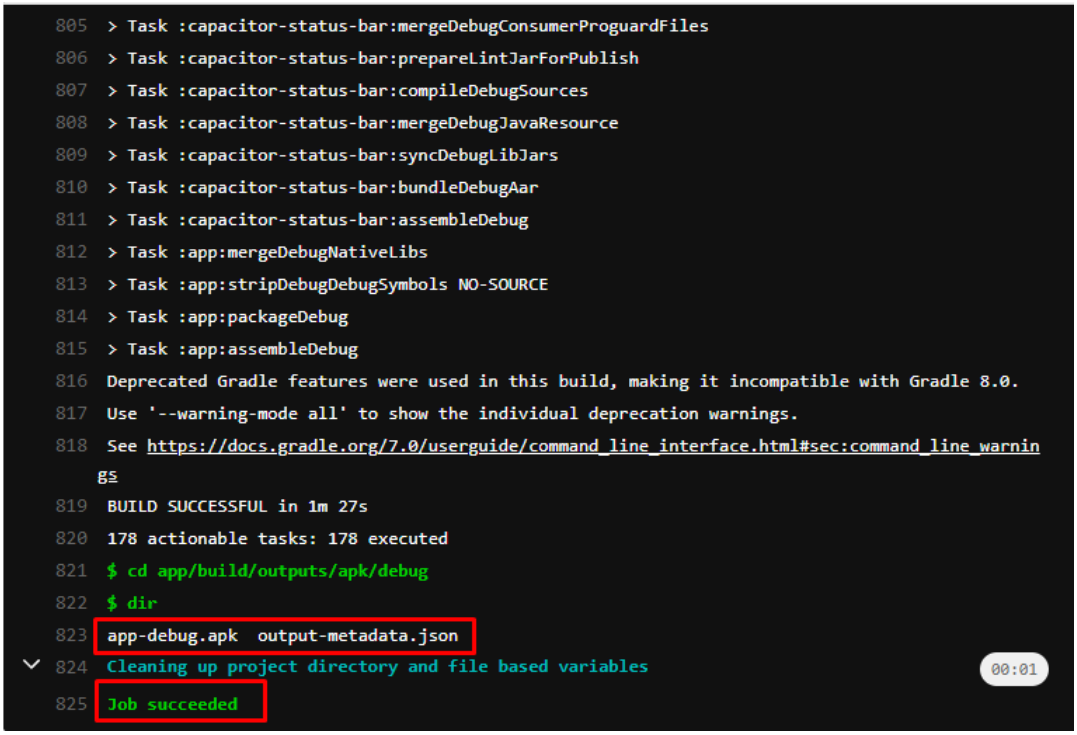

Figura 23. Resultado de la ejecución en la rama Dev

El segundo y último “stage” configurado es el de “deploy”, para lo cual necesitamos realizar un envío de datos a la rama “testing” mediante un merge desde GitLab. En la Figura 24 se observa la creación del merge desde la rama “Dev” hacia la rama “testing”.

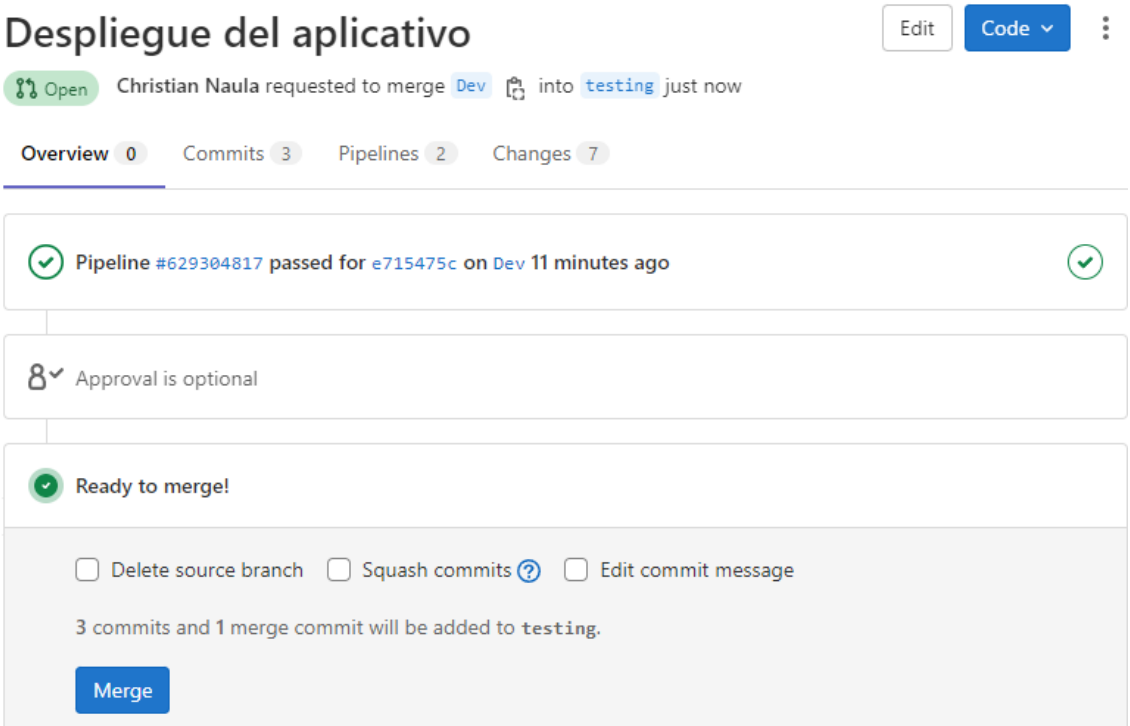

Figura 24. Creación de merge desde la rama Dev hacia la rama testing

Este envío de datos genera un nuevo flujo de DevOps, donde GitLab nos muestra un nuevo “job” ejecutándose, correspondiente a la etapa de “deploy” dentro del archivo YML. En la Figura 25 se muestra el pipeline de la rama testing en ejecución.

Figura 25. Pipeline de la rama testing en ejecución.

Una vez finalizada la ejecución de las instrucciones se despliega un mensaje indicando que el archivo APK fue cargado en App Distribution de manera correcta tal como se visualiza en la Figura 26. Adicionalmente, en la Figura 27 se muestra el pipeline ejecutado en GitLab

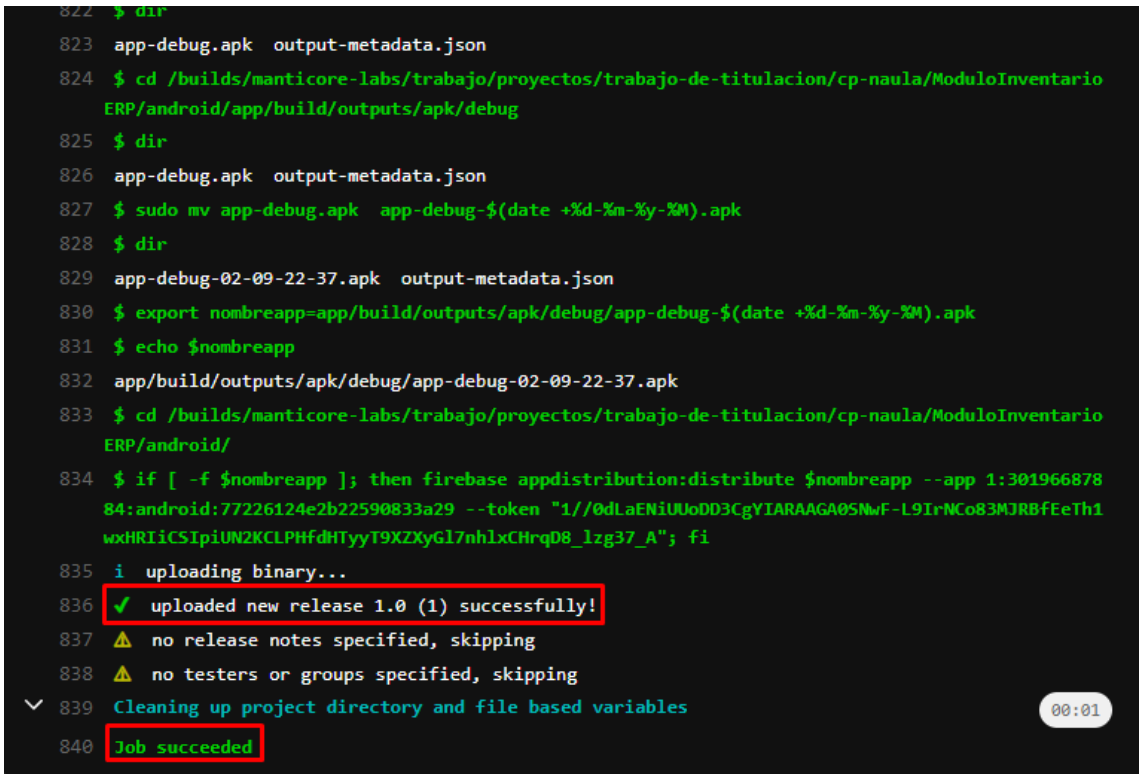

Figura 26. Resultado de la ejecución en la rama testing

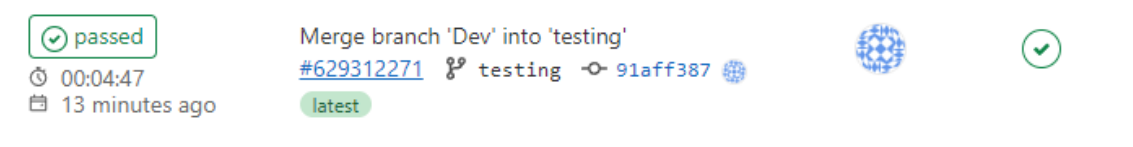

Figura 27. Pipeline de la rama testing completado

2.3 Implementación del Aplicativo

2.3.1 Requerimientos Iniciales

Con el fin de reunir los requerimientos del sistema se realizaron sesiones de entrevista con el Product Owner, donde se identificaron las necesidades a cubrir con el sistema, los cuales fueron agrupados en módulos separados.

MÓDULO	REQUERIMIENTOS
Sesión	Gestionar los usuarios del sistema
	Crear nuevos usuarios
	Registrar un nuevo usuario con 3 tipos de identificaciones: C.I., RUC o Pasaporte
Inventario	Gestionar los productos almacenados en bodega
	Listado de productos
	Crear nuevos productos
	Editar productos existentes
Proveedores	Gestionar los proveedores de productos
	Ingresar productos por proveedor
	Listado de Proveedores
Movimientos	Gestionar el movimiento de productos de las bodegas
	Listado de movimientos por producto
	Crear movimientos de 2 tipos: ingreso y egreso
	Asignar una subclase a cada movimiento
	Calcular el valor de los productos después de cada ingreso
	Manejo de histórico de movimientos
Bodegas	Gestionar bodegas
	Listado de bodegas
	Listado de productos almacenados en cada bodega
	Creación de nuevas bodegas
	Manejo de valor total por bodega

Tabla 1. Requerimientos iniciales

2.3.2 Prototipo de aplicación móvil

Luego de haber definido los requerimientos iniciales y sus respectivas historias de usuario, se continuo con el diseño del prototipo para la aplicación móvil, esto mediante el uso de la herramienta de diseño online Figma. Se puede encontrar el enlace al prototipo en el Anexo II.

Una vez aprobado el prototipo final de la aplicación, se procedió con la planificación para la implementación mediante sprints. Se puede observar una vista general del prototipo en la Figura 28. Vista general del prototipo móvil

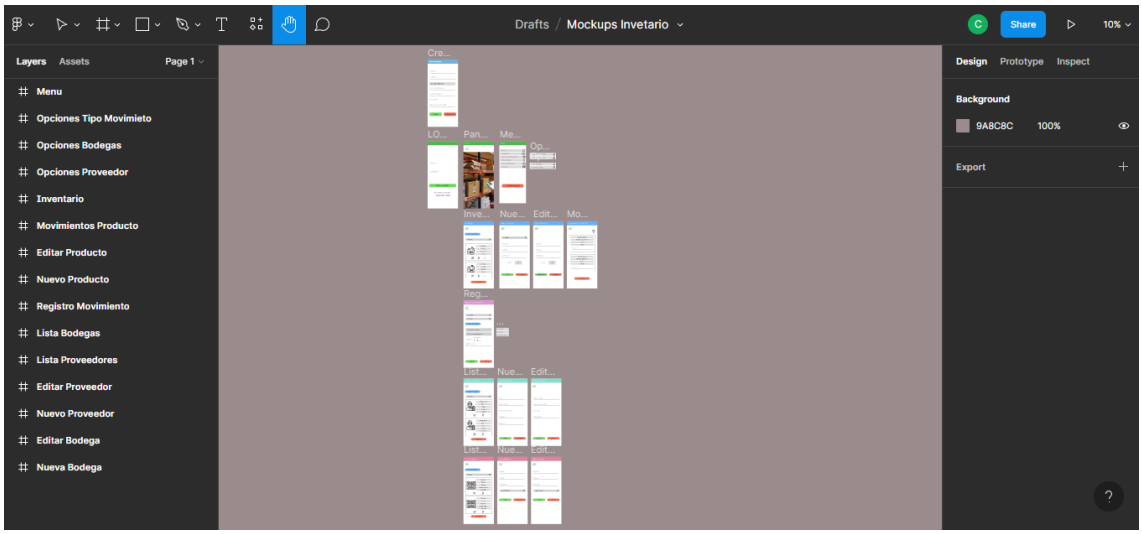

Figura 28. Vista general del prototipo móvil

2.3.3 Release Planing

Debido a que ya tenemos definido las historias de usuario en el desarrollo de Product Backlog, se estableció historias de usuario para cada uno de los sprints, los cuales servirán como una guía durante el desarrollo e implementación de la aplicación.
Para cada uno de los sprints se plantearon objetivos, los cuales son:

- Objetivo 1.- Validar el despliegue continuo mediante DevOps.
- Objetivo 2.- Gestionar la información correspondiente al inventario.
- Objetivo 3.- Gestionar la información de proveedores de productos.
- Objetivo 4.- Gestionar información de bodegas y los productos dentro de ellas.
- Objetivo 5.- Gestionar el movimiento de los productos.

Sprint 1			Sprint 2			Sprint 3			Sprint 4		
COD	**Prioridad**	**Pts**	**COD**	**Prioridad**	**Pts**	**COD**	**Prioridad**	**Pts**	**COD**	**Prioridad**	**Pts**
US02	Media	3	US6	Alta	13	US7	Alta	13	US5	Alta	16
US01	Media	3									
US03	Alta	10									
Total	Alta	16	**Total**	Alta	13	**Total**	Alta	13	**Total**	Alta	16

Nomenclatura
COD = Código de historia de usuario

Tabla 2. Release Plan

2.3.4 Arquitectura de la aplicación

La aplicación hace uso de una arquitectura modelo-vista-controlador sin servidor, mediante el uso de la plataforma de Firebase, utilizando así su servicio de Authentication para el control y gestión del acceso de usuarios, y Cloud Firestore como base de datos NoSQL o no relacional. Se muestra la arquitectura de la aplicación en la Figura 29. Arquitectura de la aplicación.

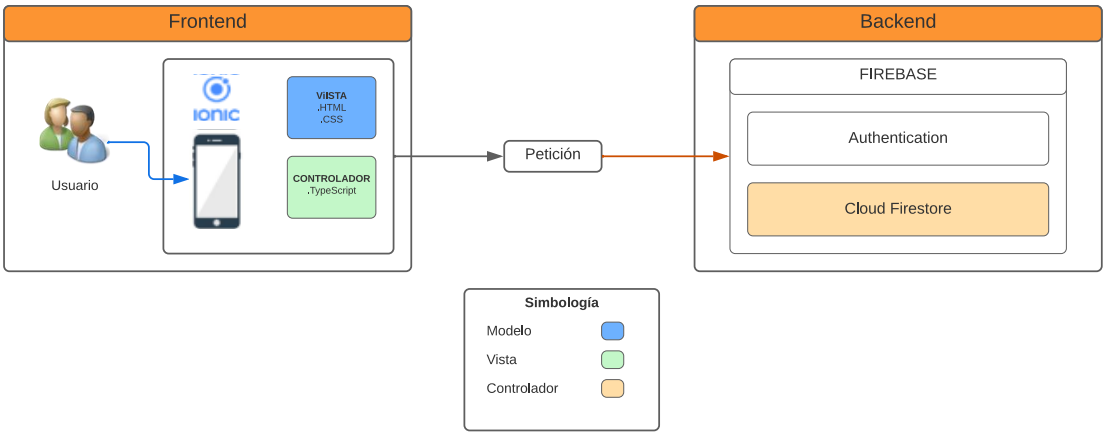

Figura 29. Arquitectura de la aplicación

2.3.5 Modelo de documentos en Firestore

Se utilizó un total de 7 colecciones de Firestore para la construcción del aplicativo, los cuales se manejan en 2 niveles diferentes de profundidad. Al nivel más alto tenemos 6 colecciones destinadas al manejo de los productos, proveedores y sus respectivos movimientos, mientras que en el segundo nivel de profundidad se manejan los productos

almacenados en cada una de las diferentes bodegas. En la Figura 30 se muestra se presenta el modelo de colecciones creadas.

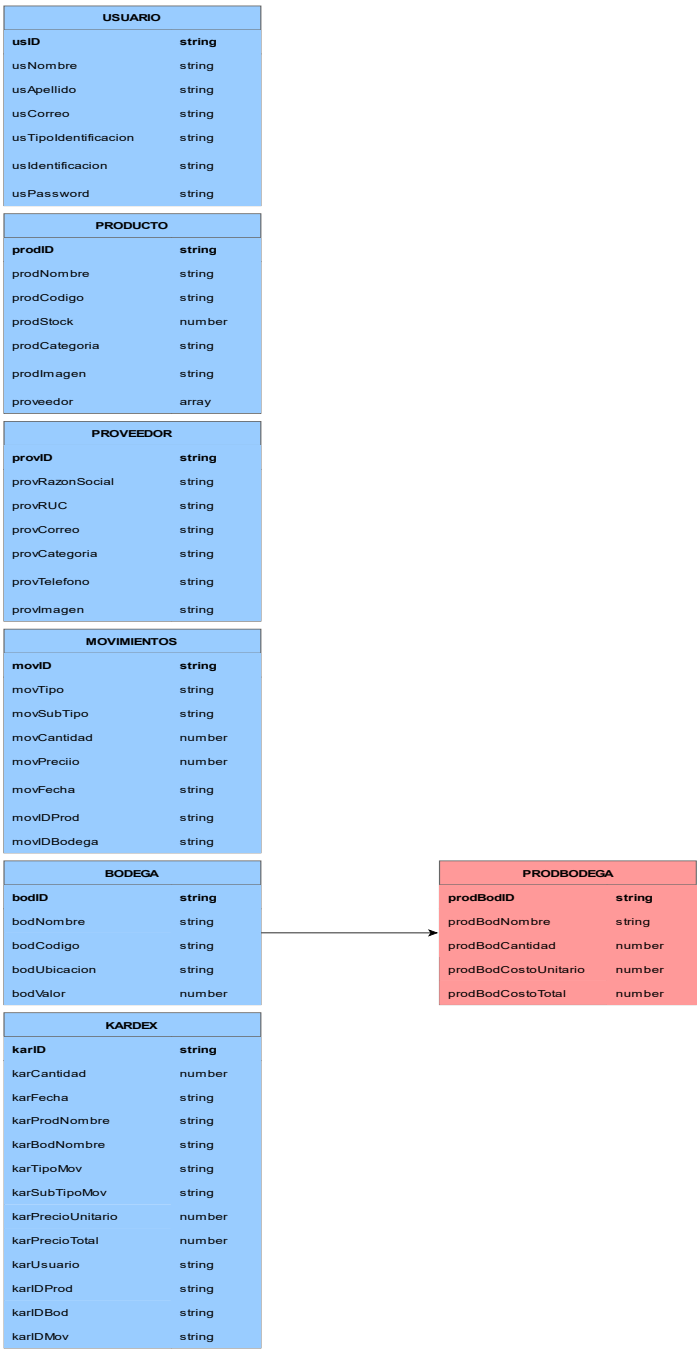

Figura 30. Modelo de datos de Firestore

2.3.6 Diagrama de Navegación

Se puede observar el diagrama de navegación de la aplicación desarrollada en la Figura 31.

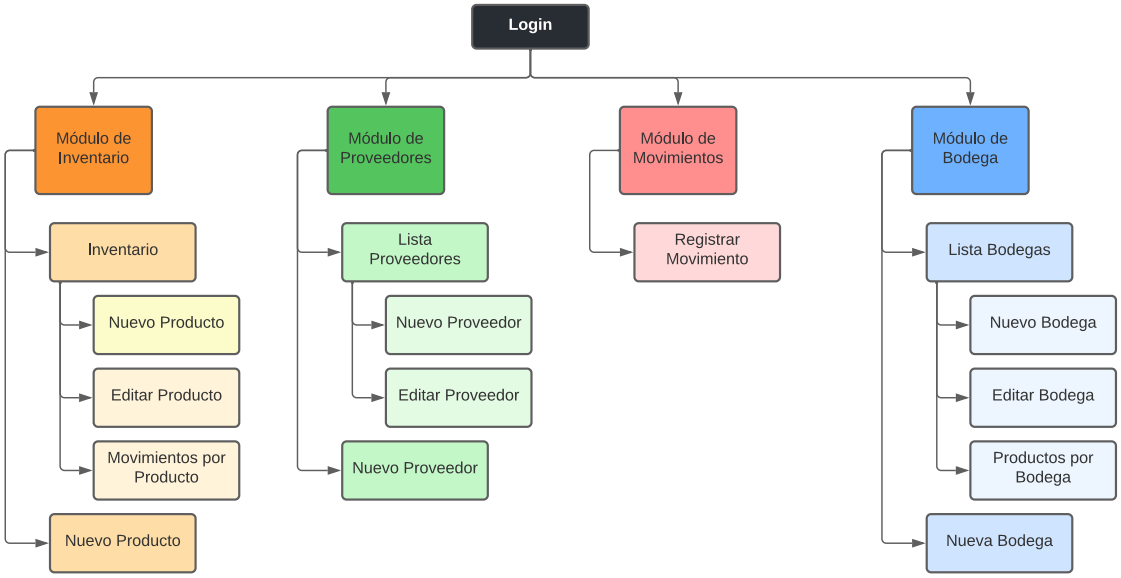

Figura 31. Diagrama de Navegación

2.3.7 Product Backlog

En la Tabla 3 se muestran los requerimientos de la aplicación móvil junto con su prioridad.

Código	Nombre de US	Descripción US	Prioridad	Pts
US01	Registro de Usuarios	Como usuario del sistema quiero crear nuevos usuarios para extender el alcance de mi aplicativo.	Media	3
US02	Inicio de Sesión	Como usuario del sistema quiero iniciar sesión en el sistema para acceder a las funcionalidades de esta.	Media	3
US03	Gestión de Inventario	Como usuario del sistema quiero poder listar, crear, editar y eliminar productos de mi inventario para llevar un control fiel de mi mercadería.	Alta	10
US05	Gestión de Registro de Movimientos	Como usuario del sistema quiero registrar movimientos de los productos para controlar los productos entrantes y salientes.	Alta	16
US06	Gestión de Proveedores	Como usuario del sistema quiero poder listar, crear, editar y eliminar proveedores para tener comunicación más efectiva con ellos.	Alta	13

US07	Gestión de Bodegas	Como usuario del sistema quiero poder listar, crear, editar y eliminar bodegas para tener un orden de los productos almacenados en cada una de ellas.	Alta	13

Tabla 3. Product Backlog

2.3.8 Sprints

Sprint 0

Objetivo: Implementar un ambiente de desarrollo estable y completar los flujos de construcción y despliego correspondientes a DevOps.

Sprint Planning

Durante el sprint 0 se realizará la configuración del ambiente de desarrollo, llevando a cabo la instalación de herramientas tales como IDE de desarrollo, IONIC como framework, Firestore como sistema de gestión de bases de datos y GitKraken como sistema de manejo de versiones de código.
Adicionalmente se realizará la configuración de los flujos de construcción de código y despliegue de versiones correspondientes a las prácticas de DevOps dentro de GitLab.

Daily meeting

Durante la ejecución del sprint se presentaron las siguientes dificultades.

Dificultades encontradas	**Solución**
Muestra mensaje de error al instalar el SDK de Android durante la ejecución de los pipelines de GitLab.	Usar una imagen prediseñada donde ya se encuentre instalado el SDK de Android por defecto.
El servicio de Firebase App Distribution prohíbe la carga de dos archivos .apk con un mismo nombre	Se asignó un nombre diferente a cada una de las versiones nuevas que se enviaban al servicio de Firebase. Este nombre tomaba la fecha y hora del commit.

Tabla 4. Resumen del daily meeting sprint 0

Codificación

En este sprint se instaló el IDE WebStorm donde se creó el proyecto de Ionic respectivo. Se creo las ramas “Dev” para la construcción del aplicativo y la rama “testing” para el despliegue de este, haciendo uso del servicio de App Distribution de Firebase y su integración con GitLab.

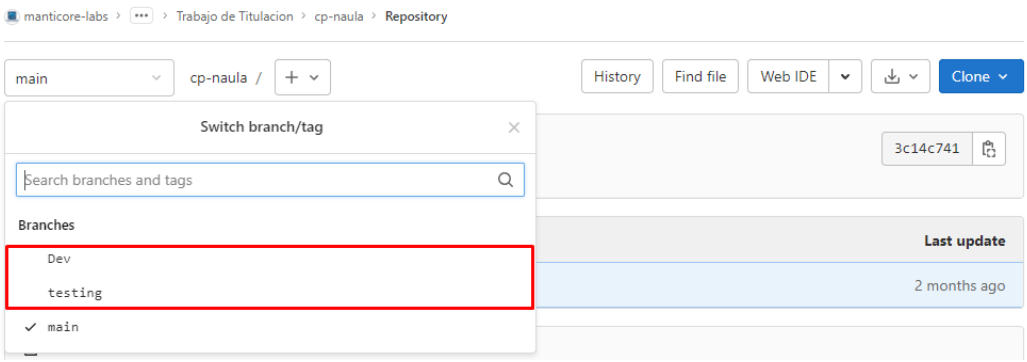

Figura 32. Ramas utilizandas dentro del repositorio

Con el fin de la integración continua, es necesario el uso de una imagen dentro de un contenedor Docker, para lo cual se hizo uso del servicio de Git Runner proporcionado por GitLab. Se puede observar el servicio Git Runner ejecutándose en el equipo de desarrollo en la Figura 33.

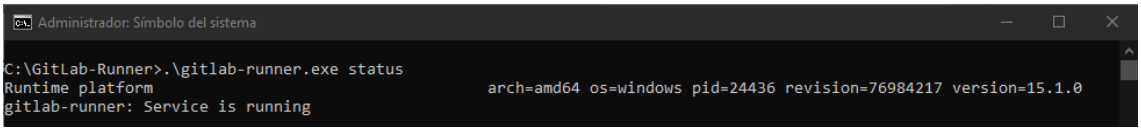

Figura 33. Servicio Git Runner en ejecución

Tal y como se mencionó, se presentó problemas en la instalación del SDK de Android por comandos, por lo que se optó por utilizar una imagen con este SDK previamente instalado. Esta imagen es circleci/android en su versión 28-node. Dentro de esta imagen se procedió con la instalación de Node Js, Ionic Framework y las librerías con dependencias de Firebase. Este proceso de instalación de dependencias y configuración mediante comandos se realiza en el archivo gitlab-ci.yml de cada una de las ramas.

Para la implementación del proyecto se consideraron dos stages, el primero para asegurar la correcta creación del archivo .apk de la aplicación cuando se realice un push

a la rama Dev. El segundo se realiza para desplegar el aplicativo mediante el servicio de App Distribution propio de Firebase, esto siempre que se realice un merge desde la rama Dev a la rama testing. En la Figura 34 y Figura 35 se pueden observar los archivos gitlab-ci.yml creados para las ramas Dev y testing.

```yaml

image: circleci/android:api-28-node

before_script:
  - sudo apt-get update
  - sudo apt-get install nodejs
  - npm --version
  - npm install
  - sudo npm install -g @angular/cli
  - sudo npm i -g @ionic/cli
  - sudo npm install -g firebase-tools@10.9.2


stages:
  - build
  - deploy


build-job:
  stage: build
  script:
    - dir
    - nodejs -v
    - cd ModuloInventarioERP
    - npm install
    - ionic build
    - ionic capacitor add android
    - ionic capacitor sync
    - cd android
    - sudo chmod +x gradlew
    - ./gradlew assembleDebug
    - cd app/build/outputs/apk/debug
    - dir
```

Figura 34. Stage de la rama Dev del archivo gitlab-ci.yml

```yaml
deploy-job:
  stage: deploy
  script:
    - dir
    - nodejs -v
    - cd ModuloInventarioERP
    - npm install
    - ionic build
    - ionic capacitor add android
    - ionic capacitor sync
    - cd android
    - sudo chmod +x gradlew
    - ./gradlew assembleDebug
    - cd app/build/outputs/apk/debug
    - dir
    #hasta aquí
    - dir
    #- cd ModuloInventarioERP/android/app/build/outputs/apk/debug
    - cd /builds/manticore-labs/trabajo/proyectos/trabajo-de-titulacion/cp-naula/ModuloInventarioERP/android/app/build/outputs/apk/debug
    - dir
    #- cambiamos el nombre a pp-debug + dia + mes + año + minutos
    - sudo mv app-debug.apk  app-debug-$(date +%d-%m-%y-%M).apk
    - dir
    #- creacmos variable global con el mismo nombre que tendrá el app
    - export nombreapp=app/build/outputs/apk/debug/app-debug-$(date +%d-%m-%y-%M).apk
    - echo $nombreapp
    #- volvemos a la carpeta android dentro del proyecto de ionic
    - cd /builds/manticore-labs/trabajo/proyectos/trabajo-de-titulacion/cp-naula/ModuloInventarioERP/android/
    - if [ -f $nombreapp ]; then firebase appdistribution:distribute $nombreapp --app 1:30196687884:android:77226124e2b22590833a29
     --token "1//0dLaENiUUoDD3CgYIARAAGA0SNwF-L9IrNCo83MJRBfEeTh1wxHRIiCSIpiUN2KCLPHfdHTyyT9XZXyGl7nhlxCHrqD8_lzg37_A"; fi
    #- cd ..
  only:
    - testing
```

Figura 35. Stage de la rama testing en el archivo gitlab-ci.yml

Finalmente, una vez se realice un push a la rama Dev o un merge a la rama testing, se ejecuta un nuevo pipeline, donde se genera un archivo .apk y se envía al servicio App Distribution. En la Figura 36 se muestran los pipelines ejecutados tras cada carga de información en las distintas ramas. En la Figura 37 se muestra los distintos archivos .apk cargados en el servicio App Distribution listos para su descarga.

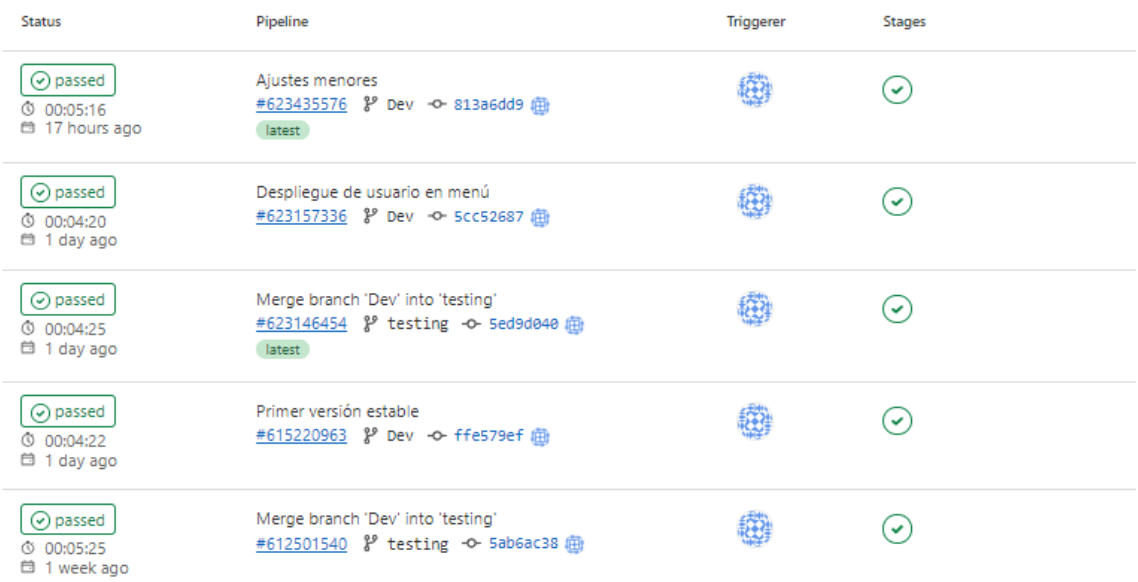

Figura 36. Historial de pipelines ejecutados

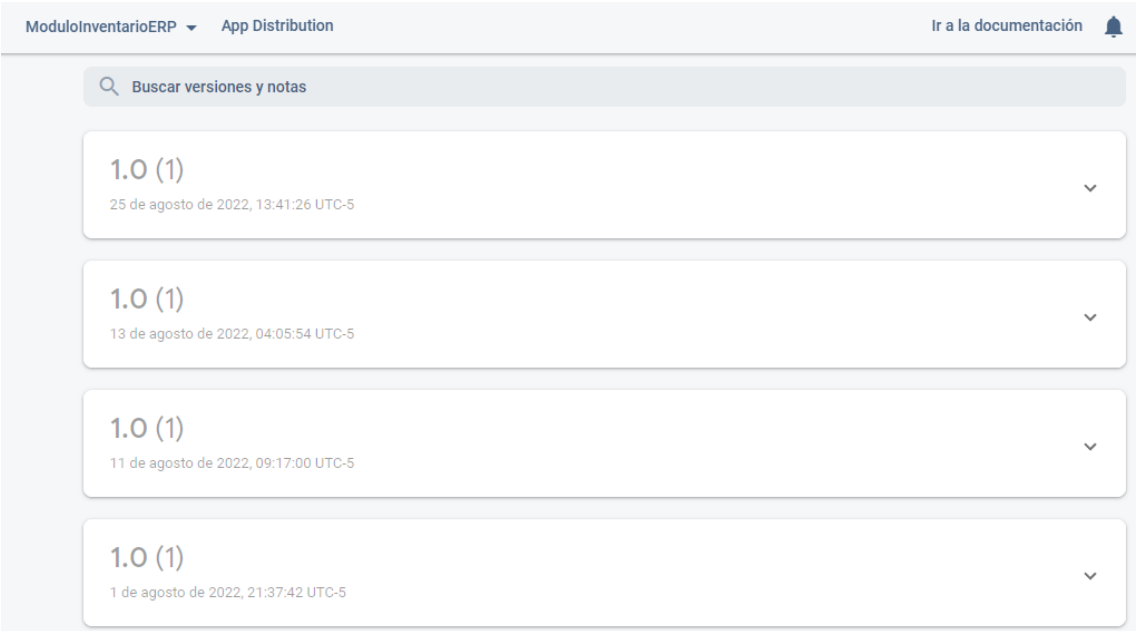

Figura 37. Versiones cargadas en el servicio de App Distribution de Firebase

Sprint Review

Se cumplió con el objetivo planteado para el sprint 0, ya que se pudieron establecer las prácticas de DevOps planteadas, las cuales fueron probadas y su éxito fue verificado.

Sprint Retrospective

No hay recomendaciones para el siguiente sprint.

Sprint 1

Objetivo: Implementar la gestión del ingreso de usuarios y el manejo del inventario del aplicativo.

Sprint Planning

En el transcurso de este sprint, se realizará la implementación del Login al aplicativo mediante el uso de Firebase Authentication y la gestión del inventario con el almacenamiento de datos en la base de datos no relacional de Firebase, Cloud Firestore.

Daily meeting

Durante la ejecución del sprint se presentaron las siguientes dificultades.

Dificultades encontradas	**Solución**
Establecer conexión entre el aplicativo y los servicios de Firebase.	Vincular el aplicativo con un nuevo proyecto de Firebase mediante el uso de su ID,
Error en la recuperación de datos de Firebase Authentication.	Utilizar el método Subscribe para recuperar información desde firebase

Restablecer una pantalla cada vez que navegamos hacia una nueva.	Implementación de funciones propias de Ionic para el manejo del ciclo de vida de cada interfaz.
No existe el despliegue de menús contextuales para la selección de una entre varias opciones en Ionic.	Investigación sobre componentes de Ionic y uso del componente Select propio del framework.
Obtención de campos específicos de documentos almacenados en Firestore.	Obtención de documentos completos y desglose de estos mediante código y variables auxiliares.
Creación de validaciones individuales para cada campo de los formularios.	Uso de Validaciones propias de los formularios proporcionados por el Framework.

Tabla 5.Resumen del daily meeting sprint 1

Codificación

Como primer paso en la codificación del presente sprint se debe crear y configurar un nuevo proyecto dentro de Firebase, esto con el fin de poder vincularlo al proyecto de Ionic creado para la construcción de nuestro aplicativo. Una vez creados los dos proyectos, procedemos con la vinculación de estos haciendo uso del ID de la aplicación web. Para esto se debe copiar la configuración de Firebase proporcionada por la propia plataforma en el enviroment del proyecto de Ionic.

En la Figura 38 y Figura 39 se puede observar la información del proyecto creado en Firebase, y en la Figura 40 la configuración de Firebase dentro del proyecto de Ionic.

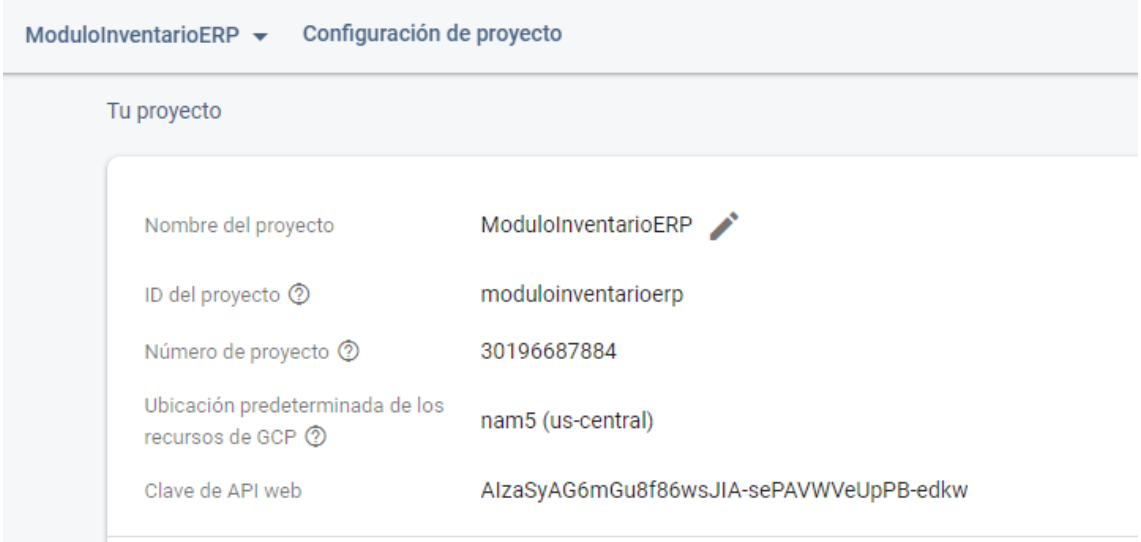

Figura 38. Vista general del proyecto de Firebase creado

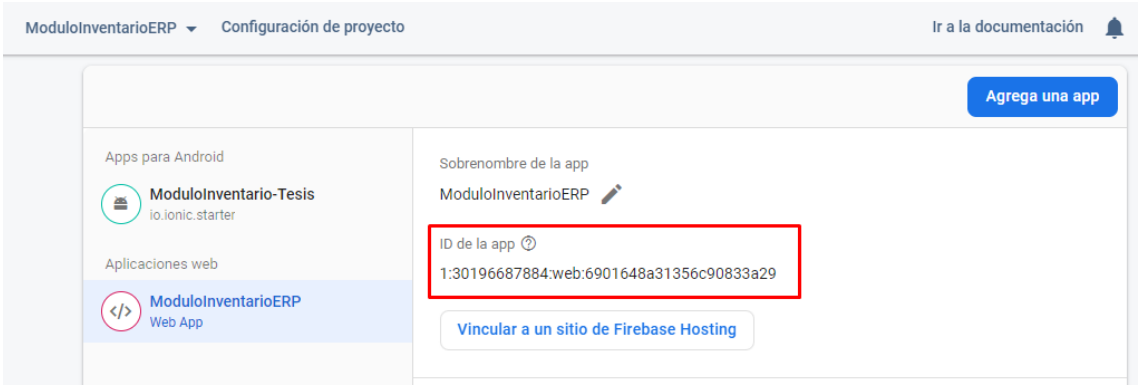

Figura 39. ID de la aplicación en Firebase

```
export const environment = {
  production: false,

  firebaseConfig: {
    apiKey: 'AIzaSyD7Tll8-oJof1FY9ocxHllxfWM8a-rHRiA',
    authDomain: 'moduloinventarioerp.firebaseapp.com',
    projectId: 'moduloinventarioerp',
    storageBucket: 'moduloinventarioerp.appspot.com',
    messagingSenderId: '30196687884',
    appId: '1:30196687884:web:6901648a31356c90833a29',
    measurementId: 'G-BNTP8H99ZK'
  }
};
```

Figura 40. Configuración de Firebase dentro del proyecto de Ionic

Se procede con la creación de la interfaz gráfica correspondiente al inicio de sesión de un usuario para el manejo de la plataforma. La validación del ingreso para usuarios registrados se manejó con el servicio de Firebase Authentication mediante la librería AngularFireAuth propia de Firebase. Adicionalmente se implentó un control para advertir que el usuario o contraseña no corresponden a un usuario válido mediante un mensaje pop-up. En la Figura 41 se muestra la pantalla de inicio de sesión diseñada, junto con el mensaje en caso de error al iniciar sesión.

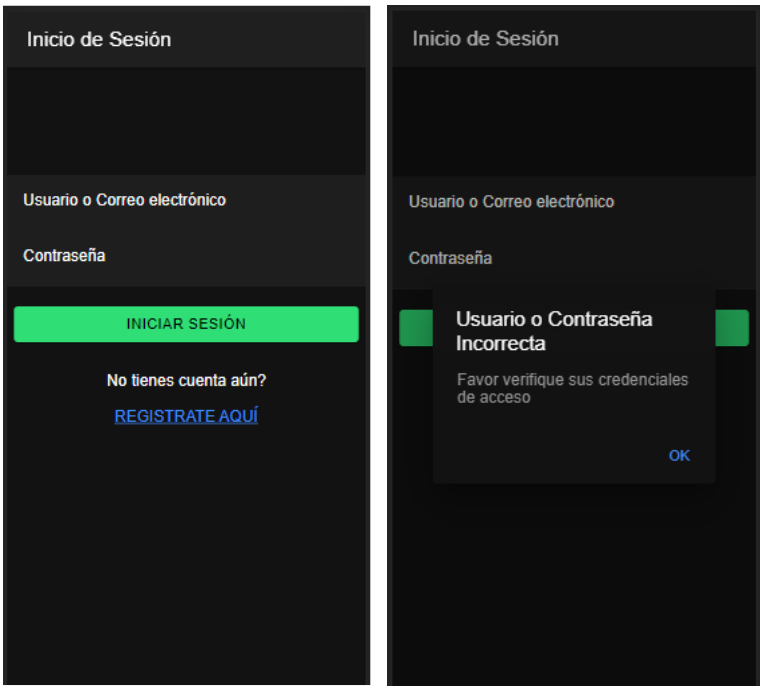

Figura 41. Pantalla de inicio de sesión de la aplicación

En la Figura 42 se muestran los usuarios creados directamente en el servicio de Authentication de Firebase. Hasta este punto del sprint se completa la primera historia de usuario (US02) designada para el sprint 1.

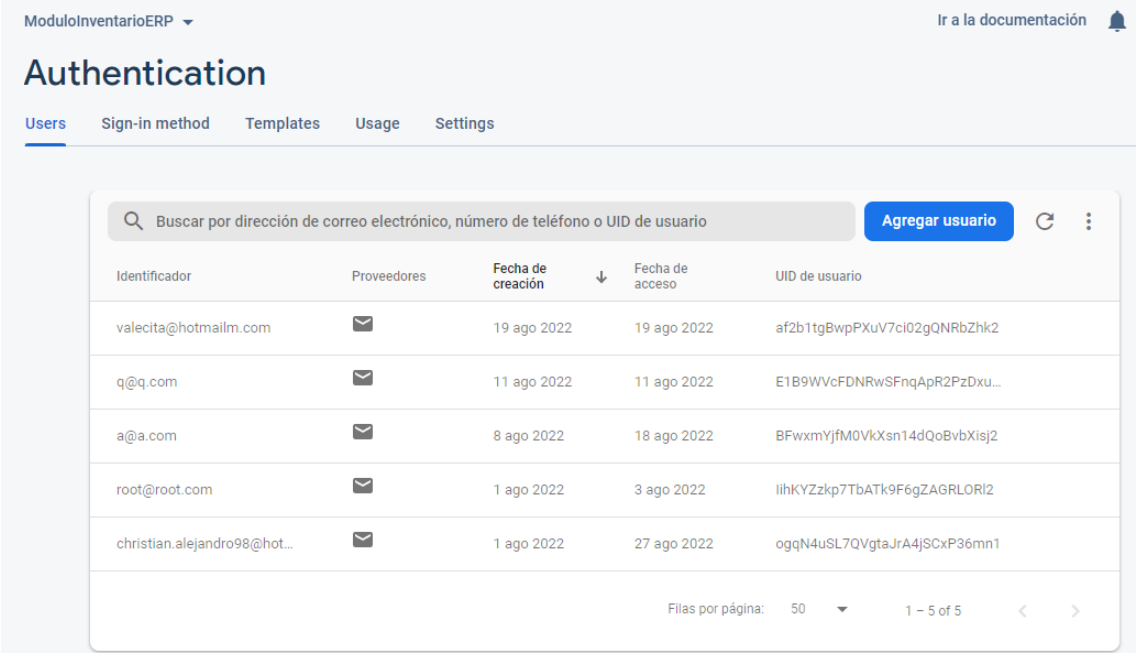

Figura 42. Usuarios registrados en Firebase Authentication

Comenzamos con la segunda historia de usuario contemplada en el presente sprint, para lo cual se crea la interfaz para el registro de un nuevo usuario no registrado. La opción de crear el usuario se habilita únicamente cuando se cumplan las condiciones de cumplimiento del formulario, y cuando se cumple el registro de manera correcta se guardará el nuevo usuario en la base de datos de Cloud Firestore y se registrará en el servicio de Firebase Authentication desplegará un mensaje indicando que el nuevo usuario ha sido creado. En la Figura 43 se muestra la pantalla de registro de usuario. Hasta este punto del sprint se completa la segunda historia de usuario (US01) designada para el sprint 1.

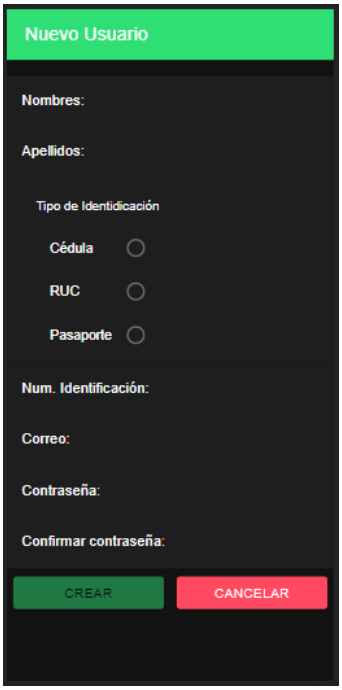

Figura 43. Pantalla de creación de nuevo usuario.

Continuamos con la creación de las interfaces para el manejo de inventario, donde se recuperarán los productos previamente creados desde la base de datos de Cloud Firestore para desplegarlos mediante tarjetas. En la Figura 44 se muestra la pantalla creada para la visualización del inventario.

Figura 44. Pantalla de visualización de productos del inventario

Dentro de la pantalla del inventario podemos crear un nuevo producto que no se encuentra registrado, adicionalmente podemos editar un producto existente, visualizar los movimientos de cada producto y finalmente eliminarlo. En la Figura 45 podemos observar las pantallas creadas para la creación, edición y visualización de movimientos de un producto, y en la Figura 46 la confirmación de que se desea eliminar un producto existente. Hasta este punto del sprint se completa la tercera y última historia de usuario (US03) designada para el sprint 1.

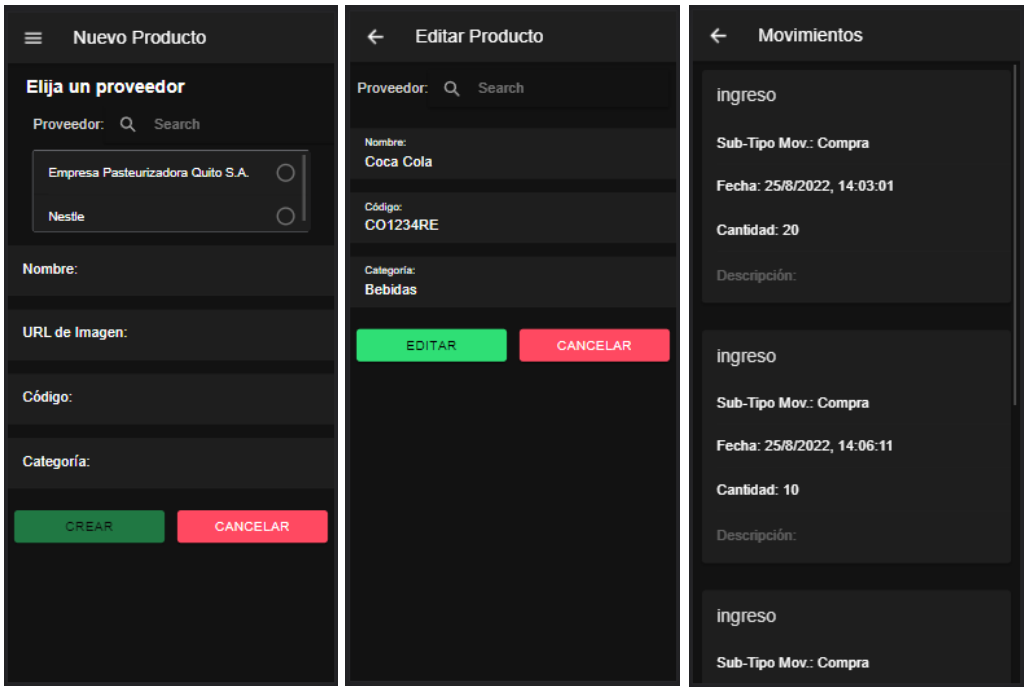

Figura 45. Pantallas creadas para la creación, edición y visualización de movimientos de un producto

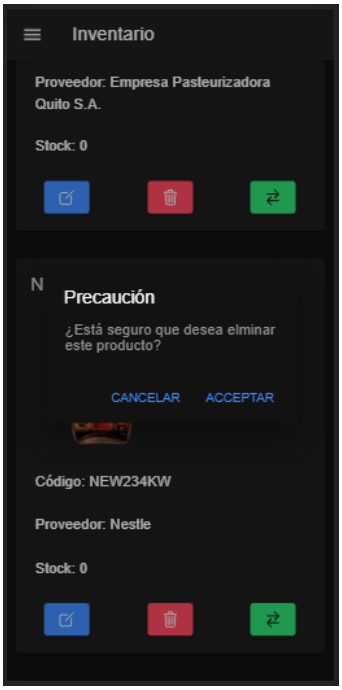

Figura 46. Mensaje de confirmación al eliminar un producto

Sprint Review

Se cumplió de manera satisfactoria con el objetivo de sprint planteado. El incremento en el aplicativo fue validado según los criterios de aceptación especificados en las historias de usuario que pueden ser revisadas en el Anexo III.

HU	Puntos Estimados	Puntos Restantes	Observaciones
US02	3	0	Se consultó sobre el uso de Firebase Authentication y cómo consumirlo desde el aplicativo en Ionic
US01	3	0	Ninguna
US03	10	0	Ninguna
Total	**16**	**0**	

Tabla 6. Revisión avances Sprint 1

Sprint Retrospective

El ingreso y creación de usuarios, además de la gestión de los productos del inventario funciona de manera satisfactoria y no hubo retrasos en el cumplimiento de las funcionalidades planificadas para el sprint 1.

El avance obtenido en el presente sprint se visualiza en la Figura 47.

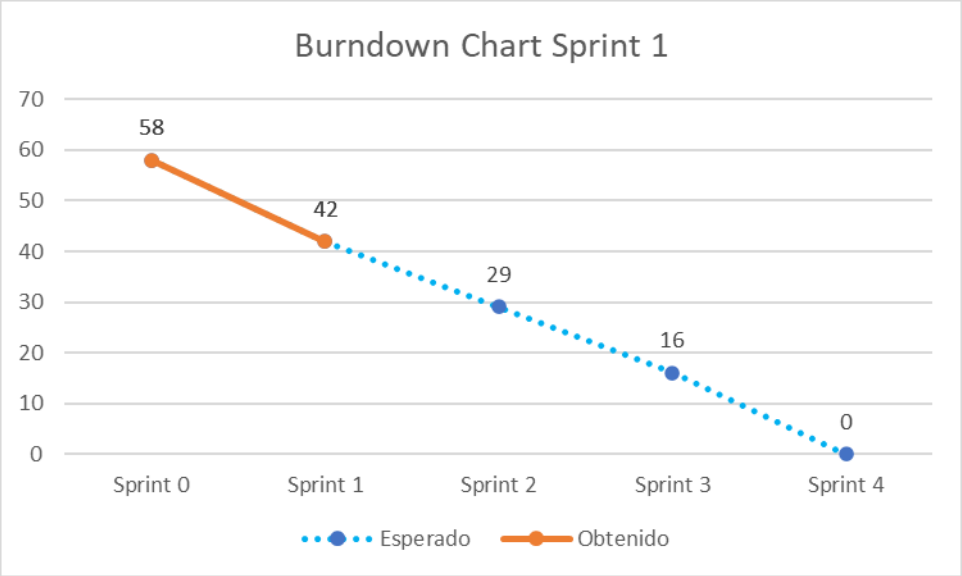

Figura 47. Burndown Chart Sprint 1

Sprint 2

Objetivo: Implementar la gestión de proveedores de productos en el aplicativo.

Sprint Planning

En el transcurso de este sprint, se realizará la implementación de la gestión de Proveedores de productos, lo mismo que abarca el despliego, creación, edición y eliminación de estos en el aplicativo. Para esto se hará uso de la base de datos Cloud Firestore.

Daily meeting

Durante la ejecución del sprint se presentaron las siguientes dificultades.

Dificultades encontradas	**Solución**
Restablecer una pantalla cada vez que navegamos hacia una nueva.	Implementación de funciones propias de Ionic para el manejo del ciclo de vida de cada interfaz.

Creación de menú de navegación entre pantallas.	Creación de un componente propio de tipo menú para la navegación e interacción entre pantallas del aplicativo

Tabla 7. Resumen del daily meeting sprint 2

Codificación

En la Figura 48 se observa la pantalla creada para el despliegue de los proveedores almacenados en la base de datos Cloud Firestore. En esta pantalla es posible realizar búsqueda de proveedores específicos por el campo Razón Social.

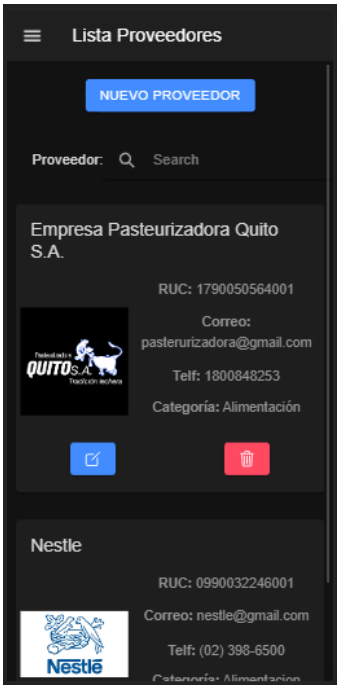

Figura 48. Pantalla lista de proveedores

Dentro de la pantalla del listado de proveedores tenemos botones que nos llevan a la creación de un nuevo proveedor, la edición de un proveedor existente y finalmente la eliminación de un proveedor luego de aceptarlo en un mensaje de confirmación. En la Figura 49 se muestran las pantallas de creación y edición de un proveedor, y en la Figura 50 el mensaje de confirmación al eliminar un proveedor. Hasta este punto del sprint se completa la única historia de usuario (US06) designada para el sprint 2.

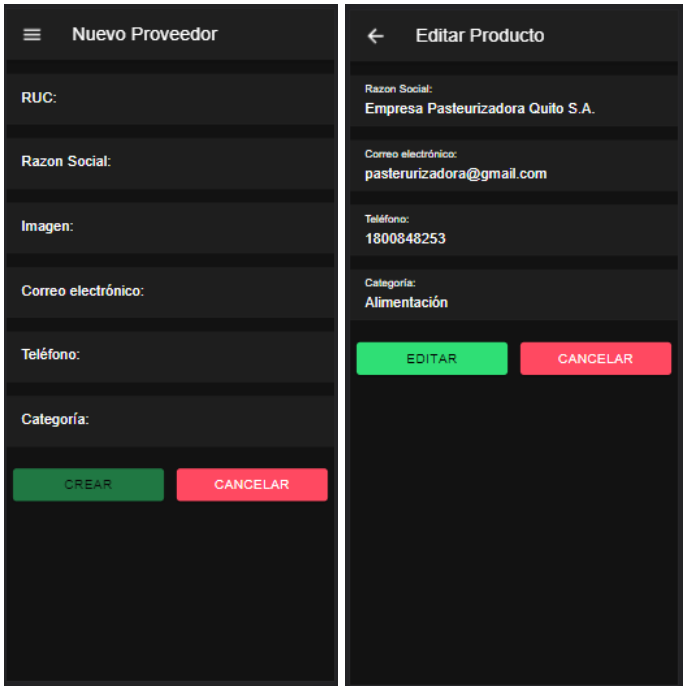

Figura 49. Pantallas de creación y edición de proveedores

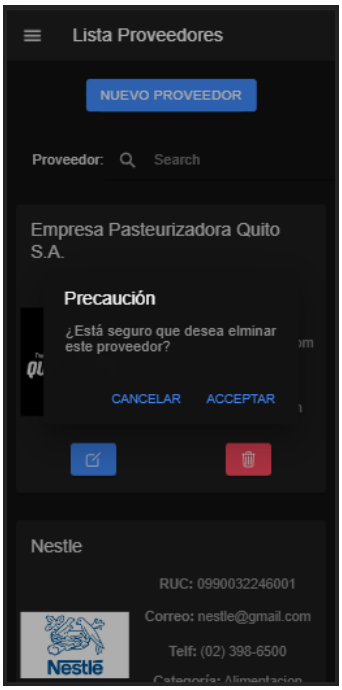

Figura 50. Mensaje de confirmación al eliminar un proveedor

Antes de finalizar el presente sprint, se creó el menú lateral utilizado para la navegación entre las pantallas principales del aplicativo. Dentro del menú se contemplaron 8 opciones elegibles y un botón para el cierre de sesión del usuario actual del aplicativo. Se puede observar el menú lateral en la Figura 51.

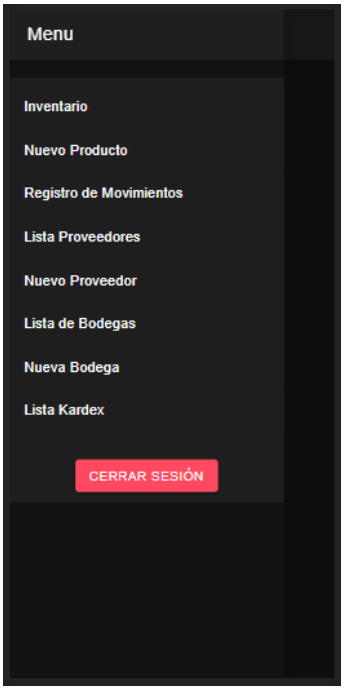

Figura 51. Menú de navegación de la aplicación

Sprint Review

Se cumplió de manera satisfactoria con el objetivo de sprint planteado. El incremento en el aplicativo fue validado según los criterios de aceptación especificados en las historias de usuario que pueden ser revisadas en el Anexo III.

HU	Puntos Estimados	Puntos Restantes	Observaciones
US06	13	0	Ninguna
Total	**13**	**0**	

Tabla 8. Revisión avances Sprint 2

Sprint Retrospective

La gestión de proveedores de productos funciona de manera satisfactoria y se cumplió con las funcionalidades y tareas planificadas para el sprint 2. Se hizo uso de código fuente previamente utilizado en el sprint 1.
El avance obtenido en el presente sprint se visualiza en la Figura 52.

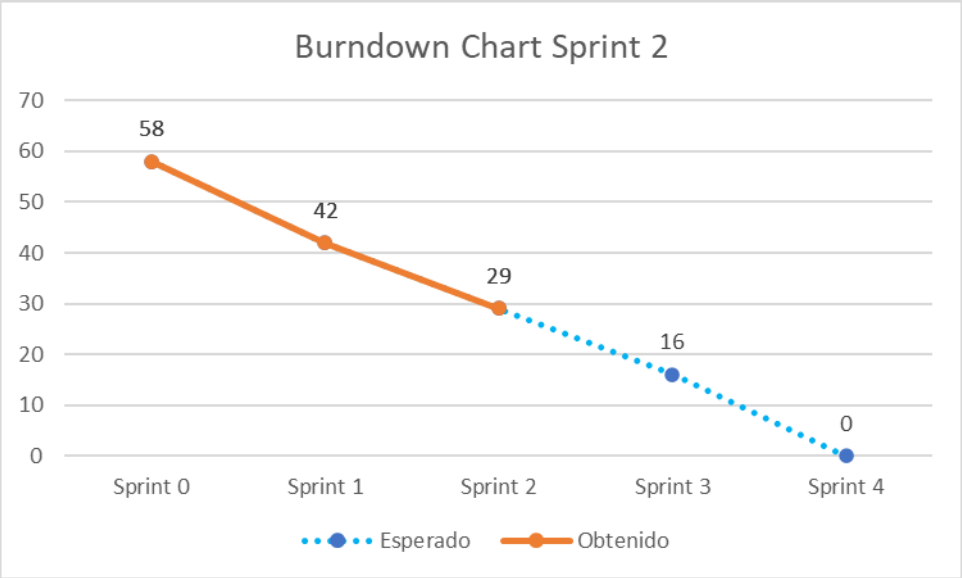

Figura 52. Burndown Chart Sprint 2

Sprint 3

Objetivo: Implementar la gestión de bodegas para el almacenamiento de productos en el aplicativo.

Sprint Planning

En el transcurso de este sprint, se realizará la implementación de la gestión de bodegas para el almacenamiento de productos, lo mismo que abarca el despliegue, creación, edición y eliminación de estas en el aplicativo. Para esto se hará uso de la base de datos Cloud Firestore.

Daily meeting

Durante la ejecución del sprint se presentaron las siguientes dificultades.

Dificultades encontradas	**Solución**
Manejo asíncrono de consultas a Cloud Firestore por Ionic.	Concatenación de consultas a la base de datos mediante el uso de funciones para garantizar un orden en las consultas.

Almacenamiento de colecciones complejas	Creación de colecciones dentro de colecciones para el manejo de datos en la base.

Tabla 9. Resumen del daily meeting sprint 3

Codificación

Se creó una pantalla para el despliegue de todas las bodegas almacenadas en la base de datos Cloud Firestore. Además, se añadió la opción de realizar búsquedas de bodegas específicas mediante el nombre de estas. En la Figura 53 se muestra la pantalla para el listado de bodegas.

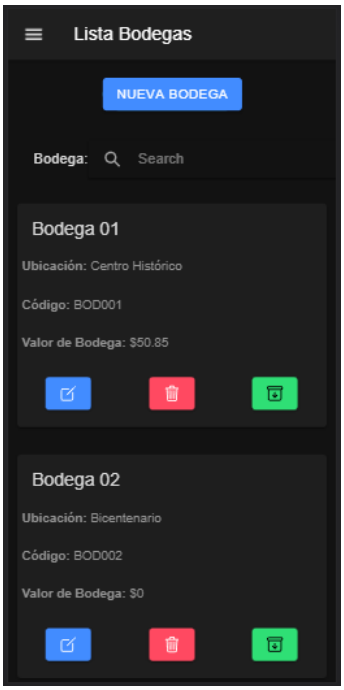

Figura 53. Pantalla de listado de bodegas

Dentro de la pantalla del listado de bodegas se creó un conjunto de opciones para navegar a las distintas pantallas referentes a la gestión de bodegas, las cuales abarcan la creación de una nueva bodega inexistente, editar una bodega existente, listar los productos almacenados dentro de una bodega y finalmente eliminar una bodega específica. En la Figura 54 se puede observar las pantallas referentes a la recreación y edición de una bodega.

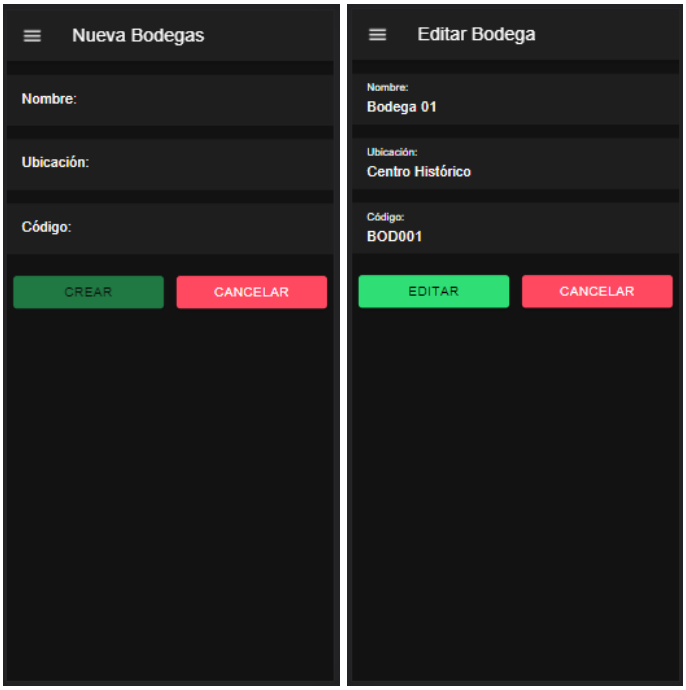

Figura 54. Pantallas de creación y edición de bodegas

Para la eliminación de una bodega se implementó un mensaje de confirmación. Esto se puede observar en la Figura 55.

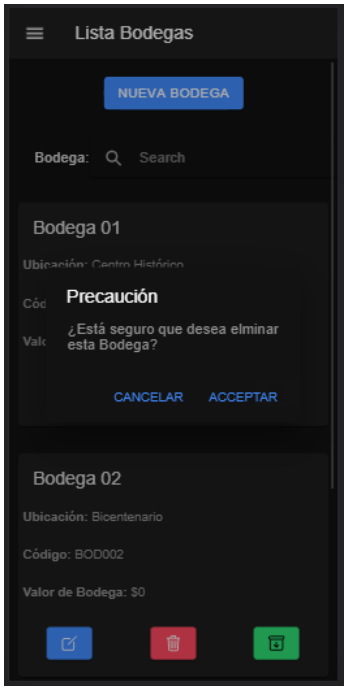

Figura 55. Confirmación al eliminar una bodega

En la Figura 56 se puede observar la pantalla creada para listar todos los productos que se encuentran almacenados en una bodega específica. Hasta este punto del sprint se completa la única historia de usuario (US07) designada para el sprint 3.

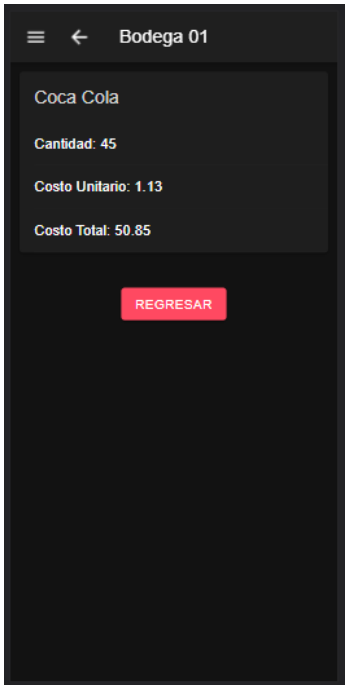

Figura 56. Listado de productos almacenados en la Bodega 01

Sprint Review

Se cumplió de manera satisfactoria con el objetivo de sprint planteado. El incremento en el aplicativo fue validado según los criterios de aceptación especificados en las historias de usuario que pueden ser revisadas en el Anexo III.

HU	Puntos Estimados	Puntos Restantes	Observaciones
US07	13	0	Ninguna
Total	**13**	**0**	

Tabla 10. Revisión avances Sprint 3

Sprint Retrospective

La gestión de bodegas funciona de manera satisfactoria y se cumplió con las funcionalidades y tareas planificadas para el sprint 3. Se hizo uso de código fuente previamente utilizado en el sprint 1 y 2.

El avance obtenido en el presente sprint se visualiza en la Figura 57.

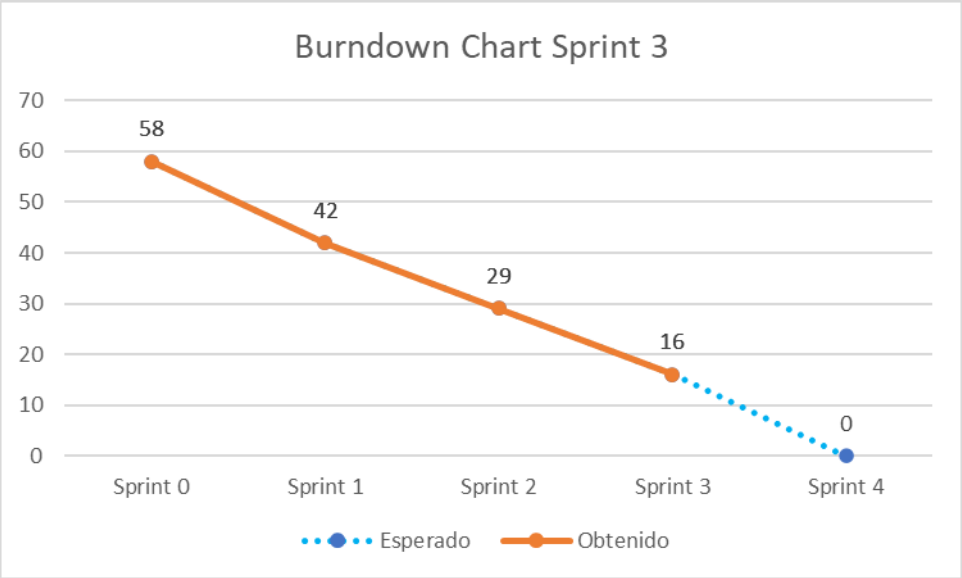

Figura 57. Burndown Chart Sprint 3

Sprint 4

Objetivo: Implementar la gestión de movimientos de productos desde y hacia las bodegas en el aplicativo.

Sprint Planning

En el transcurso de este sprint, se realizará la implementación de la gestión de movimientos desde y hacia las bodegas. Adicionalmente se implementó la revisión del Kardex y una búsqueda de este mediante diversos filtros. Para esto se hará uso de la base de datos Cloud Firestore.

Daily meeting

Durante la ejecución del sprint se presentaron las siguientes dificultades.

Dificultades encontradas	**Solución**
Manejo asíncrono de consultas a Cloud Firestore por Ionic.	Concatenación de consultas a la base de datos mediante el uso de funciones para garantizar un orden en las consultas.

Tabla 11. Resumen del daily meeting sprint 4

Codificación

Se creó la pantalla para el registro del movimiento de un producto. En este caso se debe escoger el producto a mover y la bodega con la que va a interactuar, adicionalmente se debe escoger el tipo de movimiento que se va a realizar, los cuales pueden ser de ingreso a la bodega o de egreso de la bodega. En la Figura 58 se muestra la pantalla para registrar un nuevo movimiento.

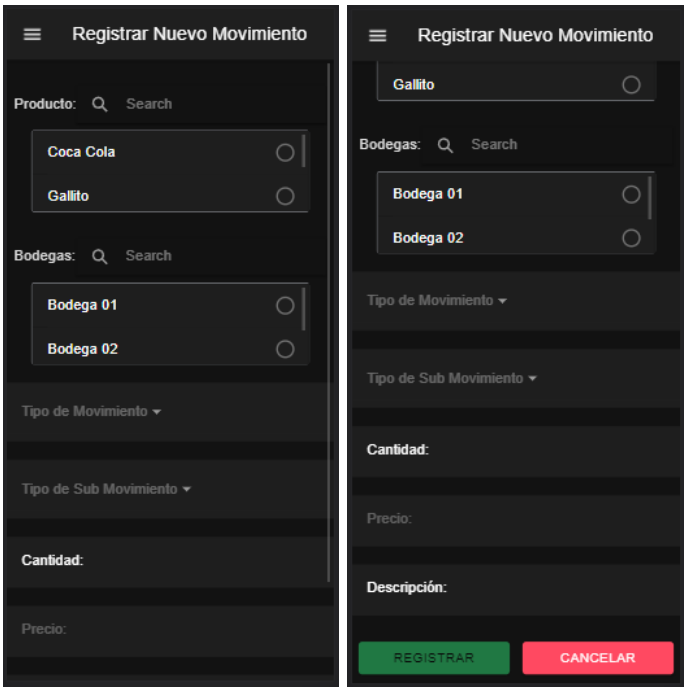

Figura 58. Pantalla para registrar nuevo movimiento

En caso de realizar por primera vez el ingreso de un producto a una bodega se deberá calcular el valor total del producto, por otro lado, si se realiza un ingreso y el producto ya se encuentra almacenado en la bodega se debe recalcular el valor unitario del producto y a su vez el valor total tanto del producto como de la bodega.

Adicionalmente se creó la pantalla de visualización del Kardex, los cuales se muestra en formato de tarjetas tal y como se puede visualizar en la Figura 59. En la pantalla del listado de Kardex es posible realizar un filtrado de movimientos específicos mediante una búsqueda de distintos campos que son por el nombre del producto, por la fecha del movimiento, por la bodega con la que interactuó el producto y finalmente por el tipo de movimiento. En la Figura 60 se muestran los tipos de campos habilitados para la búsqueda. Hasta este punto del sprint se completa la única historia de usuario (US05) designada para el sprint 4.

Figura 59. Pantalla de listado de Kardex

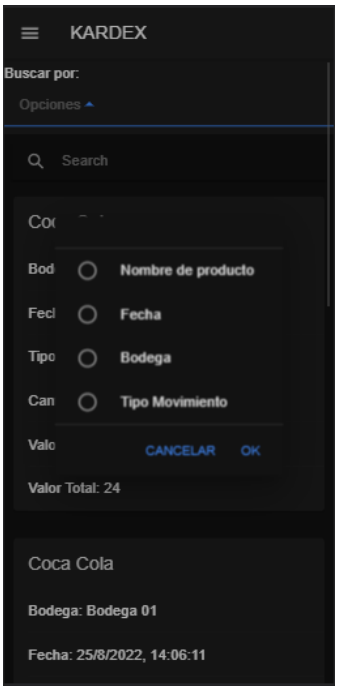

Figura 60. Opciones de filtrado de Kardex

Finalmente, para el despliegue de la versión estable del aplicativo se realiza un envío de datos mediante un commit y push hacia la rama Dev del repositorio para posteriormente realizar un merge desde esta rama a la rama de testing. Con cada uno de los envíos de datos mencionados se ejecuta los pipelines descritos en el sprint 0. En la Figura 61 se muestran los pipelines ejecutados con éxito.

Figura 61. Pipelines ejecutados correctamente

Sprint Review

Se cumplió de manera satisfactoria con el objetivo de sprint planteado. El incremento en el aplicativo fue validado según los criterios de aceptación especificados en las historias de usuario que pueden ser revisadas en el Anexo III.

HU	Puntos Estimados	Puntos Restantes	Observaciones
US05	16	0	Ninguna
Total	**16**	**0**	

Tabla 12. Revisión avances Sprint 4

Sprint Retrospective

La gestión de movimientos de productos en bodegas funciona de manera satisfactoria y se cumplió con las funcionalidades y tareas planificadas para el sprint 4. Se hizo uso de código fuente previamente utilizado en el sprint 1 ,2 y 3.
El avance obtenido en el presente sprint se visualiza en la Figura 62.

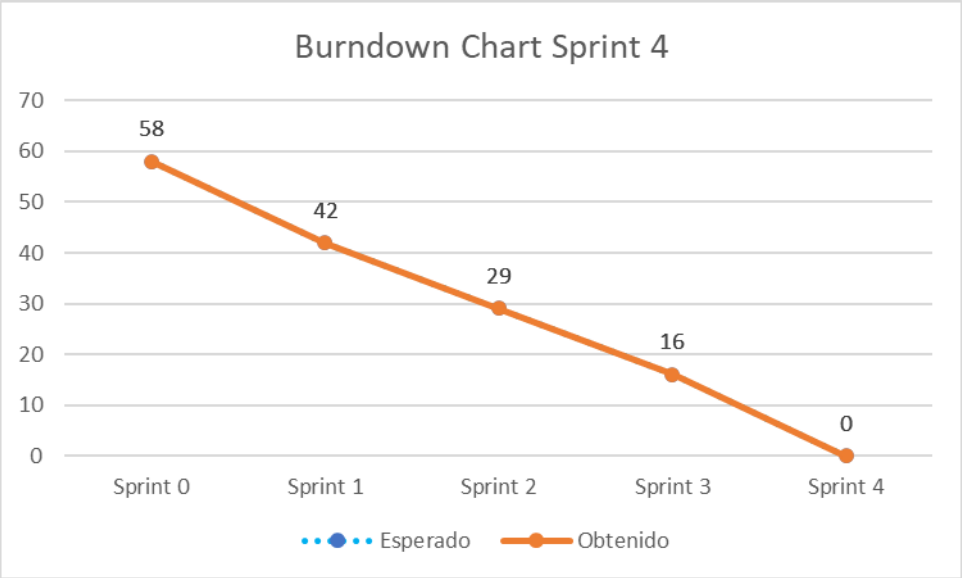

Figura 62. Burndown Chart Sprint 4

Una vez finalizado el último sprint (sprint 4) podemos observar cómo se han cumplido satisfactoriamente todas las historias de usuario definidas para el aplicativo construido. En la Figura 62 se pudo observar que no hubo ninguna desviación a lo largo de los sprints, aspecto corroborado en la Tabla 13 que muestra una comparativa entre los puntos esperados y los obtenidos.

Sprints	Esperado	Obtenido
0	0	0
1	16	16
2	13	13
3	13	13
4	16	16
Total	58	58

Tabla 13. Resultados Esperados vs Resultados Obtenidos

Si bien se presentaron ligeros contratiempos a lo largo de los sprints especialmente debido a la falta de conocimiento del Framework Ionic, se pudo solventar estos particulares mediante la investigación en la documentación oficial, logrando así cumplir con los objetivos planteados y lograr el éxito en el desarrollo de cada uno de los sprints planteados.

3 PRUEBAS Y RESULTADOS

3.1 Pruebas de construcción del aplicativo

Ejecución correcta de pipelines y flujos de DevOps

Con el fin de evidenciar el resultado de las prácticas de DevOps usadas en la plataforma de GitLab se realizó un envío de datos mediante un commit y push a la rama Dev (origin) desde el equipo en el cual se realizó el desarrollo. Una vez realizado el envío de datos observamos la ejecución correcta del stage build del pipeline correspondiente a la rama Dev. En la Figura 63 se visualiza el push realizado a la rama Dev, y en la Figura 64 la ejecución con éxito del pipeline en la plataforma de GitLab.

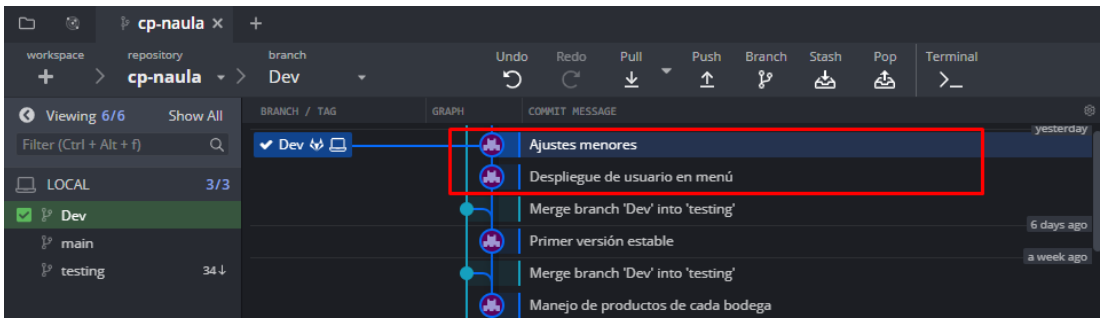

Figura 63. Push desde el equipo a la rama Dev del repositorio

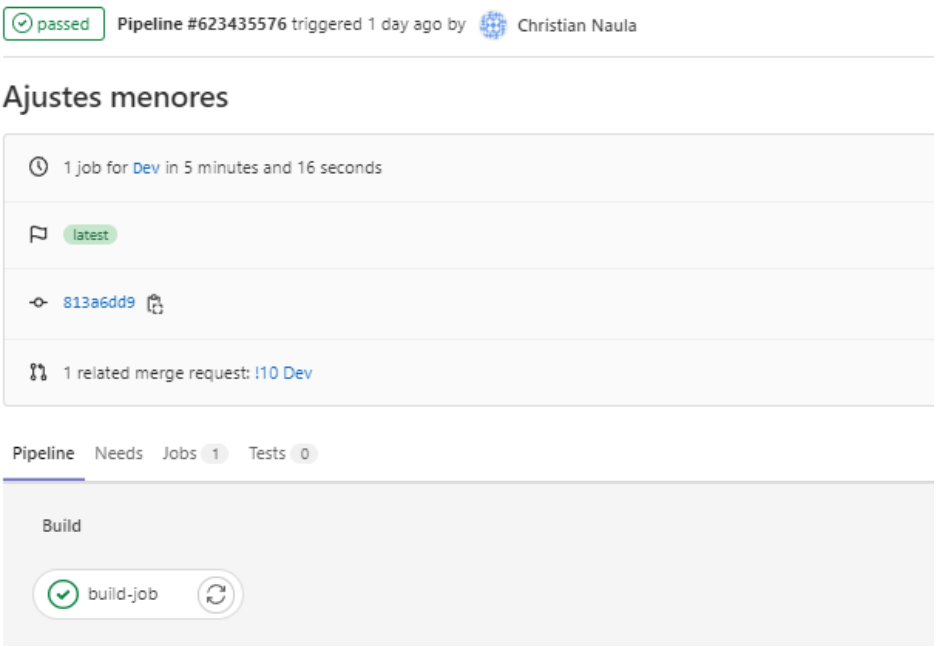

Figura 64. Pipeline ejecutado con éxito en GitLab (Dev)

En la Figura 65 podemos visualizar el buil-job detallado donde al final de su ejecución nos muestra el archivo .apk generado con éxito, el cual es el objetivo del stage en la rama Dev. Adicionalmente se muestra el mensaje que indica que el job se completó con satisfacción.

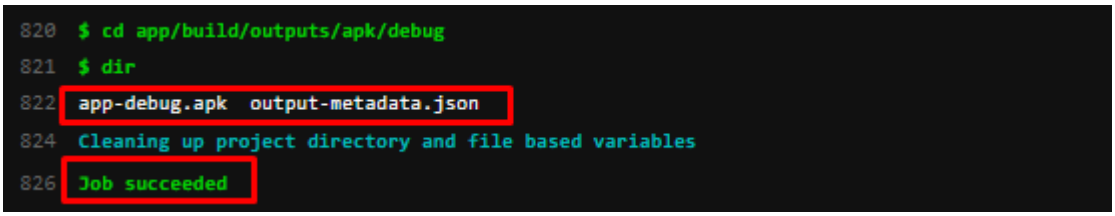

Figura 65. Build-job detallado (Dev)

Para la stage correspondiente a la rama de testing (deploy) se realizó un merge desde la rama Dev. En la Figura 66 se muestra la ejecución correcta del pipeline generado al momento del envío de datos a la rama testing, adicionalmente en la Figura 67 se observa el build-job detallado del pipeline, donde se indica el envío correcto del archivo .apk al servicio de App Distribution de Firebase y el mensaje que indica que el job se completó con satisfacción.

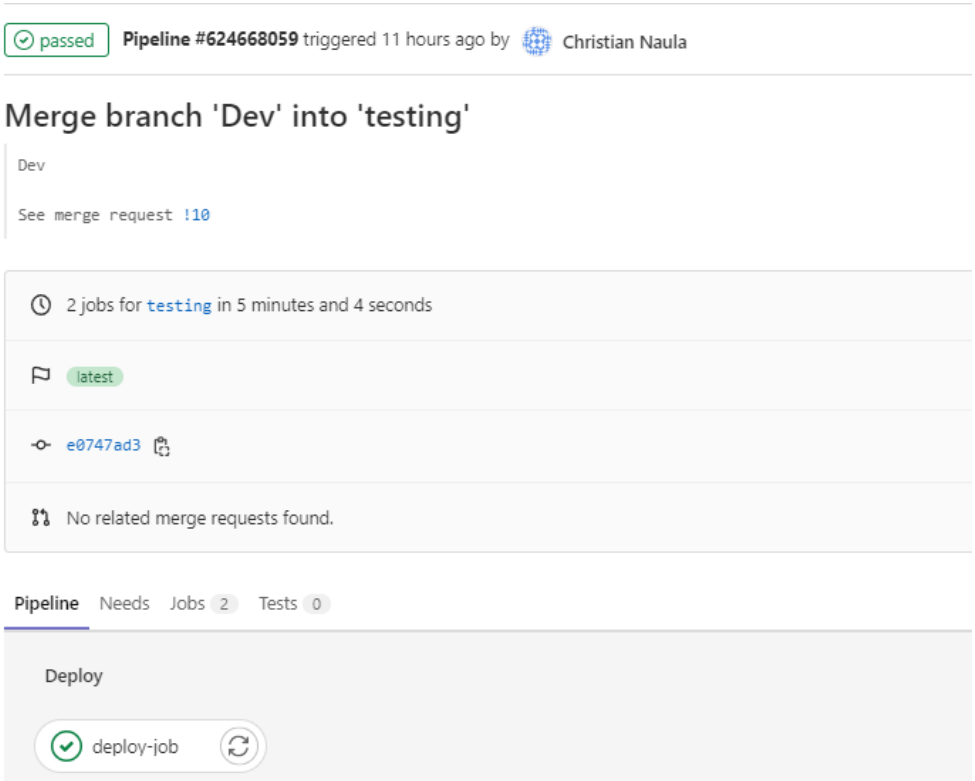

Figura 66. Pipeline ejecutado con éxito en GitaLab (testing)

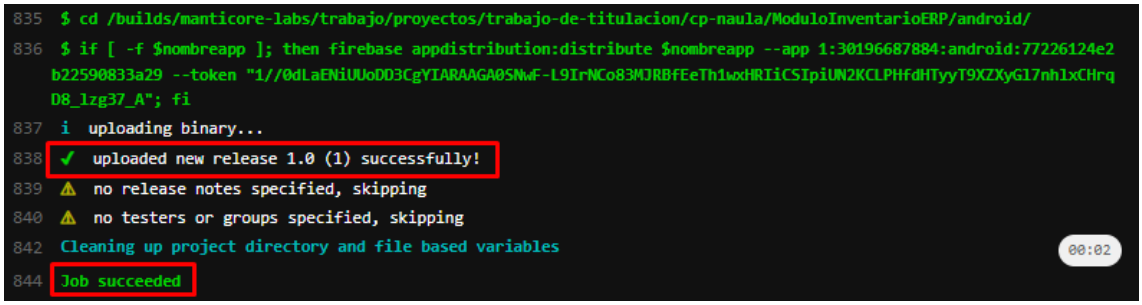

Figura 67. Build-job detallado (testing)

Con la ejecución correcta de los pipelines correspondientes a las prácticas de DevOps, medimos la cantidad de pipelines ejecutados correctamente en comparación a aquellos erróneos y los cancelados. Para esta medición tomaremos los últimos 60 pipelines ejecutados para la etapa “build” y los 7 existentes para la etapa “deploy”. En la Tabla 14 se visualizan los resultados obtenidos.

Pipelines			
Etapa	**Completados**	**Fallidos**	**Cancelados**
Build	34	21	5
Deploy	7	3	3
Total	41	24	8

Tabla 14. Resultado ejecución pipelines

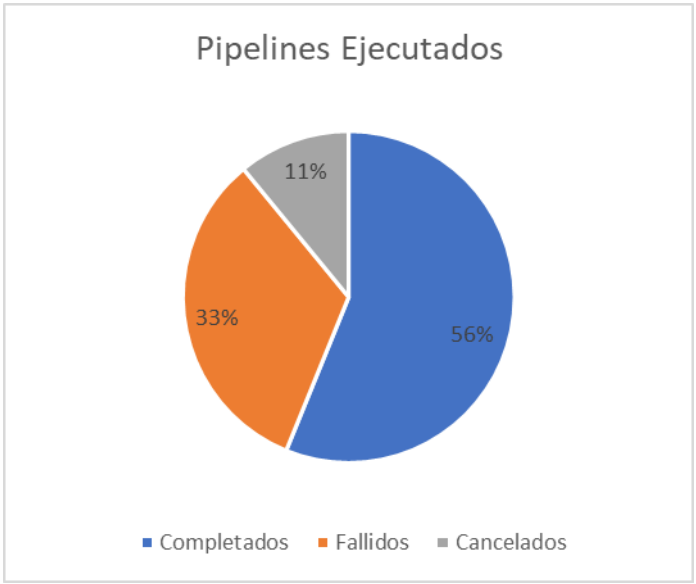

Figura 68. Resultado ejecución pipelines

En este caso, los pipelines fallidos en gran medido son debido a un error en la configuración del archivo YML, debido a que si un comando no puede ser ejecutado o

genera algún error toda la ejecución del pipeline falla. Otro motivo por el cual hubo fallos es debido a que al momento de realizar el envío de datos a la rama “Dev” la aplicación de Docker no estaba en ejecución, ocasionando que no se pueda levantar el contenedor con la imagen y generando un fallo en la ejecución.

Ventajas del uso de Flujos de DevOps.

Con la implementación de los Flujos de DevOps en el presente proyecto hemos identificado las ventajas que nos ofrecen estas prácticas durante el ciclo de vida de desarrollo. La principal diferencia de hacer uso de estos flujos, en comparación a una implementación que no los usa es la automatización de actividades que reducen la utilización de recursos y tiempo. Esto debido a que los flujos implementados de construcción (build) y despliegue (deploy) del aplicativo se configurar para realizarse de manera automática, evitando así el trabajo que corresponde el construir el aplicativo y cargarlo manualmente en un servicio de distribución, optimizando así todo el proceso de construcción durante el ciclo de vida del aplicativo.

En la Tabla 15 se muestra un resumen de las tareas necesarias para la fase de implementación del módulo de Inventario desarrollado, donde utilizando los flujos de DevOps previamente definidos fue necesaria únicamente la elaboración manual de 3 tareas. Sin embargo, en caso de afrontar una fase de implementación sin el uso de estos flujos de DevOps, sería necesaria la ejecución manual de todas las tareas mencionadas (7 tareas) ocasionando un mayor esfuerzo en tiempo y recursos.

TAREAS	CON FLUJOS DevOps	SIN FLUJOS DevOps
Construcción de código fuente	Manual	Manual
Carga de archivos al repositorio local	Manual	Manual
Carga de archivos al repositorio remoto	Manual	Manual
Construcción de archivo APK en la etapa de construcción	Automático	Manual
Construcción de archivo APK en la etapa de despliegue	Automático	Manual

Asignación de nombre diferente en cada construcción del APK	Automático	Manual
Carga del archivo APK en el servicio App Distribution	Automático	Manual

Tabla 15. Resumen de tareas con flujos DevOps vs sin flujos DevOps

Carga de archivos en Firebase App Distribution

La carga de las distintas versiones del aplicativo al servicio App Distribution proporcionado por Firebase se completó con éxito, esta debido a que en cada envío de información hacia la rama testing del repositorio del proyecto se ejecutaba un pipeline cuyo objetivo es la carga del archivo .apk en el servicio mencionado. En la Figura 69 se observan las versiones cargadas (9) en App Distribution a lo largo de toda la fase de implementación y desarrollo del aplicativo.

Una de las ventajas de usar este servicio de Firebase es que nos permite descargar el archivo .apk de cada una de las versiones cargadas, pudiendo así trabajar con cualquier versión que se desee y no necesariamente con la última cargada. En la Figura 70 se muestra con detalle las opciones que tenemos en cada una de las versiones, incluidas la opción de descarga.

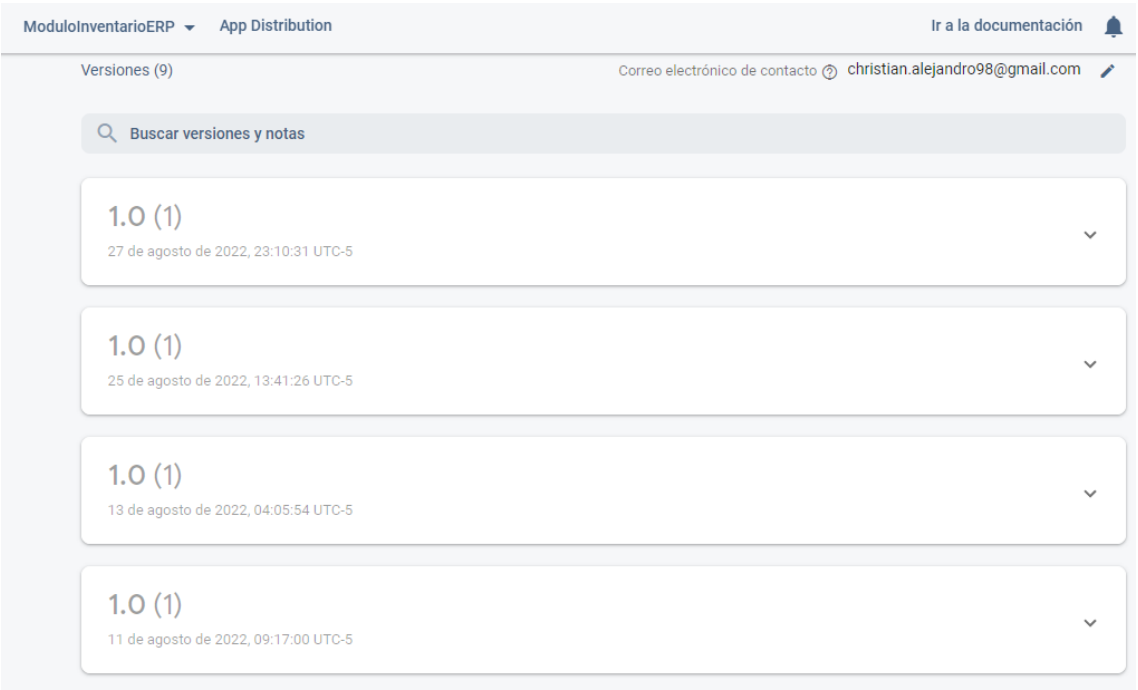

Figura 69. Versiones cargadas en Firebase App Distribution

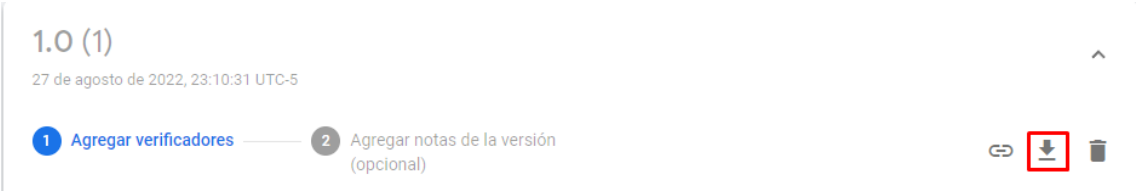

Figura 70. Opciones de cada versión cargada en App Distribution

Aplicación Móvil desarrollada

La aplicación, la cual abarca el módulo de inventario para un ERP funciona de manera correcta tanto en simuladores móviles ejecutados desde un ordenado, cómo en un dispositivo móvil (celular) Android real. La última versión disponible en el servicio de App Distribution cuenta con todas las funcionalidades previamente definidas en las historias de usuario y validadas a lo largo de los 4 sprints requeridos para su construcción. En la Figura 71 se puede observar el aplicativo instalado y ejecutado desde un dispositivo móvil Android.

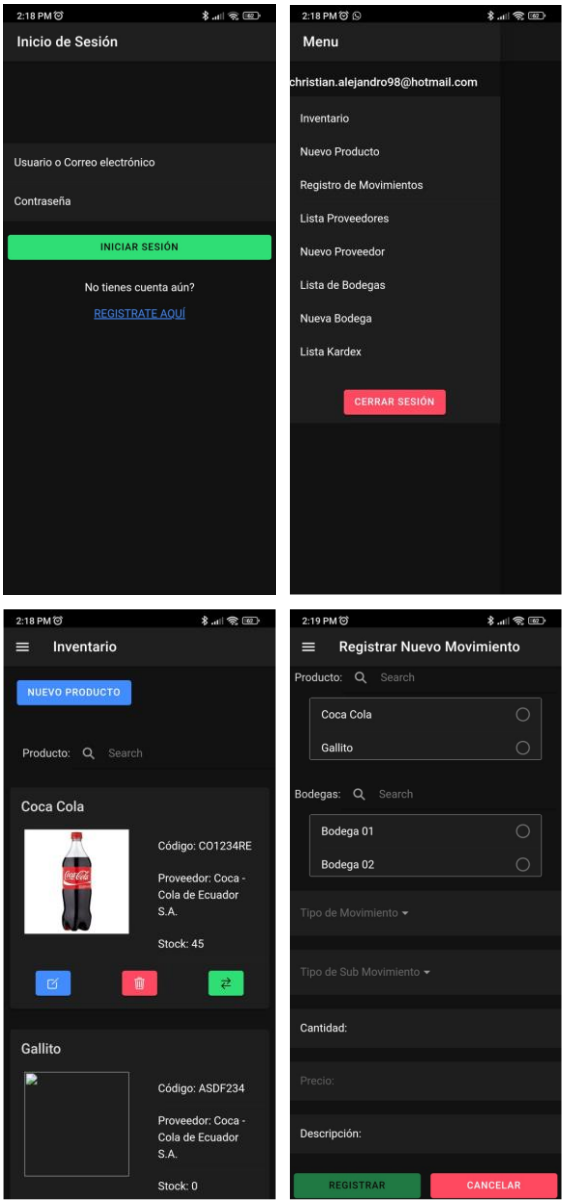

Figura 71. Aplicativo ejecutado desde un dispositivo real.

3.2 Pruebas de Usabilidad

Con el fin de evaluar la usabilidad del componente desarrollado en el presente trabajo, se decidió establecer una métrica de evaluación a través de una metodología de System Usabilility Scale (SUS), para lo cual se establecieron un total de 10 preguntas detalladas en la Tabla 16. Cada pregunta puede ser puntuada entre 1 o 5 puntos donde 1 es totalmente en desacuerdo y escala hasta el 5 que es totalmente acuerdo.

N°	Pregunta	Puntuación				
		1	2	3	4	5
1	Creo que me gustaría utilizar este sistema frecuentemente.					
2	El sistema me resultó innecesariamente complejo.					
3	Creo que el sistema es bastante fácil de utilizar.					
4	Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.					
5	Creo que las diferentes funciones del sistema se encuentran muy bien integradas.					
6	Opino que hubo demasiada inconsistencia en el sistema.					
7	Imagino que la mayoría de las personas aprendería a utilizar el sistema rápidamente.					
8	Me sentí algo incómodo al utilizar este sistema.					
9	Me sentí muy seguro al utilizar este sistema.					
10	Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.					

Tabla 16. Preguntas para SUS

La puntuación obtenible mediante esta métrica corresponde a un puntaje entre 0 y 100 puntos, donde un puntaje menor a 50 indica que el aplicativo es muy ineficiente si se habla de usabilidad. Su el puntaje está entre 50 y 68 puntos se considera marginal y aceptable si es mayor a 68 puntos [20]. Véase la Figura 72.

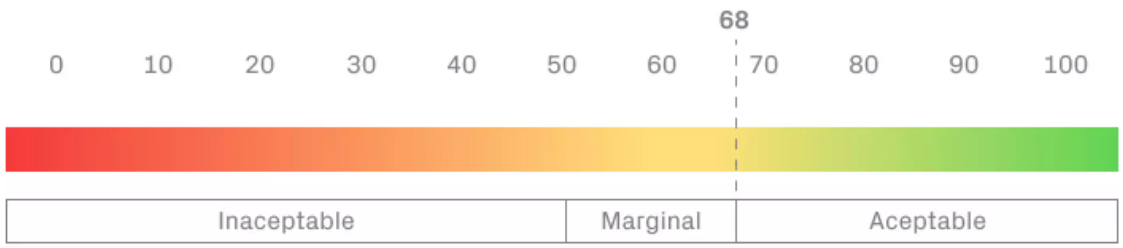

Figura 72. Referencias en puntajes SUS *[20]*

Para obtener el puntaje de la prueba al aplicativo se realizan los siguientes cálculos [20]:

- Calcular el promedio del puntaje de cada pregunta.
- Sumar el promedio de las preguntas impares (1,3,5,7,9) y al resultado restarle 5 puntos.

$$X = \left(\sum_{i=1}^{5} R_{(2i-1)}\right) - 5$$

Donde R es el conjunto de puntajes promediados.

- Sumar el promedio de las preguntas pares (2,4,6,8,10) y restar 25 puntos menos el resultado de la suma.

$$Y = 25 - \left(\sum_{i=1}^{5} R_{2i}\right)$$

Donde R es el conjunto de puntajes promediados.

- Sumar el resultado de los dos cálculos previos y lo multiplicamos por 2,5.

$$\text{Puntaje} = (X + Y) * 2{,}5$$

Esta métrica de evaluación fue llenada por 8 usuarios con promedio de edad de 24 años, mediante una encuesta creada en la herramienta Forms proporcionada por Google que puede ser vista en el Anexo VII. En la Figura 73 se muestran los resultados promediados de los puntajes referentes a las preguntas realizadas en las encuestas.

USUARIOS	PREGUNTAS									
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Usuario 1	4	1	5	4	4	2	4	1	3	2
Usuario 2	5	1	5	1	5	1	5	1	5	1
Usuario 3	5	1	5	1	5	1	5	1	5	1
Usuario 4	4	3	4	3	5	3	4	2	4	2
Usuario 5	5	2	5	1	5	1	5	1	5	1
Usuario 6	5	1	5	1	4	1	4	1	5	2
Usuario 7	4	2	4	1	4	1	3	1	5	2
Usuario 8	5	1	5	1	5	1	5	1	5	1
PROMEDIO	4,63	1,50	4,75	1,63	4,63	1,38	4,38	1,13	4,63	1,50

Tabla 17. Resultados Encuesta SUS

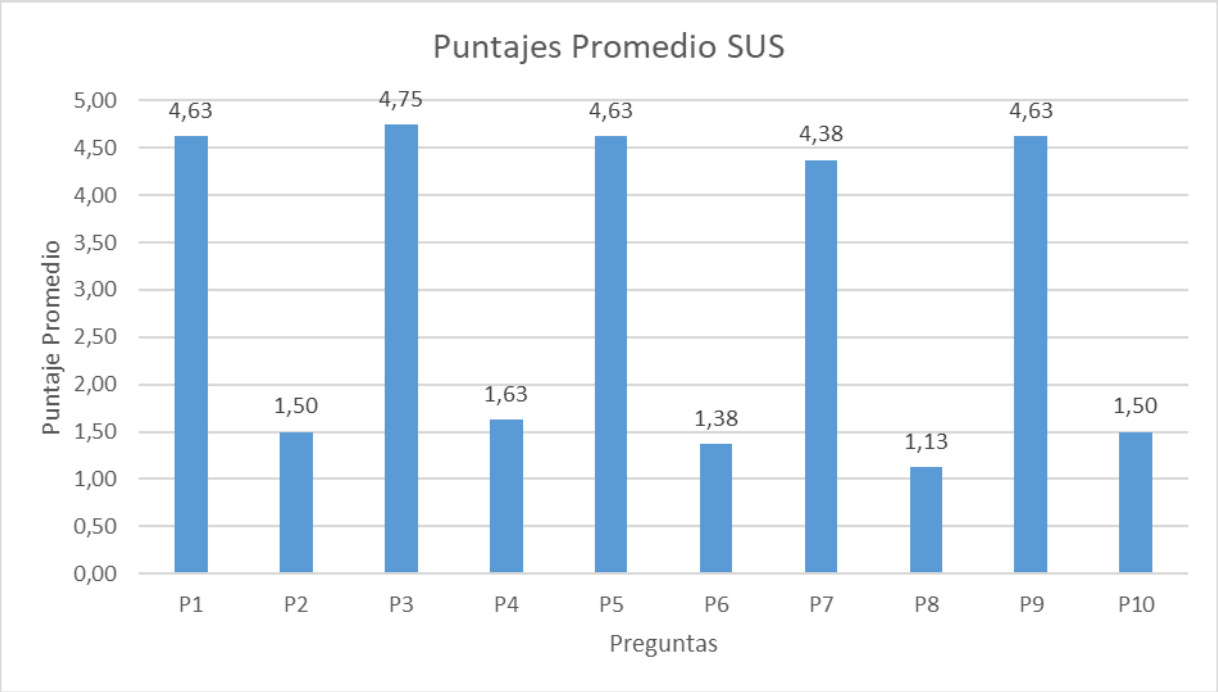

Figura 73. Resultado promedio de preguntas de SUS

Según los cálculos realizados mediante las preguntas, el valor del puntaje final siguiendo los cálculos es de 89,69 puntos, y al ser este puntaje mayor a 68, podemos decir que la usabilidad del aplicativo es aceptable.

Por otro lado, pudimos identificar que es necesaria una inducción al aplicativo ya que usuarios identificaron que puede llegar a ser necesaria la ayuda de un técnico que guíe el proceso de uso del aplicativo, además de necesitar una especificación de términos utilizados dentro del mismo, esto según los resultados obtenidos especialmente en las preguntas 4 y 10 del cuestionario.

3.3 Pruebas de Funcionalidad

Con el fin de medir la funcionalidad del aplicativo construido se realizaron pruebas específicas mediante el uso de 5 casos de prueba, los cuales fueron llevados a cabo por los mismos 8 usuarios que llenaron la encuesta SUS. A continuación, se muestran los casos de prueba creados con sus respectivas respuestas.

Número de Caso de Prueba: CP-001		Autor: Christian Naula	
Funcionalidad:	Ingresar al aplicativo móvil		
Entradas			
1. Credenciales de acceso al aplicativo correctas Usuario: usuario.prueba@gmail.com Contraseña: prueba123			
Descripción de acciones			
Nro.	**Acciones**		
1	Abrir el aplicativo desde el dispositivo móvil		
2	Ingresar las credenciales de acceso indicadas en la Entrada 1		
Resultados esperados		**Resultados Obtenidos**	
		Si	**No**	**Parcialmente**
Se accedió al sistema correctamente				
Observaciones			

Tabla 18. Caso de prueba CP-001

El caso de prueba 001 nos ayuda a validar el ingreso de la plataforma mediante autenticación utilizando un usuario previamente registrado. La Tabla 19 y la Figura 74 nos muestra los resultados obtenidos durante las pruebas realizadas.

Caso de prueba CP-001						
Resultados Esperados	**Resultados Obtenidos**			**Resultados Obtenidos (Porcentaje)**		
	Si	**No**	**Parcialmente**	**Si**	**No**	**Parcialmente**
Se accedió al sistema correctamente	8	0	0	100%	0%	0%

Tabla 19. Resultados del caso de prueba CP-001

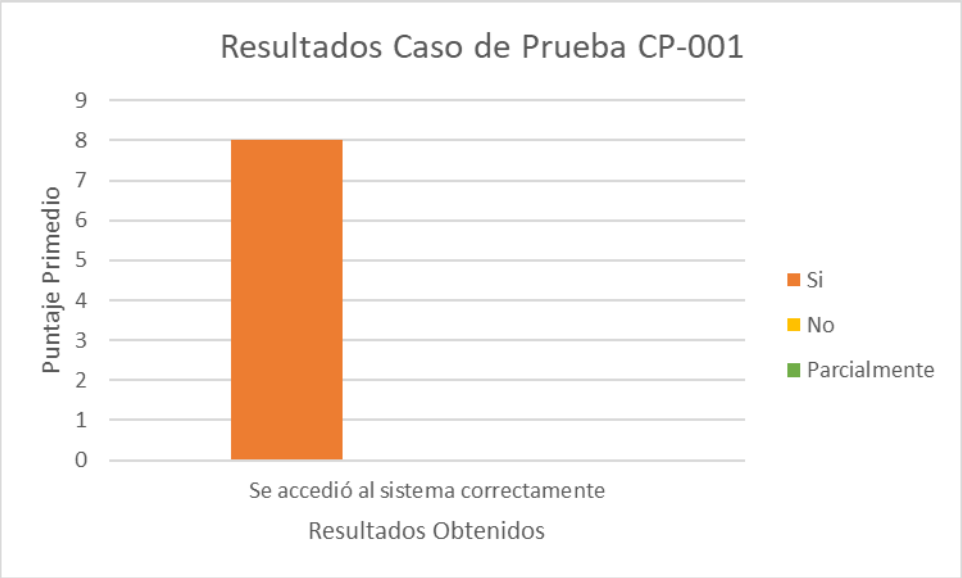

Figura 74. Resultados del caso de prueba CP-001

Número de Caso de Prueba: CP-002	**Autor: Christian Naula**
Funcionalidad:	Crear Nuevo Producto
Entradas	
1. Credenciales de acceso al aplicativo correctas Usuario: usuario.prueba@gmail.com Contraseña: prueba123 2. Formulario de nuevo producto • Proveedor: Nestle • Nombre: Natura Naranja • Código: NATN2384 • Categoría: Bebidas 3. Producto: Natura Naranja	
Descripción de acciones	
Nro.	**Acciones**
1	Abrir el aplicativo desde el dispositivo móvil
2	Abrir el menú lateral y seleccionar “Nuevo Producto”
3	Llenar el formulario de nuevo producto (Entrada 2)
4	Presionar el botón “CREAR”
5	En la pantalla de “Inventario” buscar el producto creado Entrada 3
6	Visualizar el producto creado
Resultados esperados	**Resultados Obtenidos**

	Si	No	Parcialmente
Se creó el producto satisfactoriamente			
Se Filtró el producto por su nombre desde el buscador			
Observaciones			

Tabla 20. Caso de prueba CP-002

El caso de prueba 002 nos ayuda a validar el proceso de creación de un nuevo producto y el filtrado del producto creado dentro del listado de todos los productos. La Tabla 21 y Figura 75. Resultados del caso de prueba CP-002 la nos muestra los resultados obtenidos durante las pruebas realizadas.

Caso de prueba CP-002						
Resultados Esperados	**Resultados Obtenidos**			**Resultados Obtenidos (Porcentaje)**		
	Si	**No**	**Parcialmente**	**Si**	**No**	**Parcialmente**
Se creó el producto satisfactoriamente	8	0	0	100%	0%	0%
Se filtró el producto por su nombre desde el buscador	7	0	1	88%	0%	13%

Tabla 21. Resultados del caso de prueba CP-002

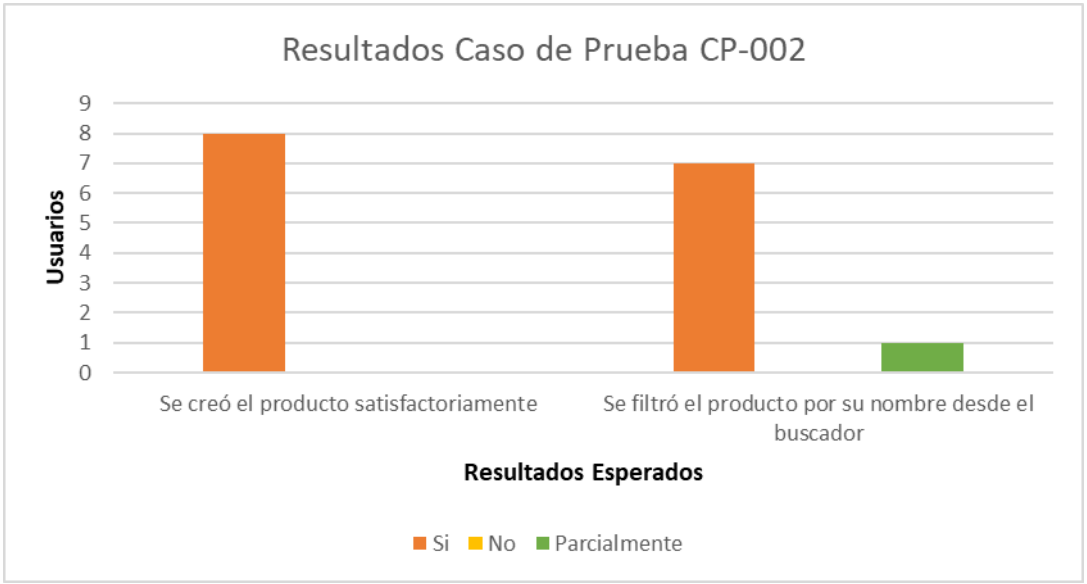

Figura 75. Resultados del caso de prueba CP-002

<table>
<tr><th colspan="3">Número de Caso de Prueba: CP-003</th><th colspan="3">Autor: Christian Naula</th></tr>
<tr><td>Funcionalidad:</td><td colspan="5">Crear Nueva Bodega</td></tr>
<tr><td colspan="6">Entradas</td></tr>
<tr><td colspan="6">1. Credenciales de acceso al aplicativo correctas
Usuario: usuario.prueba@gmail.com
Contraseña: prueba123

2. Formulario de nueva Bodega
• Nombre: Bodega 04
• Ubicación: Quito Centro
• Código: BOD004

3. Bodega: Bodega 04</td></tr>
<tr><td colspan="6">Descripción de acciones</td></tr>
<tr><td>Nro.</td><td colspan="5">Acciones</td></tr>
<tr><td>1</td><td colspan="5">Abrir el aplicativo desde el dispositivo móvil</td></tr>
<tr><td>2</td><td colspan="5">Abrir el menú lateral y seleccionar “Nueva Bodega”</td></tr>
<tr><td>3</td><td colspan="5">Llenar el formulario de nueva bodega (Entrada 2)</td></tr>
<tr><td>4</td><td colspan="5">Presionar el botón “CREAR”</td></tr>
<tr><td>5</td><td colspan="5">En la pantalla de “Lista Bodegas” buscar la bodega creada Entrada 3</td></tr>
<tr><td>6</td><td colspan="5">Visualizar el producto creado</td></tr>
<tr><td colspan="3" rowspan="2">Resultados esperados</td><td colspan="3">Resultados Obtenidos</td></tr>
<tr><td>Si</td><td>No</td><td>Parcialmente</td></tr>
<tr><td colspan="3">Se creó la bodega satisfactoriamente</td><td></td><td></td><td></td></tr>
<tr><td colspan="3">Se filtró la bodega por su nombre desde el buscador</td><td></td><td></td><td></td></tr>
<tr><td colspan="6">Observaciones</td></tr>
<tr><td colspan="6"></td></tr>
</table>

Tabla 22. Caso de prueba CP-003

El caso de prueba 003 nos ayuda a validar el proceso de creación de una nueva bodega y el filtrado de la bodega creada dentro del listado de todas las bodegas. La Tabla 23 y Figura 76. Resultados del caso de prueba CP-003 la nos muestra los resultados obtenidos durante las pruebas realizadas.

Caso de prueba CP-003						
Resultados Esperados	**Resultados Obtenidos**			**Resultados Obtenidos (Porcentaje)**		
	Si	**No**	**Parcialmente**	**Si**	**No**	**Parcialmente**
Se creó la bodega satisfactoriamente	8	0	0	100%	0%	0%
Se filtró la bodega por su nombre desde el buscador	8	0	0	100%	0%	0%

Tabla 23. Resultados del caso de prueba CP-003

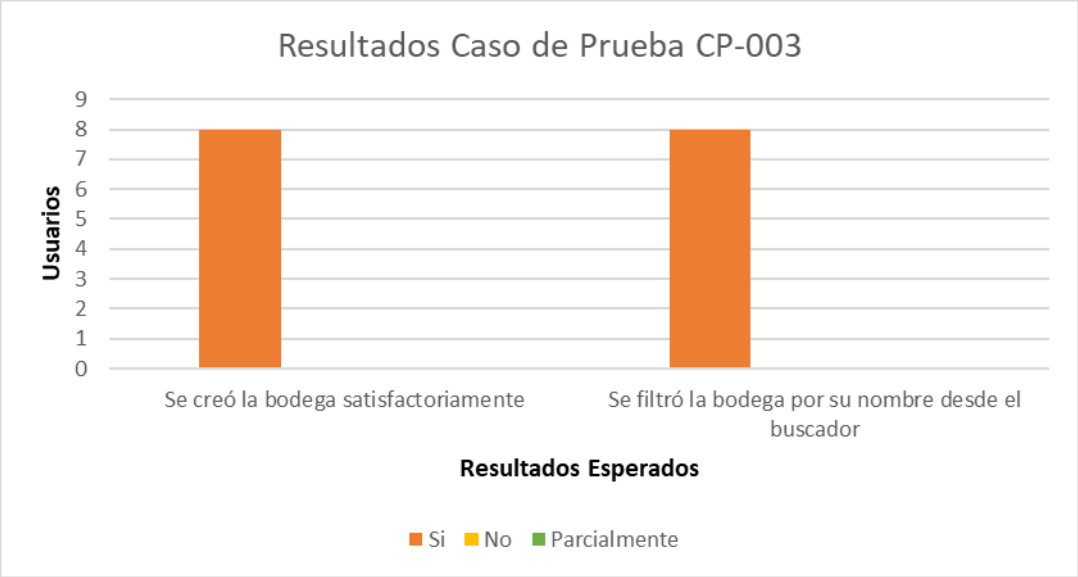

Figura 76. Resultados del caso de prueba CP-003

Número de Caso de Prueba: CP-004	**Autor: Christian Naula**
Funcionalidad:	Registrar Nuevo Movimiento
Entradas	
1. Credenciales de acceso al aplicativo correctas Usuario: usuario.prueba@gmail.com Contraseña: prueba123 2. Formulario de registro de nuevo movimiento • Producto: Natura Naranja • Bodega: Bodega 04 • Tipo Movimiento: Ingreso • Tipo de Sub Movimiento: Compra • Cantidad: 20 • Precio: 2.4 • Descripción: Prueba de movimiento 1 3. Bodega: Bodega 04	

4. Producto: Natura Naranja				
Descripción de acciones				
Nro.	**Acciones**			
1	Abrir el aplicativo desde el dispositivo móvil			
2	Abrir el menú lateral y seleccionar “Registro de Movimiento”			
3	Llenar el formulario de registro de movimiento (Entrada 2)			
4	Presionar el botón “REGISTRAR”			
5	Abrir el menú lateral y seleccionar “Lista de Bodegas”			
6	En la pantalla de “Lista Bodegas” buscar la bodega (Entrada 3)			
7	Presionar el botón de productos en la bodega (Botón verde en la esquina inferior derecha de la bodega)			
8	En la pantalla de la bodega buscar el producto (Entrada 4)			
9	Visualizar el producto ingresado en la bodega con la cantidad, el precio unitario y el precio total			
Resultados esperados		**Resultados Obtenidos**		
		Si	**No**	**Parcialmente**
Se registró el movimiento correctamente				
Se visualiza el producto ingresado en la bodega				
Observaciones				

Tabla 24. Caso de prueba CP-004

El caso de prueba 004 nos ayuda a validar el proceso de registro de un nuevo producto dentro de una bodega, y el filtrado de la del producto dentro del listado de todos los productos dentro de esa bodega. La Tabla 25 y Figura 77 la nos muestra los resultados obtenidos durante las pruebas realizadas.

Caso de prueba CP-004						
Resultados Esperados	**Resultados Obtenidos**			**Resultados Obtenidos (Porcentaje)**		
	Si	**No**	**Parcialmente**	**Si**	**No**	**Parcialmente**
Se registró el movimiento correctamente	8	0	0	100%	0%	0%

Se visualiza el producto ingresado en la bodega	8	0	0	100%	0%	0%

Tabla 25. Resultados del caso de prueba CP-004

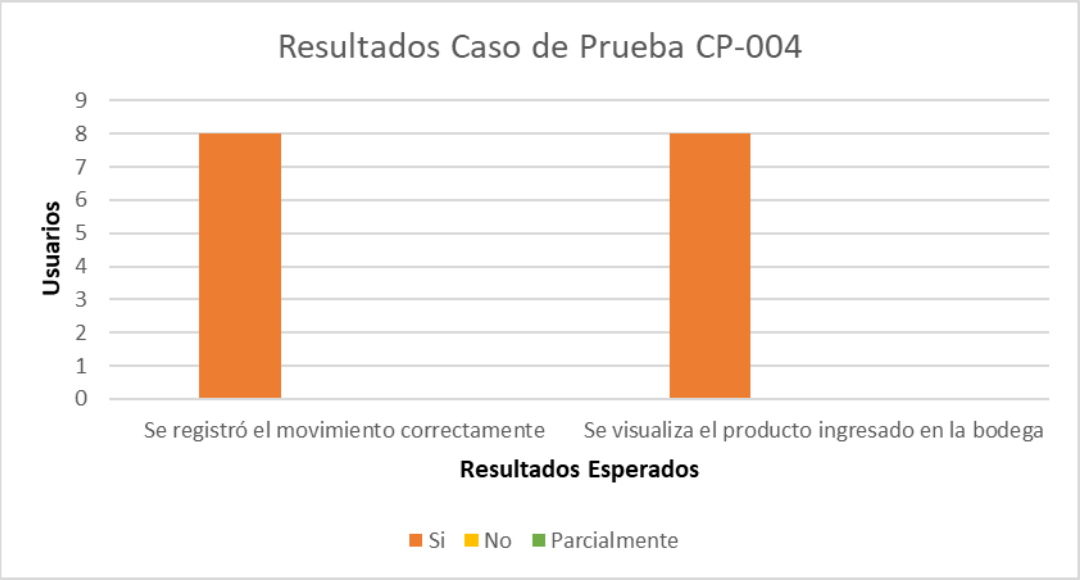

Figura 77. Resultados del caso de prueba CP-004

Número de Caso de Prueba: CP-005	**Autor: Christian Naula**
Funcionalidad:	Visualización de movimientos en Kardex
Entradas	
1. Credenciales de acceso al aplicativo correctas Usuario: usuario.prueba@gmail.com Contraseña: prueba123 2. Buscar por: Nombre del producto 4. Producto: Natura Naranja	
Descripción de acciones	
Nro.	**Acciones**
1	Abrir el aplicativo desde el dispositivo móvil
2	Abrir el menú lateral y seleccionar “Kardex”
3	Seleccionar el campo para buscar (Entrada 2)
4	Ingresar el nombre del producto (Entrada 3)
5	Visualizar el producto con su(s) respectivo(s) movimiento(s).
Resultados esperados	**Resultados Obtenidos**

	Si	No	Parcialmente
Se filtró por el nombre del producto			
Se visualiza el movimiento del producto solicitado			
Observaciones			

Tabla 26. Caso de prueba CP-005

El caso de prueba 005 nos ayuda a validar el proceso de filtrado de un movimiento específico realizado con un producto por su nombre. La Tabla 27 y Figura 78 la nos muestra los resultados obtenidos durante las pruebas realizadas.

Resultados Esperados	Resultados Obtenidos			Resultados Obtenidos (Porcentaje)		
	Si	No	Parcialmente	Si	No	Parcialmente
Se filtró por el nombre del producto	7	0	1	88%	0%	13%
Se visualiza el movimiento del producto solicitado	8	0	0	100%	0%	0%

Tabla 27. Resultados del caso de prueba CP-005

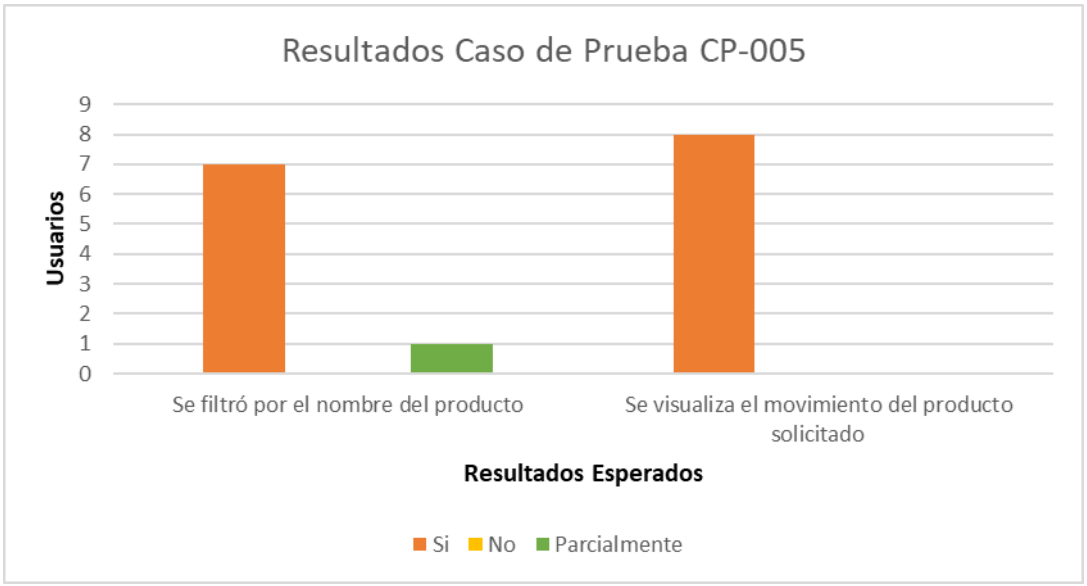

Figura 78. Resultados del caso de prueba CP-005

Mediante el uso de los casos de prueba se evidencia que el aplicativo funciona de manera satisfactoria, si bien hubo resultados con un 88% de completitud en la funcionalidad, en su mayoría se logró un 100%, lo que corrobora el funcionamiento esperado del aplicativo.

4 CONCLUSIONES

En el presente trabajo se completó el desarrollo e implementación de un módulo de manejo de Inventario de un sistema ERP, el cual está construido bajo una metodología de Scrum, haciendo uso de prácticas de DevOps y utilizando tecnologías de desarrollo web HTML, CSS y TypeScript bajo el Framework de desarrollo móvil Ionic.

Se completó la obtención de requerimientos de manera satisfactoria, con las que se construyeron historias de usuario que engloban las características deseadas para el aplicativo. Para completar este proceso se realizaron reuniones virtuales con el Product Owner donde se expusieron las solicitudes y como serían abordadas. A lo largo del proceso se presentaron inconvenientes referentes a la compresión de términos y funcionalidades específicas, como por ejemplo el objetivo del Kardex dentro del inventario y la funcionalidad que este tendría. Para solventar la presencia de estos particulares se mantuvo una comunicación efectiva y constante con el Product Owner, además se realizó una investigación sobre el manejo del Inventario y sus actividades.

La fase de diseño de las interfaces del aplicativo en la herramienta Figma se completó exitosamente junto con el modelo de base de datos creado, generando así un primer vistazo el aplicativo a construir y las funcionalidades a tener en cuenta. El proceso de diseño fue validado de manera íntegra y constante por el Product Owner. Durante esta etapa se presentaron conflictos respecto al despliegue de la información, puesto que como idea inicial se tuvo la construcción de tablas que muestren los datos, sin embargo, estas no pueden ser visualizadas de manera efectiva en dispositivos con pantallas pequeñas como es el caso de los dispositivos móviles. Como solución se optó por el despliegue de datos en forma de tarjetas para garantizar una correcta visualización.

Mediante el uso del Framework Ionic fue posible la creación de un aplicativo híbrido haciendo uso de tecnologías aplicadas al desarrollo web tales como HTML, CSS y TypeScript para la parte Frontend, y utilizar los servicios proporcionados por Firebase como Authentication y Cloud Firestore para la parte del Backend, sin necesidad de implementar un servidor. Durante el proceso de construcción se presentaron problemas referentes a la falta de conocimiento de la tecnología, como sus componentes y su alcance, el manejo del ciclo de vida de sus interfaces, métodos propios del Framework, etc. Para solventar este particular se recurrió a la investigación y aprendizaje de la herramienta mediante su documentación oficial y terceros.

El uso del marco de trabajo Scrum facilitó en gran medida el ciclo de desarrollo del aplicativo, esto debido a las fases ya establecidas que nos proporciona este marco con el fin de garantizar una organización efectiva, como es el caso del uso de sprints. Adicionalmente, mediante este marco es posible la fácil adaptación frente a los cambios que pueden presentarse en cada una de las etapas, además de la facilidad que presenta el uso de historias de usuario previamente definidas para garantizar así el funcionamiento esperado, basándonos en los criterios de aceptación de cada funcionalidad previamente acordada.

El uso de un conjunto de prácticas de DevOps permitió optimizar en tiempo y recursos distintas etapas primordiales en el desarrollo del software, ya que de esta manera es posible configurar tareas específicas a realizar al momento de cumplir o finalizar partes concretas del trabajo, como fue el caso de la Integración Continua, añadiendo el hecho de la facilidad de implementar estas prácticas, tal y como es el presente caso, donde se hizo uso de una herramienta gratuita como GitLab y obteniendo los resultados esperados sin mayor complicaciones en el ámbito del DevOps.

5 RECOMENDACIONES

Se recomienda ampliar el alcance del aplicativo incluyendo un módulo de manejo de órdenes de compra de productos con destino a los distintos proveedores, ya que se ha visualizado que es una opción muy demandada en el manejo del inventario.

De igual manera, se recomienda implementar políticas de respaldo de información de la base de datos creada en Cloud Firestore, además de añadir un sistema de reportes de productos directamente desde el aplicativo.

Tal y como se menciona a lo largo del presente documento, el aplicativo fue desarrollado en un entorno web, con destino de aplicativos móviles, por lo que corresponde a un aplicativo híbrido. Sin embargo, es prudente evaluar la implementación en un entorno Android nativo con el fin de evaluar su eficacia y rendimiento en comparación a la aplicación creada.

Se recomienda evaluar la implementación de las prácticas de DevOps en otras plataformas con el fin de compararla con la implementada en GitLab y definir mejores o peores opciones en base a su eficiencia.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] «Oracle,» [En línea]. Available: https://www.oracle.com/erp/what-is-erp/. [Último acceso: 16 agosto 2022].

[2] J. Cuello, Diseñando apps para móviles, Catalina Duque Giraldo, 2014.

[3] S. Serna, Diseño de interfaces en aplicaciones móviles, Madrid: RA-MA, 2016.

[4] A. Sarasa, Introducción a las bases de datos NoSQL usando MongoDB, Barcelona: UOC (Oberta UOC Publishing, SL), 2016.

[5] «TypeScript Comunity, "TypeScript: Documentation",» [En línea]. Available: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html. [Último acceso: 19 mayo 2022].

[6] D. Gayo, Diseño gráfico de páginas web, Oviedo, 2000.

[7] G. B, «¿Qué es CSS?,» Hostinger, 12 julio 2022. [En línea]. Available: https://www.hostinger.es/tutoriales. [Último acceso: 2022 agosto 16].

[8] J. M. A. Atmitim, «Qué es Ionic: ventajas y desventajas de usarlo para desarrollar apps móviles híbridas,» profile, 22 febrero 2021. [En línea]. Available: https://profile.es/blog/que-es-ionic/#%C2%BFQue_es_Ionic_y_para_que_sirve. [Último acceso: 2022 agosto 16].

[9] Y. Muradas, «Qué es Firebase: Conoce la plataforma de Google,» OpenWebinars, 22 junio 2021. [En línea]. Available: https://openwebinars.net/blog/que-es-firebase-de-google/. [Último acceso: 16 agosto 2022].

[10] «Cloud Firestore,» Firebase, [En línea]. Available: https://firebase.google.com/docs/firestore. [Último acceso: 16 agosto 2022].

[11] «El concepto de IDE,» Red Hat, 08 enero 2019. [En línea]. Available: https://www.redhat.com/es/topics/middleware/what-is-ide. [Último acceso: 17 agosto 2022].

[12] «El IDE más inteligente para JavaScript,» WebStorm, [En línea]. Available: https://www.jetbrains.com/es-es/webstorm/#:~:text=WebStorm%20es%20un%20entorno%20de,las%20tareas%20complejas%20con%20facilidad.. [Último acceso: 17 agosto 2022].

[13] «¿Qué es DevOps?,» Microsoft Azure, [En línea]. Available: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops/. [Último acceso: 17 agosto 2022].

[14] «¿Qué es DevOps?,» Amazon AWS, [En línea]. Available: https://aws.amazon.com/es/devops/what-is-devops/. [Último acceso: 17 agosto 2022].

[15] «¿Qué es Git?,» Microsoft Azure, 22 julio 2011. [En línea]. Available: https://docs.microsoft.com/es-es/devops/develop/git/what-is-git. [Último acceso: 17 agosto 2022].

[16] «1.3 Inicio - Sobre el Control de Versiones - Fundamentos de Git,» git, [En línea]. Available: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git. [Último acceso: 17 agosto 2022].

[17] «Conoce qué es GitLab y gestiona efectivamente los cambios que realices en tus proyectos,» Crehana, [En línea]. Available: https://www.crehana.com/blog/desarrollo-web/que-es-gitlab/. [Último acceso: 17 agosto 2022].

[18] «Introducción a GitLab,» 12 octubre 2017. [En línea]. Available: https://desarrolloweb.com/articulos/introduccion-gitlab.html. [Último acceso: 17 agosto 2022].

[19] Ken Schwaber y Jeff Sutherland, La Guía de Scrum, 2013.

[20] C. Busquets, «Medir la usabilidad con el Sistema de Escalas de Usabilidad (SUS),» ui-frommars-, [En línea]. Available: https://www.uifrommars.com/como-medir-usabilidad-que-es-sus/. [Último acceso: 03 septiembre 2022].

7 ANEXOS

ANEXO I. Enlace al repositorio que contiene el proyecto de Ionic de la aplicación construida

https://gitlab.com/manticore-labs/trabajo/proyectos/trabajo-de-titulacion/cp-naula

ANEXO II. Enlace al prototipo de interfaces del aplicativo

https://www.figma.com/file/Htl8hEsgwjIcHnc2lTfgM6/Mockups-Invetario?node-id=0%3A1

ANEXO III. Enlace a la carpeta compartida con las historias de usuario

https://epnecuador-my.sharepoint.com/:f:/g/personal/christian_naula_epn_edu_ec/Eggo0pyAZklDuSYiWpLrgq0BM5QgVEXDHhxEXjoIPmFtPQ?e=wDkClj

ANEXO IV. Enlace a la carpeta compartida con la arquitectura, diagrama de navegación y modelo de base de datos del aplicativo

https://epnecuador-my.sharepoint.com/:f:/g/personal/christian_naula_epn_edu_ec/EqNbgSElbXZPifhMiQeqo9AB7TTYiKOO1n2Nw_BSBkZ7Ug?e=iyJHoX

ANEXO V. Enlace a la carpeta compartida con el documento online del presente trabajo

https://epnecuador-my.sharepoint.com/:f:/g/personal/christian_naula_epn_edu_ec/EoM459Zd90hOsEFZ75qvXa0BiMpxrizVGS7tjcbED0AFYQ?e=zL9SPz

ANEXO VI. Enlace a la imagen circleci/android

https://hub.docker.com/r/circleci/android/tags

ANEXO VII. Enlace a la encuesta Escala de Usabilidad del Sistema (SUS)

https://docs.google.com/forms/d/e/1FAIpQLScm4IlAWRVsHd6sTcjHJvJ949EcyduZGt3X-a3xMAJF8zEinA/viewform?usp=sf_link

ANEXO VIII. Enlace a las respuestas de la encuesta Escala de Usabilidad del Sistema (SUS)

https://docs.google.com/spreadsheets/d/1YYP95u7mpx9naVRle4HCS9mqtzJEXTetRHjhxQCnia8/edit?usp=sharing