Título: Advanced Mobile Onsite Documentation System (AMODS)

Jonathan Torres Vélez

Ingeniero de Sistemas

Asesor

Javier Fernando Botía Valderrama, Doctor en ingeniería Electrónica

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de sistemas

Medellín, Antioquia

2024

Cita	Torres Vélez Jonathan, 2024
Referencia Estilo IEEE (2020)	Torres Vélez, Jonathan, "Advanced Mobile Onsite Documentation System (AMODS)", Trabajo de grado profesional, Departamento de ingeniería de Sistemas, Universidad de Antioquia, Medellín, 2024.

Créditos a Perficient Latinoamérica por permitir el desarrollo de la práctica en la empresa.

Centro de documentación de ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda.

Decano/Director: Julio César Saldarriaga.

Jefe departamento: Danny Alexandro Múnera Ramírez.

Dedicatoria

A mis hermanos, gracias por siempre impulsarme a dar lo mejor de mi y perseverar para cumplir mis objetivos.

Agradecimientos

Agradezco a mis hermanos que me han acompañado durante todo mi proceso educativo impulsándome a siempre ser mejor y motivándome en momentos en los que el proceso se tornó difícil; y a mis compañeros de trabajo que apoyaron todo mi proceso laboral para lograr desarrollar este trabajo.

Tabla de contenido

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

SCRUM	Sprint, Certified, Releases Unanimity and Meetings
BDD.	Behavior Driven Development
UdeA	Universidad de Antioquia
AMODS	Advanced Mobile Onsite Documentation System
OEMs	Original Equipment Manufacturers of Vehicles

RESUMEN

Proyecto creado para desarrollar una aplicación AMODS con el objetivo de reemplazar la aplicación de un tercero usada por el cliente. Los usuarios de esta aplicación se dedican al upfitting vehicles y a instalar accesorios en flotas de fuerza policial, aseo y transporte de pasajero. Con la realización del proyecto se apunta a la creación de una aplicación web y móvil que permita manejar inspecciones de vehículos o flotas vehiculares a través de varios módulos importantes (inspecciones, reportería, gestión de usuarios y roles, configuración) desarrolladas bajo el marco de trabajo SCRUM. Entre las principales razones por las que se realiza el desarrollo y el cambio de la aplicación de terceros hacia un producto propio, se encuentran la administración del código fuente de la aplicación, realizar mejoras a la interfaz de usuario, mejorar la experiencia del usuario, tener compatibilidad con el sistema operativo IOS (aplicación móvil a reemplazar solo cuenta con versión para Android), mejorar el rendimiento, funcionalidad y mantenibilidad.

Palabras clave **— Ingeniería de sistemas, AMODS, arquitectura limpia, arquitectura hexagonal, SCRUM, web, móvil, inspección vehicular.**

ABSTRACT

Project created in order to develop an AMODS application with the purpose of replacing an old third party application used by the client. Users of this application are dedicated to upfitting vehicles and install accessories in police force, cleaning and passenger transport fleets.

With the development of this project the aim is to create a web and mobile application that allows the user to handle vehicle or fleet inspections through several important modules (inspections, reporting, user and role management, settings) developed under SCRUM framework. Among the main reasons this development and application changed from third party to own is carried out is the source code management, improve user interface, improve user experience, get IOS compatibility for the mobile application (third party app is only compatible with Android), improve performance, functionality and maintainability.

***Keywords* — (Ejemplo) Systems engineering, AMODS, clean architecture, hexagonal architecture, SCRUM, web, mobile, vehicle inspections.**

I. INTRODUCCIÓN

Una empresa cuya función es dedicarse a la venta, diseño e instalación de soluciones inteligentes de seguridad para vehículos y flotas vehiculares, utilizando tecnología como video o gps con el fin de brindar una perspectiva de un ambiente más seguro para conductores, pasajeros y peatones principalmente en América del norte, realiza múltiples inspecciones de vehículos y flotas para diferentes clientes, este proceso de inspección lo realizaban a través de una aplicación de terceros sobre la cual se ha manifestado una inconformidad con su rendimiento y usabilidad, especialmente por el personal encargado de realizar dichas inspecciones a través de una aplicación móvil, adicionalmente se hace énfasis en la necesidad de poder trabajar sin conexión a internet, ya que en ocasiones estos procesos se realizan en lugares donde no se cuenta con buena conexión a internet como sótanos y parqueaderos multinivel. Por lo anterior, se toma la decisión de migrar a una aplicación propia, sobre la cual se pueda tener control del código, ser mantenida, escalada, mejorada y adaptada a las necesidades del proceso, que cuente con las funcionalidades que actualmente facilitan inspeccionar vehículos más algunas adicionales propias del cliente, para de esta manera cubrir las falencias de la solución actual.

La solución se piensa desde dos frentes, uno web y otro móvil, por lo que se desarrollan dos aplicaciones, la primera para configuración y reportería y la segunda para realizar inspecciones a vehículos automotores. Se trabaja bajo la metodología ágil SCRUM (Sprint, Certified, Releases Unanimity and Meetings) y Behavior Driven Development (BDD) facilitando entregas de valor parciales, siguiendo el concepto de arquitectura limpia y arquitectura hexagonal para el desarrollo de los productos a ser entregados, los cuales serán revisados por bajo el ojo crítico del frente de desarrollo y aseguramiento de la calidad para garantizar que el producto cumpla con los estándares acordados; análisis de código estático con *SonarQube*, pruebas de software y pruebas finales con los usuarios de la aplicación confirmando que el producto satisface las necesidades del cliente.

Se logra realizar las primeras versiones de la solución con el mínimo producto viable, al cual se continúa haciendo mejoras y sumando funcionalidades pactadas con los usuarios finales; este producto es desplegado en la infraestructura del cliente y las diferentes tiendas para dispositivos móviles luego de cumplir con el ciclo de vida de desarrollo, permitiendo que los

usuarios actualmente puedan darle uso y continuar con la operación sin el uso de la herramienta anterior.

II. OBJETIVOS

A. Objetivo general

Desarrollar dos aplicaciones como solución a los requerimientos del cliente, una web y otra móvil, que permita al usuario realizar sus funciones de programación y realización de inspecciones de las flotas vehiculares propias y de sus clientes, además de acceder de manera fácil y rápida a la información obtenida de las mismas con tiempos de respuesta cortos y una interfaz gráfica pensada en mejorar la experiencia del usuario, permitiendo tener control sobre el código fuente, y reducir gastos en membresías de la aplicación que se usaba anteriormente como herramienta para realizar estos procesos sobre la cual se manifiesta inconformidad en su funcionamiento y usabilidad.

B. Objetivos específicos

- Analizar las necesidades y problemas del cliente, como del software que se planea cambiar, además de los lenguajes, metodologías y frameworks de desarrollo que más se adapten a estas necesidades para garantizar que el producto final cumpla con todas las expectativas del cliente y en realidad si cumplan con dichas necesidades, permitiendo que el software se convierta en una herramienta clave en los procesos de los usuarios.
- Desarrollar una aplicación móvil que facilite al usuario la realización de procesos de inspección de flotas vehiculares basadas en plantillas de inspección, , que le permita almacenar la información obtenida durante dicho proceso así como dejar comentarios y guardar contenido multimedia como evidencia adicional de la información ingresada por el usuario; con tiempos de respuesta cortos y una interfaz de usuario que sea intuitiva y pueda usarse sin conexión a internet (offline).
- Generar una aplicación web que brinde al usuario la posibilidad de configurar las inspecciones que se realizarán para cada vehículo o flota vehicular, visualización de

reportes sobre los procesos que se han llevado a cabo y gestión de usuarios de una manera intuitiva con una interfaz de usuario amigable.

- Validar que el entregable del proyecto (ambas aplicaciones desarrolladas) cumplan con los estándares de calidad pactados y establecidos así como con los requerimientos levantados inicialmente, garantizando que el producto entregado cumpla con las necesidades y exigencias del cliente para pueda ser utilizado sin problemas en el desarrollo de las funciones del cliente.

III. MARCO TEÓRICO

Se busca desarrollar un software que será implementado como herramienta para realizar inspecciones vehiculares para el cliente, gestionar y configurar plantillas de inspección y usuarios, visualizar reportes de los datos ingresados mediante este proceso; para el desarrollo de este proyecto se han implementado metodologías ágiles, en este caso SCRUM, como herramienta de gestión de proyectos, permitiendo que el trabajo en equipo sea un poco más eficiente y ordenada, dividiendo el desarrollo del proyecto en pequeños hitos entregables que serán abordados en ciclos de trabajo cortos (2 semanas por Sprint) que agregan valor y funcionalidad al usuario paulatinamente.

SCRUM [1] es un marco de gestión de proyectos de metodología ágil, en el que se aplican un conjunto de buenas prácticas para trabajar colaborativamente. Con esta metodología, se realizan entregas parciales del producto esperado regularmente, priorizadas por el beneficio que aportan a la solución final, razón por la cual este marco de trabajo es muy recomendable para proyectos muy complejos en los que se necesita entregar resultados pronto. En SCRUM se ejecutan ciclos de trabajo cortos u Sprints (2 semanas para este proyecto) en los que sea posible para el equipo de trabajo realizar características que vayan agregando funcionalidad al producto final y se definen ciertas reuniones o ceremonias para cada ciclo que permitan abordar la manera en la que se trabajará cada iteración o ciclo de trabajo.

Fig 1. Proceso de SCRUM [2]

BDD es una estrategia de desarrollo de software en la que se plante un lenguaje común llamado *Gherkin* para definir casos de prueba relacionados a historias de usuario [3], donde el hecho de tener un lenguajes que todo el equipo de trabajo pueda entender claramente facilita el entendimiento de las historias de usuario por parte de todo el equipo ayudando significativamente también el proceso de testing, pues los escenarios a probar son claros.

Además, al incluir un lenguaje común poco técnico, se facilita que incluso los miembros del equipo con poco conocimiento técnico puedan entender y aportar su punto de vista al desarrollo de una historia de usuario, pues este hecho facilita el debate para poder aterrizar el comportamiento de la funcionalidad que se va a abordar antes de que los desarrolladores involucrados inicien con la escritura del código, con esta metodología se realizan una serie de pasos listados a continuación:

- Se analizan los requisitos de software y objetivos de la funcionalidad.
- Se describen detalladamente las funciones del software en escenarios que ilustran el comportamiento del software
- Se valida la respuesta del software en cada escenario.

Implementar BDD en un proyecto de software, por lo general, genera ciertas ventajas como la mejora de la comunicación interna del equipo, generación de acuerdos previos al desarrollo, posee una curva de aprendizaje corto por lo que puede ser implementada fácilmente y tiene buena compatibilidad con metodologías ágiles como la que se implementa en este proyecto.

Arquitectura Hexagonal y *Vertical Slicing*, como parte de las decisiones de los arquitectos de software del proyecto se ha optado por implementar una arquitectura hexagonal, la cual sugiere que la aplicación sea dividida en diferentes capas con responsabilidades muy propias. De esta

manera, se busca que todas las capas sean desacopladas, facilitando el entendimiento de la aplicación y facilitando cambios en la misma sin que afecten funcionalidades de las demás capas haciendo el software más escalable, facilitando la reutilización de componentes de software y permitiendo que cada capa pueda ser probada de manera individual y separada de las demás. Esta arquitectura suele representarse en forma de hexágonos anidados representando cada capa, cada una de ellas expone diferentes puestos o interfaces que posibilitan la interacción entre ellas donde las capas externas dependen de las capas internas y no viceversa [4].

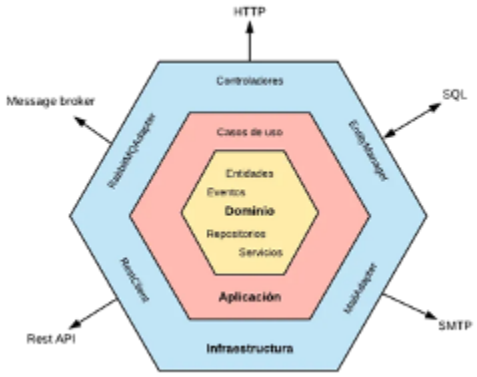

Fig 2. Arquitectura Hexagonal [5]

Adicionalmente, con la implementación del concepto de *Vertical Slicing*, el cual busca dividir el proyecto también por módulos clasificados debido a su funcionalidad en los que cada módulo implementado posee la estructura de la arquitectura hexagonal, permitiendo que el proyecto no solo se encuentra desacoplado a nivel de capas sino también de funcionalidades dando como resultado una gran mantenibilidad, escalabilidad y reducción de acoplamiento entre componentes de software.

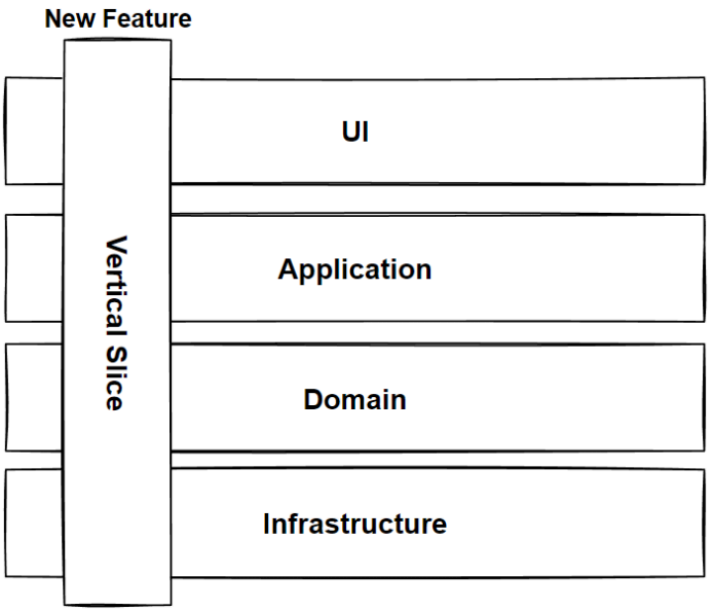

Fig 3. Vertical Slicing [6]

Cómo medida de aseguramiento de la calidad se han implementado varias estrategias, con el fin de mantener la mejor calidad en todos los componentes desarrollados, entre ellas se encuentra la implementación de pruebas de software **(En el proyecto se realizan pruebas unitarias, pruebas de contrato, y pruebas de componentes)** permitiendo que gran parte de las pruebas puedan ser automatizadas y de esta manera, reducir la cantidad de pruebas manuales que se deben realizar. Se ha configurado *SonarQube* como apoyo para el análisis de código estático siguiendo los parámetros listados a continuación:

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

Fig 4. Métricas SonarQube [7]

Además, antes de aprobar cada uno de los *pull requests* de código realizados hacia alguna de las ramas principales, se realiza un proceso de revisión por pares como condición para hacer mezclas de código, en donde se verifica por parte del equipo de desarrollo que el código cumpla con los

estándares pactados permitiendo un seguimiento de las buenas prácticas de desarrollo en el equipo de trabajo. Adicionalmente, se han configurado firmas para cada uno de los *commits* subidos a los repositorios pertenecientes a la organización, asegurando así que todo el código que se sube a los repositorios provenga de personas autorizadas para esto.

Cómo estrategia de *branching* se usa *GitFlow* estandarizando la creación de ramas dentro del repositorio de código; esta estrategia propone la creación de unas ramas bases y diferentes ramas para el trabajo de funcionalidades que serán mezcladas cuando termina el desarrollo y son aprobadas por el equipo de trabajo hacia las ramas principales. En el proyecto se han definido las siguientes ramas dentro del repositorio utilizado:

- *Feature branch*: Ramas donde se trabajan funcionalidades nuevas que serán agregadas al producto
- *Develop branch*: Rama como todos los desarrollos aprobados
- *Release branch*: Rama que contiene los cambios específicos que serán liberados para uso de los usuarios finales
- *Hotfix branch*: En la cual se trabajan soluciones a errores.

El trabajo con estas ramas sigue el siguiente flujo basado en la metodología de *GitFlow*

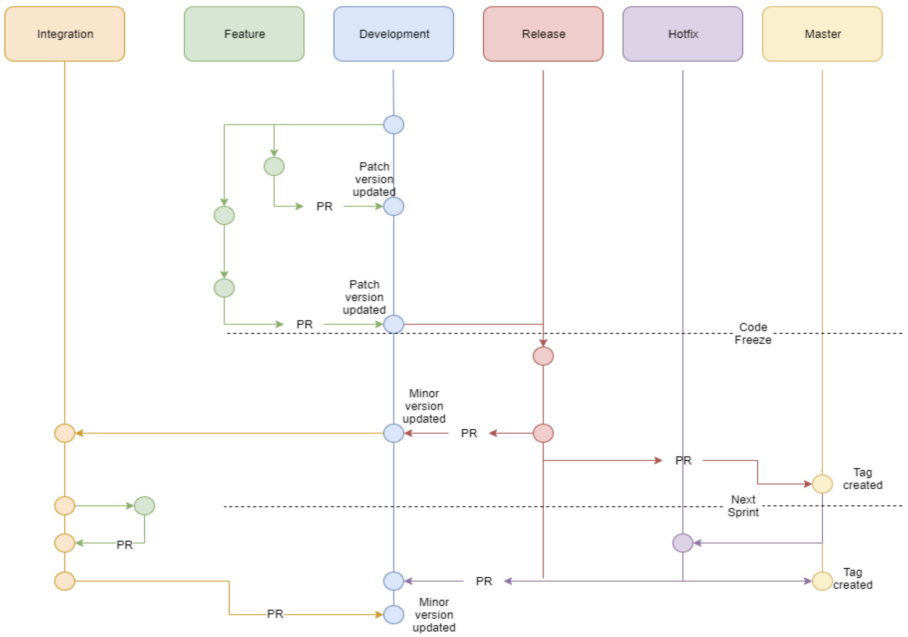

Fig 5. Estrategia de branching [8]

IV. METODOLOGÍA

Inicialmente, a través de recolección de información, focus groups, reuniones, talleres, requerimientos escritos (insumo compartido por los *stakeholders*), análisis de la situación del usuario con la aplicación que se encontraba utilizando y de los problemas e inconformidades manifestadas sobre la misma, se logra tener la información necesaria para un levantamiento de requisitos enfocado en las necesidades del cliente y la solución a los problemas que tiene al momento de realizar su proceso de inspecciones, así como también tener la información inicial que apoyara las decisiones tomadas por los arquitectos de software y desarrolladores basándose en esta información.

El desarrollo de las aplicaciones es apoyado por diferentes frentes de trabajo entre los cuales se distribuye el trabajo basado en sus funciones y se dividen las responsabilidades, entre estos frentes de trabajo se pueden identificar:

- *Product Managers*: Encargados de liderar los pasos del ciclo de vida del producto, enfocados en el producto final y el cliente, garantizando que el producto satisfaga las necesidades del cliente.
- *Arquitectura*: Apoyan al equipo de trabajo con la toma de decisiones técnicas muy significativas como el diseño de software, arquitectura, selección de tecnologías a utilizar o patrones de diseño a implementar en el desarrollo, asegurando estándares de calidad.
- *Diseño*: Brindan apoyo con el diseño visual y apariencia en general de las aplicaciones desarrolladas.
- *Desarrollo*: Encargados de desarrollar las funcionalidades de la aplicación basados en los requisitos levantados con el cliente, diseño, arquitectura y patrones de diseño a implementar..
- *Analistas de calidad*: Encargados de garantizar que la solución cumpla con todos los estándares de calidad pactados.

Con estos frentes, se logra formar un equipo de trabajo capaz de llevar a cabo el proyecto bajo los mejores estándares, el cual se reunirá periódicamente cumpliendo con las ceremonias propuestas por la metodología SCRUM.

La implementación del proyecto se lleva a cabo basado en el concepto de arquitectura limpia procurando que la solución sea modular y que haya poca cohesión entre estos módulos,

garantizando que sea fácil de entender, mantener o implementar mejoras al producto con el paso del tiempo, además implementando el patrón de diseño de arquitectura hexagonal separando la aplicación en capas que se comunican a través de interfaces, disminuyendo el acoplamiento entre componentes de software y facilitando cambios y aumentando la escalabilidad.

Al desarrollar una funcionalidad, los entregables de código son mezclados al código principal siguiendo la estrategia de *GitFlow* en la cual se definen ramas principales a las que serán mezclados paulatinamente el código de funcionalidades para ser desplegado en diferentes ambientes, y se define que para cada funcionalidad nueva que se va a agregar debe ser mezclada desde una rama de funcionalidad hasta las ramas principales.

Como medidas de aseguramiento de la calidad, se llevan a cabo análisis estáticos de código con *SonarQube*, análisis de código *pre-commit*, revisión por pares de todas las solicitudes de mezcla a ramas principales, pruebas de software a cargo del frente de desarrollo, pruebas manuales y automatizadas a cargo del frente de calidad del equipo, quienes se encargan de validar el cumplimiento de todos los requisitos funcionales y no funcionales y las historias de usuario desarrolladas durante cada *Sprint* antes de ser enviados al último filtro de calidad que consta de pruebas realizadas por los usuarios finales en ambientes pre-productivos.

V. RESULTADOS

En esta sección se mencionará brevemente lo resultados obtenidos con la aplicación ya que debido a restricciones de privacidad firmadas con el cliente no es posible dar mucho detalle sobre su funcionamiento, sin embargo, se hará claridad que los resultados obtenidos con la aplicación por parte del usuario y el equipo han sido exitosos, y que ambas partes manifiestan satisfacción con el producto.

Se logra cumplir con el desarrollo del mínimo producto viable para el usuario final en los tiempos establecidos en el *timeline* con el cliente, alcanzando un estado del producto con el cual el usuario puede trabajar día a día, tanto los usuarios *backoffice* (usuarios de la aplicación web) como los inspectores (usuarios de la aplicación móvil) logran apoyar sus procesos laborales con las aplicaciones desarrolladas, manifestando conformidad con el producto, la aplicación continua en desarrollo añadiendo cada vez más funcionalidades que faciliten y aceleren los procesos del cliente.

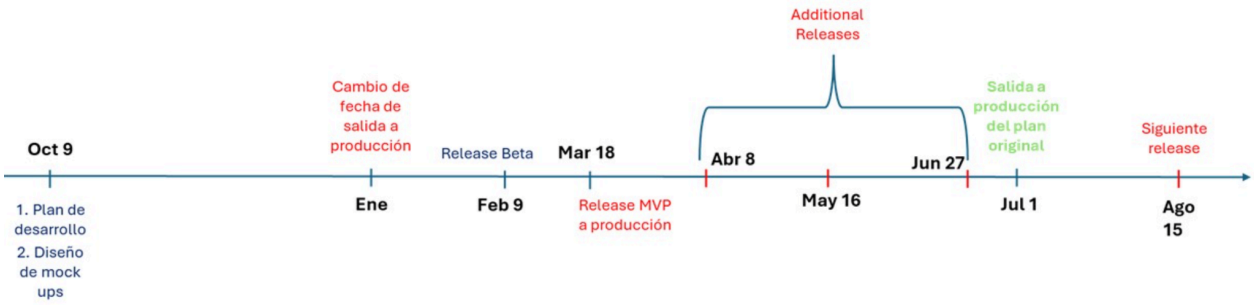

Fig 6. Timeline [9]

VI. ANÁLISIS

Inicialmente con el correcto levantamiento de requisitos del sistema, levantamiento que se lleva a cabo satisfactoriamente gracias a la investigación realizada por el equipo de producto con el cliente, se logra definir un *timeline* con actividades y tiempo alcanzables que satisfagan tanto al cliente como al equipo de trabajo, dando tranquilidad al equipo de que hay tiempo suficiente para desarrollar las actividades organizadas y priorizadas de manera que al entregar parcialmente funcionalidades el usuario también sienta tranquilidad. El producto se logra realizar gracias a la correcta organización del equipo, levantamiento de requerimientos, decisiones de diseño y arquitectura acertadas, formando un conjunto de parámetros que permitieron que el desarrollo de la aplicación se llevará a cabo de manera satisfactoria. Adicionalmente, parte de la tranquilidad manifestada por el cliente con el desarrollo realizado se debe también a los procedimientos de aseguramiento de calidad y realización constante de pruebas tanto por parte del equipo cómo del usuario final, que estuvo dispuesto a realizar pruebas constantemente y retroalimentar permitiendo la mejora constante del producto.

Como se mencionó anteriormente, en este documento se realizaron diferentes tipos de pruebas de software, las cuales fueron:

- *Pruebas unitarias*: Son bloques de código que permiten probar el funcionamiento de otro bloque pequeño de código dentro de la aplicación, en general dentro del proyecto se mantuvo un porcentaje de código cubierto por pruebas del 80% cómo se pactó con el cliente, y se hicieron pruebas unitarias con entradas de datos típicos, entradas periféricas o datos limites y entradas erróneas, procurando que el código escrito se probará de la mejor manera posible por parte del equipo de desarrolladores.
- *Pruebas de contrato*: Estas pruebas se hacen con la finalidad de asegurar que los servicios y la aplicación se puedan comunicar de manera correcta con los datos de entrada y respuesta pactados en los contratos, garantizando que la comunicación entre los frentes

back-end y front-end funcione correctamente y mitigando la aparición de problemas de integración en la aplicación.

- *Pruebas de componentes*: Aseguran que el funcionamiento de los componentes de software fluya correctamente a través de cada interacción que pueda hacer el usuario, sirve para automatizar algunas pruebas manuales por ejemplo en el front-end ya que se pueden verificar flujos completos en diferentes pantallas de la aplicación garantizando que el sistema siga los pasos esperados a la acción que se realiza.

Estas pruebas de software ayudan a generar más seguridad en el cliente, ya que este estaba informado de ellas y exigieron que fueran realizadas, y aunque suman tiempo en el desarrollo y al ser varias pruebas que se deben desarrollar fue estimado su esfuerzo dentro de cada historia de usuario siguiendo la metodología de puntos de SCRUM.

Como limitantes se encuentra que el tiempo es un recurso indispensable para generar un *timeline* hacia una aplicación mejor, por eso el equipo es consciente de que aunque la aplicación se encuentre en producción aún hay mejoras que hacer y funcionalidades que agregar.

VII. CONCLUSIONES

La buena organización de un proyecto, desde su idea hasta su materialización son las mejores bases para empezar correctamente un desarrollo que pueda ser culminado de manera exitosa. En un equipo de desarrollo tener claras las funciones y responsabilidades de cada uno de los miembros del equipo es clave para llevar a cabo la implementación de los objetivos, y mantener la calidad aunque conlleve a mayores esfuerzos es parte crucial para la satisfacción de los clientes y la disminución de errores en ambientes productivos.

Se logra cumplir con todos los objetivos propuestos, gracias a una buena gestión de los puntos planteados anteriormente, desde el análisis previo a la situación del cliente hasta el desarrollo de las aplicaciones siguiendo y cuidando los estándares de calidad pactados, resaltando la importancia de definir de manera clara y oportuna metodologías a seguir y planes de acción para las situaciones que se puedan presentar en el desarrollo del proyecto.

Con el desarrollo de las pruebas de software se logra probar gran parte de la aplicación, para garantizar que este funcione de manera correcta, cubriendo mínimamente un 80% de las

líneas de código de las carpetas del proyecto para todos los frentes (web, mobile, backend) cumpliendo con generar una aplicación con buenos estándares de calidad, liderado por el frente de aseguramiento de la calidad del equipo.

Cómo mejora futura se plantean varias funcionalidades adicionales con el cliente que están siendo desarrolladas, priorizando las necesidades del cliente procurando que la aplicación evolucione como el cliente lo desea, adicionalmente, se podría pensar en mejorar el tema de reportería adicionando modulos de analitica de datos lo que enriquecería mucho la información exportada desde la aplicación para el usuario.

REFERENCIAS

[1] Julia Martins. (2024, Febrero 15). Scrum: conceptos clave y cómo se aplica en la gestión de proyectos [Online]. Available: https://asana.com/es/resources/what-is-scrum

[2] Onofre Araneda. Procesos SCRUM. [Imagen] (2022, septiembre 26). Disponible: https://atenos.com/agile/como-funciona-scrum/

[3] Alberto Blanch, Fernán García, Fernando Fuentes, Manuel León, Marco Lozano, María García, Sergio Arias and Susana Juan. (2023, marzo 13). Qué es Behavior Driven Development (BDD) [Online]. Available: https://www.arsys.es/blog/behavior-driven-development#:~:text=Behavior%20Driven%20Development%20(BDD)%2C,experimentar%20al%20interactuar%20con%20ella.

[4] Edu Salguero (2018, Junio 22). Arquitectura Hexagonal [Online]. Available: https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

[5] Edu Salgero. *Arquitectura Hexagonal*. [Imagen] (2018, junio 22). Disponible: https://miro.medium.com/v2/resize:fit:720/format:webp/1*yR4C1B-YfMh5zqpbHzTyag.png,

[6] Mehmet Ozkaya. Vertical slices. [Imagen] (2023, Febrero 15). Disponible: https://medium.com/design-microservices-architecture-with-patterns/the-problem-with-clean-architecture-vertical-slices-111537c0ffcb

[7] Autor (2024). Métricas SonarQube.

[8] Autor (2024). Estrategia de branching.

[9] Autor (2024). Timeline.