



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

**Implementación del proceso de pruebas para API
REST aplicado a un proyecto de software de una
entidad bancaria**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Cluny Judiht ROJAS BARRAZA

ASESOR

Dra. Luzmila Elia PRÓ CONCEPCIÓN

Lima, Perú

2022



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Rojas, C. (2022). *Implementación del proceso de pruebas para API REST aplicado a un proyecto de software de una entidad bancaria*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Software]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Cluny Judiht Rojas Barraza
Tipo de documento de identidad	DNI
Número de documento de identidad	70434410
URL de ORCID	https://orcid.org/0000-0002-2993-5104
Datos de asesor	
Nombres y apellidos	Luzmila Elisa Pró Concepción
Tipo de documento de identidad	DNI
Número de documento de identidad	08862360
URL de ORCID	https://orcid.org/0000-0003-0622-1173
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Norberto Ulises Román Concha
Tipo de documento	DNI
Número de documento de identidad	08510560
Miembro del jurado 1	
Nombres y apellidos	Pablo Jesús Romero Naupari
Tipo de documento	DNI
Número de documento de identidad	06182185
Miembro del jurado 2 (Asesor)	
Nombres y apellidos	Luzmila Elisa Pró Concepción
Tipo de documento	DNI
Número de documento de identidad	08862360
Datos de investigación	
Línea de investigación	No aplica

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2022
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Software

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 19:10 horas del día 25 de agosto del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Lic. Román Concha Norberto Ulises (Presidente), Lic. Romero Naupari Pablo Jesús (Miembro) y la Dra. Pró Concepción Luzmila Elisa (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/nvd-sypk-tbu>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DEL PROCESO DE PRUEBAS PARA API REST APLICADO A UN PROYECTO DE SOFTWARE DE UNA ENTIDAD BANCARIA”**, por la Bachiller **Rojas Barraza Cluny Judiht**; para obtener el Título Profesional de Ingeniera de Software.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó a la Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

La Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, la Bachiller obtuvo la nota de **DICIENUEVE (19)**.

A continuación el Presidente de Jurados el Lic. Román Concha Norberto Ulises, declara a la Bachiller **Ingeniera de Software**.

Siendo las 19:57 horas, se levantó la sesión.

Presidente

Lic. Román Concha Norberto Ulises

Miembro

Lic. Romero Naupari Pablo Jesús

Miembro Asesor

Dra. Pró Concepción Luzmila Elisa



Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería Software

INFORME DE EVALUACIÓN DE ORIGINALIDAD
N° 025-EPISW-FISI-2022

1. Autoridad Académica que emite el Informe de Originalidad:	Directora de la Escuela Profesional de Ingeniería de Software
2. Apellidos y Nombres de la autoridad académica:	Dra. Nora Bertha La Serna Palomino
3. Operador del programa informático de similitudes:	Dra. Nora Bertha La Serna Palomino
3. Documento evaluado:	Tesis para Pregrado Título: "Implementación del Proceso de Pruebas para Api Rest Aplicado a un Proyecto de Software de una Entidad Bancaria"
5. Autores del documento:	Rojas Barraza, Cluny Judiht
6. Fecha de recepción de documento	Recepción: 10/09/2022
7. Fecha de aplicación del programa detector de similitudes:	Revisión: 10/09/2022
8. Software utilizado:	Turnitin
9. Configuración del programa detector de similitudes:	Excluye textos entrecomillados: Sí Excluye biografías: Sí Excluye cadenas menores a 40 palabras: Sí Otro criterio (especificar): No
10. Porcentaje de similitudes según programa detector de similitudes	Ocho por ciento (8%)
11. Fuentes originales de las similitudes encontradas	Se adjuntan en 02 (dos) fojas al presente informe
12. Observaciones:	Ninguna
13. Calificación de originalidad i. Documento cumple criterios de originalidad, sin observaciones ii. Documento cumple criterios de originalidad, con observaciones iii. Documento no cumple criterios de originalidad.	Documento cumple criterio de originalidad, sin observación
14. Fecha del Informe:	14/11/2022

Dra. Nora Bertha La Serna Palomino
Directora (e) de la EPISW

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

**Implementación del proceso de pruebas para API REST aplicado a un Proyecto de
Software de una Entidad Bancaria**

Autor: Bach. Rojas Barraza, Cluny Judiht

Asesor: Dra. Pró Concepción, Luzmila Elisa

Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de
Ingeniero de Software

Fecha: Mayo de 2022

RESUMEN

Las pruebas para las API no son las mismas que las aplicadas en aplicaciones Front, pero tienen igual o mayor importancia. Una prueba adecuada de estos componentes garantizará una asequible y menos compleja integración con el frontal. Además, permite que los errores se identifiquen en las primeras etapas de desarrollo. En estos últimos años en la entidad bancaria se ha incrementado el consumo de las API REST y, sin embargo, no tienen un proceso de prueba bien definido, ya que se centran en las pruebas de frontend. Es por eso que en el presente trabajo de suficiencia profesional vamos a desarrollar la definición e implementación de cada etapa del proceso de pruebas de la API REST, y de esta manera asegurar la calidad de estos componentes y por ende del producto final desarrollado en el proyecto Contratación de Multiproductos. Estas definiciones se basan en los conceptos, procedimientos y metodología desarrollada en ISTQB Foundation e ISTQB Agile Tester y el proyecto de software se desarrolla bajo los lineamientos de la metodología Scrum. Así mismo, se implementa la automatización de las pruebas de regresión, porque en esta entidad bancaria, cualquier aplicación o canal digital se comunica con el backend a través de la API, lo que genera un incremento en los reusos de estos componentes y, por tanto, una constante ejecución de pruebas de regresión.

Palabras claves: API (Interfaz de programación de aplicaciones), ISTQB, Frontend, Backend, prueba de regresión, automatización de prueba, prueba funcional.

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS
FACULTY OF SYSTEMS AND INFORMATICS ENGINEERING
PROFESSIONAL SCHOOL OF SOFTWARE ENGINEERING

**Implementation of the testing process for API REST applied to a software project
of a banking entity**

Author: Bach. Rojas Barraza, Cluny Judiht

Advisor: Dra. Pró Concepción, Luzmila Elisa

Title: Professional Sufficiency Work to opt for the Professional Title of Software Engineer

Date: May 2022

ABSTRACT

The tests for APIs are not the same as those applied to Front applications, but they are of equal or greater importance. Proper testing of these components will ensure an affordable and less complex integration with the front end. It also allows bugs to be identified in the early stages of development. In recent years in the bank, the consumption of REST APIs has increased and yet they do not have a well-defined testing process, as they focus on front-end testing. That is why in this professional sufficiency work we are going to develop the definition and implementation of each stage of the REST API testing process, and thus ensure the quality of these components and therefore the final product developed in the Multiproduct Contracting project. These definitions are based on the concepts, procedures and methodology developed in ISTQB Foundation and ISTQB Agile Tester and the software project is developed under the guidelines of the Scrum methodology. Likewise, the automation of regression tests is implemented, because in this bank, any application or digital channel communicates with the backend through the API, which generates an increase in the reuse of these components and, therefore, a constant execution of regression tests.

Key words: API (Application Programming Interfaces), ISTQB, Front-end, Back-end, regression test, test automation, functional test.

TABLA DE CONTENIDO

RESUMEN	iii
ABSTRACT	iv
INTRODUCCIÓN	1
CAPÍTULO I - TRAYECTORIA PROFESIONAL	3
1.1. Presentación profesional	3
1.2. Experiencia Laboral	3
1.3. Formación Académica Profesional	5
1.4. Formación Académica Complementaria	6
1.5. Certificaciones Internacionales	7
CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA..	8
2.1. Empresa - Actividad que realiza	8
2.2. Visión	8
2.3. Misión	8
2.4. Organización de la empresa	9
2.5. Área, cargo y funciones desempeñadas	9
2.5.1. Área y Cargo	9
2.5.2. Funciones desempeñadas	9
2.6. Experiencia Profesional realizada en la Organización	10
CAPÍTULO III - ACTIVIDADES DESARROLLADAS	12
3.1. Situación Problemática	12
3.1.1. Definición del problema	12
3.2. Solución	12
3.2.1. Objetivos	13
3.2.2. Alcance	13
3.2.3. Etapas y metodología	13
3.2.4. Fundamentos utilizados	15
3.2.4.1. Metodologías Tradicionales	15
3.2.4.2. Metodología Agiles	16
3.2.4.3. Metodologías Tradicionales vs Metodologías Agiles	17
3.2.4.4. Scrum	19
3.2.4.5. ISTQB (International Software Testing Qualifications Board)	23
3.2.4.6. Herramientas y Tecnologías	28

3.2.5.	Implementación de las áreas, procesos, sistemas y buenas prácticas ...	30
3.3.	Evaluación.....	78
3.3.1.	Evaluación Económica	78
3.3.2.	Evaluación Técnica.....	79
CAPÍTULO IV. REFLEXIÓN CRÍTICA DE LA EXPERIENCIA		80
CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES		81
5.1.	Conclusiones.....	81
5.2.	Recomendaciones	81
5.3.	Fuentes de Información.....	82

INDICE DE TABLAS

Tabla 1:	Experiencia Profesional.....	3
Tabla 2:	Formación académica profesional.....	5
Tabla 3:	Formación académica complementaria.....	6
Tabla 4:	Certificaciones internacionales	7
Tabla 5:	Tabla comparativa de metodologías.....	17
Tabla 6:	Equipo Scrum del proyecto	35
Tabla 7:	Listado de API nuevas.....	37
Tabla 8:	Listado de API Modificadas.....	37
Tabla 9:	Listado de API de Reúso	38
Tabla 10:	Listado de canales para el proyecto.....	39
Tabla 11:	Evaluación Económica.....	78
Tabla 12:	Tabla comparativa de la evaluación técnica.....	79

INDICE DE FIGURAS

Figura 1:	Organigrama de la empresa	9
Figura 2:	Modelo del flujo de trabajo del proyecto	14
Figura 3:	Actividades del proceso de pruebas	15
Figura 4:	Fases metodología en cascada	15
Figura 5:	Línea de tiempo metodologías	16
Figura 6:	Valores del Manifiesto Ágil.....	16
Figura 7:	Principios del Manifiesto Ágil.....	17
Figura 8:	Modelo de tablero Kanban.....	18
Figura 9:	Principales características de XP.	18
Figura 10:	Flujo de Scrum para un sprint.....	19
Figura 11:	Planificación del Sprint	20
Figura 12:	Descripción de los Roles de Scrum	22
Figura 13:	Actividades del proceso de pruebas	25
Figura 14:	Estructura funcional de Postman	29
Figura 15:	Línea de tiempo de la API	30
Figura 16:	Modelo del API REST.....	31
Figura 17:	Estructura de una petición/solicitud.....	33
Figura 18:	El Proyecto y la interacción con otros equipos	35
Figura 19:	Flujo de tareas de la Planificación de pruebas	36
Figura 20:	Arquitectura del API REST	37
Figura 21:	Flujo para obtener token de autenticación	38
Figura 22:	Flujo de tareas del Análisis y Diseño de pruebas.....	40
Figura 23:	Definición de Historia de usuario en JIRA	41
Figura 24:	Ficha técnica – Servicio que registra la solicitud de una tarjeta.....	41
Figura 25:	Documento D520 – Transacción backend del servicio multicanal.....	42
Figura 26:	Catalogo API Tarjetas	43
Figura 27:	Documento Funcional del servicio “Registrar solicitud de la tarjeta”.....	43
Figura 28:	Documento C204, sección de casos de prueba	45

Figura 29: Documento C204, sección de Resumen.....	45
Figura 30: Caso de pruebas de Obligatoriedad.....	46
Figura 31: Casos de prueba de Valor Inválido	47
Figura 32: Casos de prueba de Longitud de campo.....	48
Figura 33: Casos de prueba Funcionales	48
Figura 34: Diseño de casos de prueba priorizados	49
Figura 35: Entornos de Prueba.....	50
Figura 36: Flujo de tareas de la implementación de pruebas.....	50
Figura 37: Especificación de casos de prueba API tarjetas – registro de solicitud	51
Figura 38: Configuración del ambiente Test	52
Figura 39: Configuración del ambiente Calidad	52
Figura 40: Estructura de las colecciones para las pruebas de API.....	53
Figura 41: Configuración de datos en Postman.....	53
Figura 42: Datos de entrada en Postman	54
Figura 43: Pre requisitos del servicio Registrar la solicitud de la tarjeta.	55
Figura 44: Implementación test script de validación - Caso de prueba de Obligatoriedad	55
Figura 45: Implementación test script de validación - Caso de prueba de Valor inválido	56
Figura 46: Implementación test script - Caso de prueba de Longitud de campo	56
Figura 47: Estructura de un caso de prueba	57
Figura 48: Test script - Caso de prueba funcional “Registrar solicitud de tarjeta”	58
Figura 49: Test script - Caso de prueba Funcional “Listar solicitud de la tarjeta”	59
Figura 50: Colecciones de API implementadas	59
Figura 51: Flujo de tareas de la ejecución de pruebas	60
Figura 52: Ejecución del servicio de autenticación	61
Figura 53: Ejecución del servicio “listar datos del cliente”	61
Figura 54: Ejecución de la transacción backend	62
Figura 55: Ejecución del servicio “Registrar la solicitud de la tarjeta”	62
Figura 56: Resultado de los test scripts de validación del caso de prueba.....	63
Figura 57: Ejecución del servicio “Listar solicitud de la tarjeta”	63

Figura 58: Subcarpeta del caso de prueba	64
Figura 59: Ventana de configuración	64
Figura 60: Resumen de ejecución	65
Figura 61: Opción para ejecutar la colección API.....	65
Figura 62: Ventana de configuración de la colección API	66
Figura 63: Resumen de ejecución de la colección API	66
Figura 64: Ciclo de vida del error/defecto.....	68
Figura 65: Reporte de un error por obligatoriedad	68
Figura 66: Reporte de un error por valor Inválido.....	69
Figura 67: Reporte de un error funcional	69
Figura 68: Registro de tickets de error	70
Figura 69: Sección de resumen del documento de evidencias.....	71
Figura 70: Sección de evidencias de la prueba.....	72
Figura 71: Lista de documentos de evidencia	72
Figura 72: Historia de usuario de las pruebas de no impacto API Tarjetas	73
Figura 73: Firma digital del documento de especificación de pruebas	74
Figura 74: Pruebas de regresión API Tarjetas	74
Figura 75: Visto Bueno de los canales impactados	75
Figura 76: Ticket de atención para certificación de pruebas de rendimiento.....	76
Figura 77: Vistos buenos para pase a producción.	77
Figura 78: Repositorio de entregables de prueba	78

INTRODUCCIÓN

La entidad bancaria a lo largo de cada año prioriza proyectos que ayuden a mejorar los procesos de contratación de activos y productos financieros desde sus diferentes canales (Banca por Internet, oficina bancaria, aplicativo móvil, ATM Cajero automático, zona pública).

El proyecto en el que se aplica el presente trabajo, consiste en diseñar e implementar una carretera única para el proceso de contratación de los productos financieros (préstamos, tarjetas de crédito, etc.), que involucre a todos los canales digitales y de oficina. De esta manera, permite a los clientes y no clientes iniciar una comunicación por un canal y pueda continuarla por otra de una manera natural y transparente.

Para el desarrollo de un proyecto, cada aplicación o canal digital se comunica con el backend a través de la API. Es ahí donde resalta la importancia de un buen análisis, diseño, desarrollo y pruebas de estos componentes.

El presente trabajo de suficiencia profesional desarrolla la definición e implementación de cada etapa del proceso de pruebas de la API REST y de esta manera asegurar la calidad de estos componentes y por ende del producto final del proyecto Contratación de Multiproductos

Actualmente, la empresa no tiene un procedimiento definido para las pruebas de API REST, por lo que en el presente trabajo de suficiencia profesional se irá describiendo cada una de las actividades de pruebas definidas e implementadas en el proyecto y basados en los conceptos y procedimientos de ISTQB Foundation, ISTQB Agile Tester y Scrum. Este último para la gestión del proyecto y el trabajo colaborativo y ágil del equipo.

El presente trabajo se estructura como sigue a continuación:

En el CAPITULO I, se describe la experiencia profesional que el autor ha adquirido a lo largo de los años, del más reciente al más antiguo. Se detalla cada rol desempeñado y las funciones realizadas. Así mismo se describe la formación académica y complementaria adquirida por el autor.

En el CAPITULO II, se describe la información de la empresa donde se realiza este trabajo, su visión, misión y estructura organizacional. La autora describe en detalle

el área, cargo y funciones desempeñadas, así como la experiencia profesional adquirida en esta empresa.

En el CAPITULO III, es la parte resaltante del presente documento donde se detalla todo el trabajo desarrollado. Se expone la situación problemática, la solución, metodología, procedimientos y fundamentos utilizados en la definición e implementación del proceso de pruebas de API REST.

En el CAPITULO IV, se describe la reflexión crítica de la experiencia, puntos resaltantes y de importancia que se presentaron a lo largo del desarrollo del presente trabajo y los aportes que la autora tiene frente a ellos.

En el CAPITULO V, la parte final del documento donde se listan y detallan las conclusiones, recomendaciones y fuentes de información que dejaron el desarrollo del presente trabajo.

CAPÍTULO I - TRAYECTORIA PROFESIONAL

1.1. Presentación profesional

La autora del presente trabajo cuenta con grado académico Bachiller en Ingeniería de Software egresada de la Universidad Nacional Mayor de San Marcos. Con experiencia en análisis, desarrollo y sobre todo en Aseguramiento y Control de Calidad del Software.

A lo largo de estos años se ha desempeñado en los sectores de Banca y Finanzas en proyectos que han ayudado a mejorar los flujos de contratación de productos financieros, aplicando sus conocimientos, experiencia y demostrando ser una profesional comprometida, responsable, proactiva, ordenada y con gran capacidad de gestión en entornos cambiantes.

1.2. Experiencia Laboral

A continuación se describe la experiencia profesional del autor, especificando las funciones, la empresa y el periodo en el que laboro.

Tabla 1

Experiencia laboral

Empresa / Institución	Funciones y principales tecnologías	Periodo
ENELGY Cliente Entidad Bancaria Cargo: Analista de Aseguramiento y Control de Calidad.	Proyectos de Contratación productos Financieros en canales digitales y de oficina Funciones - Realizar el análisis funcional y técnico del aplicativo. - Preparar de la estrategia de prueba. - Elaborar el plan de pruebas. - Realizar el diseño de casos de prueba. - Realizar la especificación de casos de prueba. - Preparar los datos de prueba. - Ejecutar las pruebas de humo. - Implementar Test script de validación como parte de la automatización del API Testing. - Documentar evidencias. - Realizar análisis, ejecución e informe de pruebas de rendimiento con JMeter. - Gestionar y participar en el proceso certificación en pruebas de rendimiento.	Febrero 2020 - Actualidad

	- Gestionar y dar seguimiento del proceso de pase a producción. - Realizar el análisis para la automatización de pruebas de sistema (frontend) y de regresión con herramientas como selenium, cucumber.	
	Principales Tecnologías	
	- Java, JavaScript, Postman, Jmeter, Teslink, Git (Bitbucket), Atenea, Figma, Jira.	
ENTELGY	Área Arquitectura de Servicios	Abril 2019
Cliente Entidad Bancaria	Funciones	– Enero 2020
Cargo: Gestor y tester de servicio	- Realizar el despliegue de las API en entornos de Desarrollo, Test y Calidad. - Gestionar las configuraciones requeridas para la funcionalidad de las API (cifrados, gestión de errores, urls backend, proxys, certificados). - Identificar rutinas y dependencias asociadas al servicio. - Validar el servicio y elaborar el documento de prueba. - Dar Soporte durante las pruebas del Frontend. - Generar códigos de cambio Remedy. - Gestionar pases productivos. - Dar soporte y revisar incidencias post productivas. - Validar servicios de autenticación en canales nuevos.	
	Principales Tecnologías:	
	- Java, Git (Bitbucket), Jenkins, Nagios, Dimension, Remedy, Mongo Log, Atenea, Jira	
EXPERIS	Funciones:	Septiembre 2017 –
Cliente Equifax	- Elaborar plan de pruebas.	
Cargo: Analista de Aseguramiento de Calidad del Proyecto de Migración del Sistema Experto	- Definir casos de prueba. - Gestionar y preparar ambientes de prueba. - Revisar el cumplimiento de las métricas de Sonar y Fortify. - Elaborar Test Junit basados en los casos de prueba definidos. - Analizar e implementar aplicativos que permitan la automatización de las pruebas. - Gestionar tareas apoyados en la herramienta Jira. - Pruebas de rendimiento en componentes web y web services (soap/rest) con JMeter.	Abril 2019

	- Pruebas de Integración y Regresión del sistema Web Experto con Selenium.	
	Principales Tecnologías:	
	- Java, SVN, JUnit, Oracle 11g, Tomcat, Docker, Jenkins, Sonar, GoCD, Eclipse, WinSCP, Putty, SoapUI, Jmeter, Confluence.	
Quipucamayoc UNMSM	Funciones:	Junio
Cargo: Analista programador	- Dar mantenimiento a los sistemas Logísticos. - Optimizar las consultas SQL dentro de los sistemas Logísticos. - Realizar pruebas funcionales a los sistemas del módulo de Logística. - Atender consultas y resolver incidencias reportados por los usuarios. - Elaborar documentación técnica para el desarrollo de los sistemas - Análisis y diseño de requerimientos nuevos.	2015– Septiembre 2017
	Tecnologías:	
	- Java 7, Git, SQL, Toad for Oracle, Bizagi, Rational Rose, Trello, Redmine.	

Nota. En esta tabla se describe la empresa, las funciones desempeñadas y el periodo de trabajo del autor. Fuente: Elaboración propia.

1.3. Formación Académica Profesional

En la tabla 2 se describe la institución educativa, el grado académico y el periodo de estudios del autor

Tabla 2

Formación académica profesional

Institución / Centro de Estudios	Grado Académico	Periodo
Universidad Nacional Mayor de San Marcos Facultad de Ingeniería de Sistemas e Informática	Bachiller en Ingeniería de Software	Marzo 2012 – Julio 2017

Nota. Fuente: Elaboración propia.

1.4. Formación Académica Complementaria

En la siguiente tabla se lista los estudios complementarios obtenidos por la autora, donde se especifica la institución educativa, nombre del curso y el periodo de estudios.

Tabla 3

Formación académica complementaria

Institución / Centro de Estudios	Nombre del Curso	Periodo
Agile Learning	Automatización de Pruebas Móviles	Marzo 2022
JB Group Enterprise	Automatización de Pruebas API REST con Rest Assured y Cucumber	Febrero 2022
Entelgy Adr-formacion	Contenedores Docker	Febrero 2021 – Abril 2021
JB Group Enterprise	ISTQB Agile Tester	Febrero 2021 – Marzo 2021
Entelgy Adr-formacion	Business Intelligence y Transformación Digital	Febrero 2020 – Abril 2020
New Horizons Perú	ISTQB Certified Tester Foundation Level	Julio 2019
JB Group Enterprise	Programa de Gestión Ágil Scrum Master Scrum Developer Scrum PO	Diciembre 2018 – Enero 2019
CIBERTEC	Java 8.0 Fundamentals Developer	Diciembre 2017
CIBERTEC	Fundamentos de ITIL para la Gestión De Servicios	Junio 2017

Nota. Fuente: Elaboración propia.

1.5. Certificaciones Internacionales

En la siguiente tabla se describe las certificaciones obtenidas por el autor, donde se indica la institución, el nombre de la certificación obtenida, el código que acredita la certificación y la fecha en la que se otorgó.

Tabla 4

Certificaciones internacionales

Institución	Certificación	Código	Fecha
CertiProf	Scrum Foundation Profesional Certificate (SFPC)	CLVPRQSSSY- SSGJHHGV- XHRCYMTRSY	Junio 2020
iSQI	ISTQB Certified Tester Foundation Level	19-CTFL- 168271-09	Noviembre 2019
SCRUMstudy	Scrum Developer Certified	679736	Enero 2019
SCRUMstudy	Scrum Master Certified	681047	Enero 2019
SCRUMstudy	Scrum Product Owner Certified	682373	Enero 2019

Nota. Fuente: Elaboración propia.

CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1. Empresa - Actividad que realiza

La compañía es una entidad financiera sólida con una enorme reputación en el ámbito nacional e internacional. Está autorizada por la Superintendencia de Banca, Seguros y AFP (SBS) para operar al amparo de la Ley N° 26702, la Ley General de los Sistemas Financieros y de Seguros y la Ley Orgánica de la SBS, que establece un marco regulatorio para las empresas que operan en el sector financiero y los sistemas de seguro.

Desde hace más de 20 años en el Perú, brindando productos y servicios financieros acordes a las necesidades del usuario a través de una experiencia sencilla y conveniente, así mismo ofrece asesoría a sus clientes para ayudarlos a tomar las mejores decisiones financieras. Opera a través de una red distribuida en todo el país, con su sede central en el distrito de San Isidro.

En el ámbito tecnológico, aporta un modelo digital compuesto por tres aspectos básicos: el primero asociado al contenido, que es lo que se entrega como venta; el segundo es la experiencia del cliente, que es cómo se presenta y consume; y el tercero es la plataforma tecnológica, que determina la producción y la distribución.

Una de las prioridades tecnológicas de la entidad es crear productos de forma global utilizando componentes reutilizables, con el objetivo de aumentar la eficiencia, acelerar el time to market y crear una experiencia de usuario unificada en todos los países en los que opera esta entidad financiera.

2.2. Visión

La entidad financiera define su visión como la búsqueda continua de un futuro mejor para las personas, comprende que el desarrollo de su negocio está íntimamente relacionado con la prosperidad de las personas, por lo que aspira a que, como empresa, su trabajo contribuye al desarrollo de construyendo un futuro mejor para todos.

2.3. Misión

La entidad bancaria define su misión enfocada principalmente en entregar las mejores soluciones para sus clientes, un crecimiento rentable y sustentable para sus accionistas y en las sociedades en las que opera.

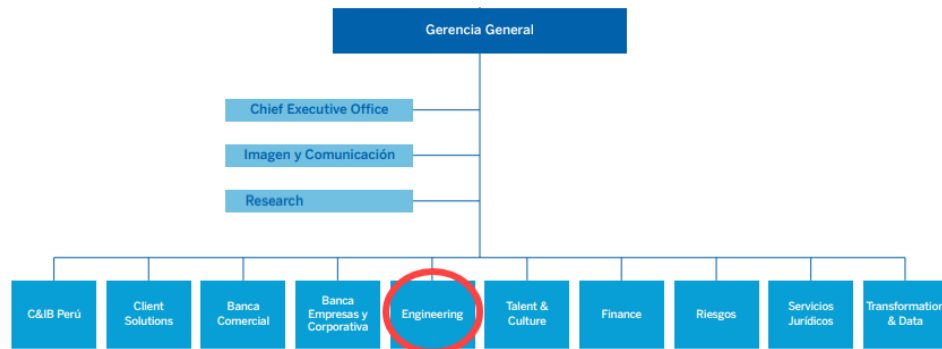
2.4. Organización de la empresa

La empresa en Perú está estructurada por el directorio, la gerencia general y las áreas descritas en la figura 1.

Ingeniería, es el área donde se desarrolla el proyecto descrito en el presente trabajo.

Figura 1

Organigrama de la empresa



Nota. La figura representa la estructura orgánica de la empresa. Fuente: (Empresa, 2021).

2.5. Área, cargo y funciones desempeñadas

2.5.1. Área y Cargo

Las actividades descritas en el presente documento de suficiencia profesional se desarrollaron en el proyecto contratación de multiproductos de la entidad bancaria. La autora se desempeñó bajo el rol de Analista de aseguramiento y control de calidad.

2.5.2. Funciones desempeñadas

Las funciones realizadas por la autora se detallan a continuación:

- Realizar el análisis funcional del negocio.
- Realizar el análisis técnico de la arquitectura y solución propuesta para el proyecto.
- Preparar la(s) estrategia(s) de prueba.
- Realizar la planificación de las pruebas.
- Analizar y diseñar casos de prueba.
- Especificar los casos de prueba.
- Preparar los datos de prueba.

- Ejecutar pruebas de humo.
- Ejecutar pruebas funcionales.
- Implementar test scripts de validación como parte de la automatización del API Testing.
- Gestionar y dar seguimiento de los defectos o errores reportados.
- Documentar las evidencias de prueba.
- Ejecutar pruebas de regresión automatizadas.
- Analizar, ejecutar y realizar el informe de pruebas de rendimiento con JMeter.
- Gestionar y participar en el proceso certificación en pruebas de rendimiento.
- Gestionar y dar seguimiento del proceso de pase a producción.

2.6. Experiencia Profesional realizada en la Organización

La autora trabaja actualmente como Analista de Aseguramiento y Control de Calidad en el proyecto Simplificación de tarjeta garantizada. El proyecto se encuentra en la fase inicial, por lo que se está participando en la revisión y análisis de la solución planteada. La experiencia y conocimiento adquirido previamente en otros proyectos, permite a la autora sugerir alternativas de solución y resolver dudas del funcionamiento de algunos componentes.

Antes del proyecto previamente mencionado, la autora participo en dos proyectos de gran impacto e interés para el negocio (Product owners y Sponsors), de los cuales se pasa a describir a continuación.

Primero, el proyecto contratación digital de activos financieros, cuyo objetivo es mejorar y simplificar el proceso de contratación de activos (tarjetas de crédito, préstamos) que manejan los agentes, ejecutivos y especialistas que trabajan en las oficinas de la entidad financiera. Este flujo incluye la participación del área de Riesgos, encargada de dar la conformidad a las condiciones establecidas y formalizar la solicitud generada por los agentes y ejecutivos de oficina. El sistema desarrollado en este proyecto ya está en uso.

Segundo, el proyecto contratación de multiproductos, cuyo objetivo es diseñar e implementar una solución que permita tener una sola vía de contratación de productos financieros (tarjetas de crédito, préstamos, cuentas) desde los diferentes canales que maneja la entidad bancaria. La solución se llevó inicialmente con el canal Zona Pública, permitiéndole al cliente acceder a la plataforma web, ingresar datos básicos (documento de identidad y fecha de nacimiento) y ser evaluado con respuesta inmediata, considerando que si es aprobado recibirá una lista(s) de oferta(s) y con ello poder completar la contratación del producto financiero brindado (tarjeta de crédito), sin necesidad de acercarse a una oficina bancaria. La plataforma web ya está productiva y cualquier usuario puede acceder desde la página principal de la entidad financiera.

Al tener ambos proyectos un flujo complejo y de integración con distintas áreas y/o canales, la autora tuvo que ser minuciosa y detallista para armar una estrategia de prueba adecuada para cada proyecto. De esta manera, asegurar la calidad en las pruebas de las API REST. Así mismo se implementaron test scripts de validación para la automatización de las pruebas de regresión.

La autora y profesional del presente trabajo inicio en esta entidad financiera como gestor de servicios en el área de Arquitectura de servicios, teniendo como parte de sus funciones el despliegue de las API en ambientes previos (Desarrollo, Test y Calidad), gestionar las configuraciones externas requeridas para la funcionalidad de las API tales como cifrados, gestión de errores, urls backend, proxys y certificados. Esta experiencia le sirvió para adquirir un mayor conocimiento técnico de la construcción y las configuraciones de estos componentes. Es por ello que es solicitada y promovida a proyectos que desarrollan a mayor detalle la funcionalidad de negocio de cada API.

CAPÍTULO III - ACTIVIDADES DESARROLLADAS

3.1. Situación Problemática

La institución bancaria es una gran comunidad que aprovecha el intercambio y el conocimiento de sus sucursales más relevantes para crear y entregar soluciones globales para los clientes basadas en capacidades diseñadas desde una perspectiva comercial global para implementar componentes que les permitan reutilizar y desarrollar.

Basados en esta definición, la entidad bancaria a lo largo de cada año prioriza proyectos que ayuden a mejorar los procesos de contratación de activos y productos financieros desde sus diferentes canales, tales como Banca por Internet, oficina bancaria, aplicativo móvil, ATM Cajero automático, zona pública entre otros.

Es así que para el desarrollo de un proyecto, cada aplicación o canal digital se comunica con el backend a través de las API. Es ahí donde resalta la importancia y la necesidad de realizar la definición e implementación de un proceso de pruebas para API REST. De esta manera permitir asegurar la calidad de estos componentes y por ende del producto final de un proyecto.

3.1.1. Definición del problema

3.1.1.1. Problema Principal

El problema principal se basa en que los proyectos de tecnología desarrollados en esta entidad financiera, no tienen definido un procedimiento adecuado para realizar las pruebas de las API REST. Por esta razón, en el presente trabajo se define e implementa un proceso de pruebas ajustado a la necesidad del proyecto de software.

3.1.1.2. Problemas Secundarios

En la entidad financiera, el crecimiento potencial de la construcción y uso de estas API genera altos impactos entre un canal/área y otro. Lo que provoca que se incrementen las pruebas de regresión y por ende el consumo de más tiempo y recursos.

3.2. Solución

En base a lo especificado en la situación problemática, se definen a continuación los objetivos generales y específicos desarrollados en este trabajo.

3.2.1. Objetivos

3.2.1.1. Objetivo General

Implementar el proceso de pruebas para API REST aplicado al proyecto de software Contratación de Multiproductos de una Entidad Bancaria.

3.2.1.2. Objetivos Específicos

- Definir la estrategia y el plan de pruebas.
- Diseñar y especificar los casos de prueba.
- Implementar test scripts de validación como parte de automatización.
- Ejecutar las pruebas.
- Evaluación de resultados y cierre de las pruebas.

3.2.2. Alcance

El alcance del presente trabajo involucra definir y realizar la implementación del proceso de pruebas para API REST aplicados en el desarrollo del proyecto Contratación de multiproductos de la entidad bancaria, el cual cubre la definición de la estrategia y plan de pruebas, el diseño, especificación, desarrollo y ejecución de las pruebas, valoración de los criterios de salida, informe sobre el resultado de las pruebas y finalmente el cierre de las pruebas. Actividades que aseguren la calidad de las pruebas API REST y por tanto del producto final.

Con respecto al alcance del proyecto se debe tomar en cuenta lo siguiente:

- Tarjetas de crédito será el producto con el que se da inicio el piloto.
- Zona Pública será el canal con el que se da inicio el piloto.

Por tanto la cantidad de API REST a probar se basa en el alcance del proyecto.

3.2.3. Etapas y metodología

El proyecto en el que se desarrolla este trabajo se gestiona bajo las buenas prácticas de la metodología SCRUM y está estructurada por cuatro PI (Program Increment) el cual se toma como referencia para especificar las etapas.

El proceso de pruebas se define apoyados en los conceptos de ISTQB Foundation e ISTQB Agile Tester.

1. Etapa inicial del Proyecto

En esta etapa se determina los equipos de trabajo del proyecto que será conformado por tres equipos SCRUM, cada uno liderado por su Scrum Master

y un Product Owner asignado a los tres equipos. Así mismo se lleva a cabo el kick off por parte del PO. Esta etapa se desarrolla dentro del PI-1.

2. Análisis y Planificación

En esta etapa se revisa el modelo solución y toda la documentación definida y presentada en la etapa inicial del proyecto.

Con respecto al proceso de pruebas, en esta etapa se define la actividad de Planificación de pruebas, donde se procede a definir y especificar la estrategia y plan de pruebas. Esta etapa es desarrollada como parte del PI-1.

3. Desarrollo e implementación

Esta etapa está conformada por las siguientes actividades que se desarrollaran de forma iterativa en cada sprint del PI-2 y PI-3.

- Análisis y diseño de pruebas
- Implementación de pruebas
- Ejecución de pruebas

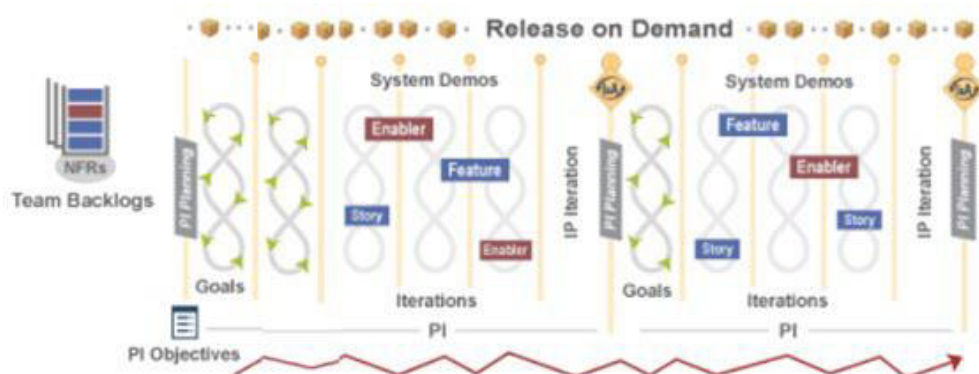
4. Gestión de QA para Producción

En esta etapa se realiza las gestiones necesarias para el pase a producción de las API REST. Esta etapa se realiza en el PI final del proyecto. Como parte del flujo de aseguramiento de calidad, previamente se deben ejecutar las pruebas de no impacto y de rendimiento.

En la siguiente imagen se visualiza el flujo de trabajo para cada PI que está conformada por 6 sprint y trabaja bajo los lineamientos de SCRUM.

Figura 2

Modelo del flujo de trabajo del proyecto

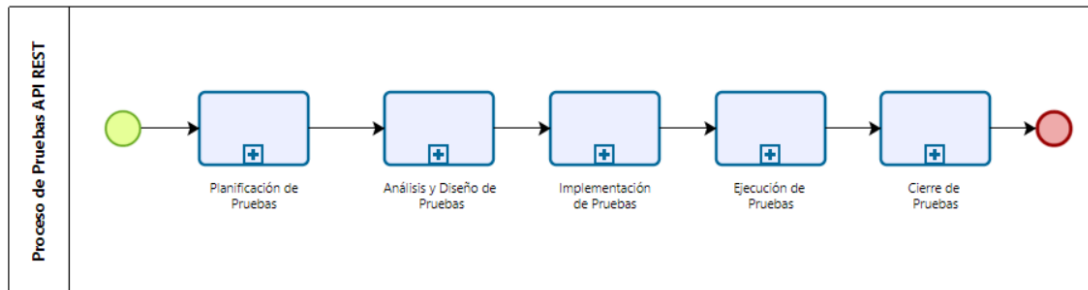


Nota. Fuente: Adaptada de Safe.

En la siguiente imagen se visualiza las actividades para la definición e implementación de las pruebas de API REST.

Figura 3

Actividades del proceso de pruebas



Nota. Fuente: Elaboración propia.

3.2.4. Fundamentos utilizados

3.2.4.1. Metodologías Tradicionales

Las metodologías tradicionales tienen una estructura en cascada donde el inicio de una fase está supeditado al término de la anterior. En este tipo de metodologías se prioriza la documentación detallada sobre el funcionamiento del software. En la siguiente imagen se listan las etapas que tiene esta metodología.

Figura 4

Fases metodología en cascada



Nota. La figura representa cada etapa que conforma una metodología en cascada.

Fuente: (Jordán Marcos, 2020).

Como se puede visualizar en la figura 2, la fase de pruebas es una de las ultimas en ser realizada, lo que genera que el costo de detectar los fallos sean más altos.

Cuando se trata de desarrollo de software, las metodologías tradicionales siempre han sido los más utilizados. Sin embargo, debido a los problemas con este método de trabajo, están siendo reemplazados gradualmente por un estilo de desarrollo ágil. (Jordán Marcos, 2020).

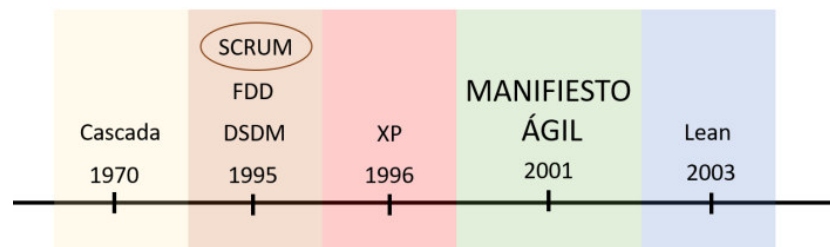
3.2.4.2. Metodología Ágiles

Las metodologías ágiles se definen como un conjunto de tareas y procedimientos orientados a diferentes campos, y que permite a las organizaciones y empresas gestionar sus proyectos con mayor rapidez y con mejores resultados.

De acuerdo a (Jordán Marcos, 2020), si se habla de metodologías ágiles, es más adecuado referirse a los frameworks ágiles. Estos marcos ágiles han existido antes, pero desde 2001 se incluyen bajo el mismo nombre a partir de la firma del Manifiesto Ágil. La siguiente figura muestra la línea de tiempo de estas metodologías.

Figura 5

Línea de tiempo metodologías



Nota. Fuente (Jordán Marcos, 2020)

Manifiesto Ágil: El manifiesto ágil está definido bajo cuatro valores claves y doce principios.

Figura 6

Valores del Manifiesto Ágil



Nota. La figura presenta la definición de los cuatro valores del manifiesto ágil.

Fuente: Adaptación de (Fowler, Marick, Beck, & Beedle, 2001).

Figura 7

Principios del Manifiesto Ágil



Nota. La figura presenta los doce principios del manifiesto ágil. Fuente: Adaptada de (Fowler, Marick, Beck, & Beedle, 2001).

3.2.4.3. Metodologías Tradicionales vs Metodologías Ágiles

En la siguiente tabla se muestra las diferencias entre la metodología tradicional y metodología ágil.

Tabla 5

Tabla comparativa de metodologías

	Ágil	Tradicional
Escala	Pequeños y medianos	Grandes
Orientación	Personas	Proceso
Preferencias	Adaptación	Predicción
Modelo de desarrollo	Iterativo, entrega evolutiva	Lineal
Requisitos	Dinámicos	Definidos al inicio
Equipo de trabajo	Auto-organizado, multidisciplinar	Especializados
Implicación de cliente	Alta	Baja
Planificación	Sprints	Exhaustiva
Tareas	Facilitador o Scrum Master	Gestor del proyecto
Documentación	Sólo la necesaria	Exhaustiva
Procesos	Aportan valor. Mejora continua	Control férreo.
Presupuesto	Abierto a cambios	Cerrado
Jerarquía	Equipo	Individual

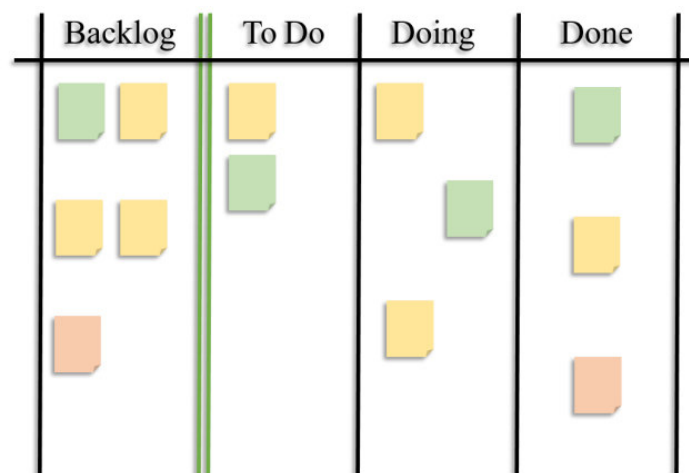
Nota. La tabla muestra aspectos que diferencian a la metodología tradicional de la ágil. Fuente: (Jordán Marcos, 2020).

A continuación una breve definición de las metodologías ágiles más populares en las organizaciones de desarrollo de software.

- i) **Kanban:** De acuerdo a (Jordán Marcos, 2020) “El objetivo principal de este enfoque es mostrar de forma rápida y visual todo el trabajo (tareas no iniciadas, tareas en progreso y tareas terminadas), así como mostrar las personas encargadas de cada una de las tareas. De esta manera, puede obtener una visión general del estado del proyecto sin sobrecargar a ningún miembro del equipo”. A continuación se muestra un tablero Kanban básico.

Figura 8

Modelo de tablero Kanban.



Nota. Fuente: (Jordán Marcos, 2020).

- ii) **Xtremme Programming (XP):** Esta metodología se usa en proyectos con tiempos de entrega cortos. La idea es hacer una programación rápida y con participación del usuario final en el equipo. A continuación se muestra las principales características de esta metodología. (Jordán Marcos, 2020).

Figura 9

Principales características de XP



Nota. Fuente: (Jordán Marcos, 2020).

3.2.4.4. Scrum

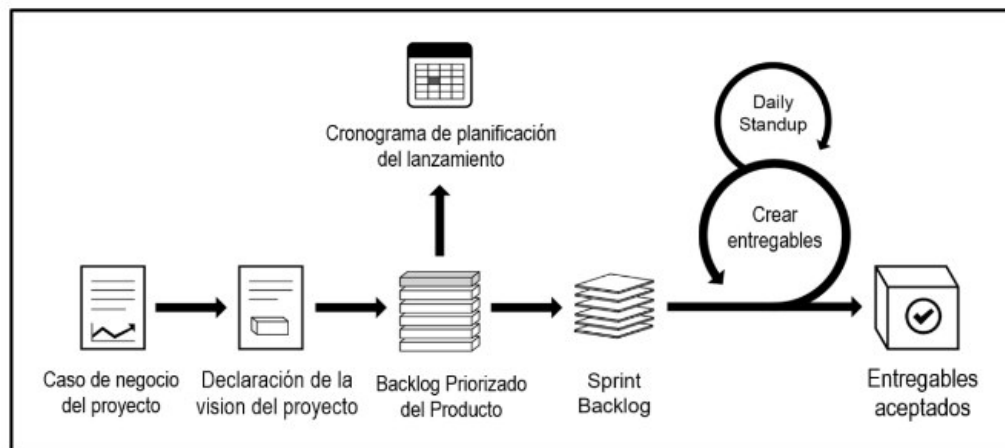
Es el framework ágil más popular. De acuerdo a (SCRUMstudy, 2017), “Scrum es adaptable, iterativo, rápido, flexible y eficaz, diseñado para ofrecer un valor considerable en forma rápida a lo largo del proyecto. Garantiza la transparencia en la comunicación y crea un ambiente de responsabilidad colectiva y de progreso continuo”.

Una ventaja clave de Scrum es que utiliza equipos multifuncionales, auto-organizados y empoderados que dividen su trabajo en ciclos de trabajo cortos y enfocados llamados Sprints. (SCRUMstudy, 2017).

En la siguiente figura se puede visualizar una visión general de flujo de un proyecto Scrum.

Figura 10

Flujo de Scrum para un sprint



Nota. En la figura se muestra la lista de historias priorizadas en el backlog y las que irán pasando al backlog del Sprint para ser ejecutado durante el sprint y generar los entregables al término de este. Fuente: (SCRUMstudy, 2017).

Aunque Scrum a menudo se considera un marco ágil de gestión de proyectos, también incluye un conjunto de reuniones, herramientas y funciones que ayudan a los equipos a organizar y gestionar su trabajo de forma coordinada. (Atlassian, 2022).

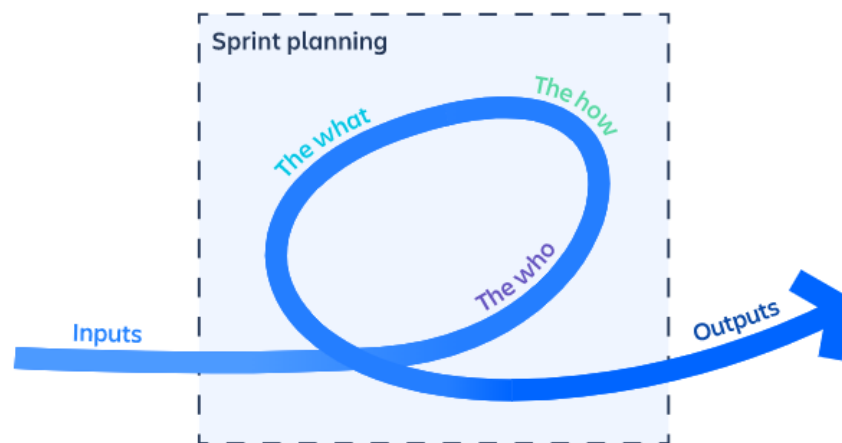
A continuación se definen los conceptos trascendentales de Scrum y con lo que se desarrolla el proyecto del presente trabajo.

A) Eventos Scrum

- **Sprint:** Un sprint es un período de tiempo corto y fijo durante el cual un equipo Scrum completa cierta cantidad de trabajo. (Atlassian, 2022).
- **Planificación del Sprint:** Este evento de Scrum inicia el sprint y se da en colaboración con todo el equipo. El objetivo es definir lo qué se puede entregar en un sprint y cómo se puede hacer para lograrlo. (Atlassian, 2022).

Figura 11

Planificación del Sprint



Nota. En la figura 3 se visualiza que la planificación del sprint está definida a identificar ¿El qué? Objetivo del sprint, ¿El Cómo? Priorizar y estimar HUs, y ¿El quien? PO y Equipo de desarrollo. Fuente: (Atlassian, 2022).

- **Scrum Diario:** Esta es una actividad de 15 minutos para el Equipo Scrum. Con el fin de reducir la complejidad, se realiza en el mismo lugar y hora todos los días hábiles del Sprint.

El objetivo del Daily Scrum es mejorar las comunicaciones, identificar impedimentos y facilitar la toma de decisiones rápida, eliminando así la necesidad de reuniones adicionales. (Schwaber & Sutherland, 2020).

- **Revisión de Sprint:** El propósito de este evento es examinar el resultado del Sprint y determinar futuros ajustes o cambios. El equipo Scrum presenta los resultados de su trabajo a las partes interesadas y analizan el progreso hacia los objetivos del producto. Se revisa lo que se logró en el Sprint y lo que ha cambiado en su entorno. (Schwaber & Sutherland, 2020)

- **Retrospectiva de Sprint:** El propósito de este evento es planificar formas de aumentar la calidad y la eficacia. El Equipo Scrum verifica el progreso del último Sprint en términos de personas, interacciones, procesos, herramientas y su definición de finalización. Reflexiona sobre los problemas encontrados y cómo se resolvieron (o no).

B) Roles de un proyecto Scrum

Roles Centrales: Los roles centrales definidos en Scrum son:

- **Product Owner:** De acuerdo a (SCRUMstudy, 2017), “es la persona responsable de maximizar el valor del negocio para el proyecto. Este rol es responsable de articular los requisitos del cliente y de mantener la justificación del negocio del proyecto. El Product Owner representa la voz del cliente”.
- **Scrum Master:** De acuerdo a (SCRUMstudy, 2017), “es un facilitador que asegura que el Equipo Scrum esté dotado de un ambiente propicio para completar con éxito el desarrollo del producto. Es el que guía, facilita y enseña las prácticas de Scrum a todos los participantes del proyecto, elimina los impedimentos que enfrenta el equipo y se asegura de que se estén siguiendo los procesos de Scrum”.
- **Equipo Scrum:** Es un grupo de personas responsables de entender los requisitos del negocio especificados por el Product Owner, de estimar las historias de usuarios y de la creación de los entregables del proyecto. (SCRUMstudy, 2017).

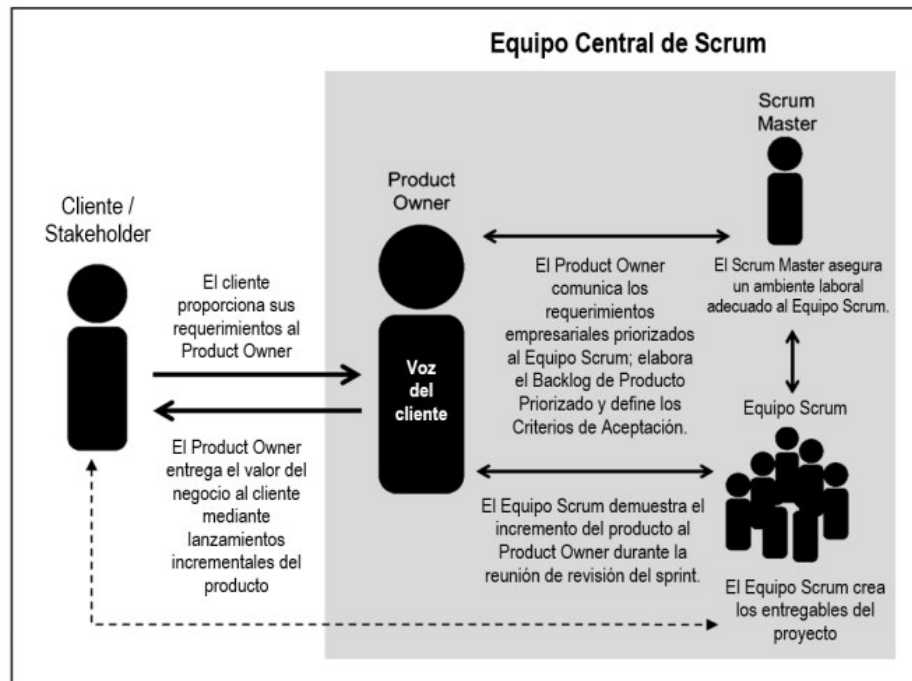
Roles No Centrales: A continuación se definen algunos roles no centrales descritos en SCRUM.

- **Stakeholder(s):** Este es un término general que incluye clientes, usuarios y patrocinadores que normalmente interactúan con el Product Owner, Scrum Master y Scrum Team, brindándoles información y facilitando la creación de productos, servicios u otros resultados del proyecto. (SCRUMstudy, 2017).

- **Patrocinador (Sponsor):** Es la persona o la organización que financia, provee recursos y ofrece el apoyo para la ejecución del proyecto. (SCRUMstudy, 2017).

Figura 12

Descripción de los Roles de Scrum



Nota. En la figura se visualiza la interacción entre los roles del equipo central de Scrum así como el PO interactúa con los roles no centrales. Fuente: (SCRUMstudy, 2017).

C) Incremento del Programa (Program Increment)

Un Incremento de programa (PI) es un intervalo durante el cual un equipo entrega valor incremental en forma de software y sistemas en funcionamiento y probados.

El PI suele durar de 8 a 12 semanas. Durante este tiempo, se deben planificar, construir y validar los incrementos completos del sistema, demostrando valor y obteniendo retroalimentación inmediata.

La entidad financiera tiene cuatro PI durante el año, en base a esta estructura se estima el desarrollo de los proyectos.

3.2.4.5. ISTQB (International Software Testing Qualifications Board)

De acuerdo a (TesteandoSoftware, 2014) ISTQB “Es una organización de certificación de la calidad del software que opera internacionalmente. El ISTQB fue fundado en noviembre de 2002 en Edimburgo y está legalmente registrado en Bélgica. Esta organización se encarga de soportar y definir un esquema de certificación internacional. Suministra el plan de estudios y el glosario sobre los que se definen los que se establecen las guías para la acreditación y evaluación de los profesionales del testing”.

ISTQB, tiene un conjunto de conceptos y programas, los cuales servirán de base para el desarrollo e implementación de proceso de pruebas API REST.

A) ISTQB Foundation

“ISTQB Foundation es la certificación de Nivel Básico dirigida a cualquier persona que se encuentre involucrada en la prueba de software. Incluidas las personas que asumen los roles de probadores, analistas de pruebas, ingenieros de pruebas, consultores de pruebas, jefes de prueba, probadores que participan en la prueba de aceptación y desarrolladores de software”. (ISTQB, 2018).

“El contenido del programa de estudio no es una descripción de toda el área de conocimiento de la prueba de software, sólo refleja el nivel de detalle que se cubrirá en los cursos de formación de Nivel Básico. Se centra en conceptos y técnicas de prueba que pueden aplicarse a todos los proyectos de software, incluidos proyectos Ágiles”. (ISTQB, 2018).

B) ISTQB Agile Tester

“Un probador en un proyecto ágil trabajará de forma diferente a uno que trabaje en un proyecto tradicional. En este programa los probadores deben comprender los valores y principios en los que se basan los proyectos ágiles, y cómo los probadores son parte integrante de un enfoque de equipo completo junto con los desarrolladores y representantes de negocio”. (ISTQB, 2018)

“Los miembros de un proyecto ágil se comunican entre sí desde el principio y con frecuencia, lo que ayuda a eliminar los defectos desde el principio y a desarrollar un producto de calidad”. (ISTQB, 2018).

C) Proceso de Pruebas

ISTQB define las pruebas como: "El proceso que consiste en todas las actividades del ciclo de vida, tanto estáticas como dinámicas, relacionadas con la planificación, la preparación y la evaluación de los productos de software y los productos de trabajo relacionados para determinar que satisfacen los requisitos especificados, para demostrar que son aptos para el propósito y para detectar defectos". (ISTQB, 2018).

De acuerdo a (Niittyviita, 2016), el proceso de pruebas en las primeras etapas del ciclo de desarrollo del software puede ayudar a evitar que se generen defectos en el código y estos a su vez provocar fallos en el sistema.

El proceso de pruebas que se va desarrollar en el presente trabajo sigue la siguiente estructura:

- **Control de Pruebas:** El control de pruebas, es la actividad continua que compara el progreso actual con el progreso planificado e informa del estado en el que se encuentra.
- **Planificación de pruebas:** La planificación de pruebas, es la actividad donde se definen los objetivos de prueba y se especifican las actividades de las pruebas.
- **Análisis y Diseño:** El análisis y diseño de pruebas es la actividad durante la cual se definen las condiciones y casos de prueba tangibles.
- **Implementación y Ejecución:** Es la actividad donde los procedimientos y scripts de pruebas son especificados incluyendo informaciones necesarias para la ejecución de las pruebas tales como los entornos y datos de prueba.
- **Evaluación de los criterios de salida y reporte de las pruebas:** En esta actividad, los resultados de la ejecución de la prueba se evalúan frente a los objetivos definidos en el plan de prueba.
- **Cierre:** En esta actividad se recoge los datos completados para consolidar la experiencia, obtener informes y estadísticas con respecto a los resultados. En esta actividad se genera lo que se conoce como Testware.

En la siguiente figura se visualiza cada una de las actividades del proceso de pruebas y la secuencia que siguen.

Figura 13

Actividades del proceso de pruebas



Nota. Fuente: Elaboración propia basada en ISTQB.

D) Pruebas de API

Las pruebas de API implican probar directamente las interfaces de programación de aplicaciones como parte de las pruebas de integración para determinar si cumplen con el resultado esperado de la funcionalidad, confiabilidad, rendimiento y seguridad.

Las pruebas de API son fundamentales para la automatización de las pruebas, porque la API es ahora la interfaz principal para la lógica de la aplicación y porque las pruebas de la interfaz gráfica de usuario son difíciles de mantener con los ciclos de lanzamiento cortos y los cambios constantes que se utilizan habitualmente con el desarrollo de software ágil y DevOps. (Inflectra, 2022).

E) Niveles de Prueba

- Pruebas Unitarias

De acuerdo a (ISTQB, 2018), “la prueba unitaria también conocida como prueba de componente o de modulo, se centra en los componentes que se pueden probar por separado”.

- **Pruebas de Integración**

De acuerdo a (ISTQB, 2018), “la prueba de integración se centra en las interacciones entre componentes o sistemas. Existen dos niveles de prueba de integración diferentes que se describen a continuación”.

- **Prueba de integración de componentes:** Estas pruebas se centran en las interacciones e interfaces entre los componentes integrados.

La prueba de integración de componentes ocurre después de la prueba de componentes y generalmente está automatizada. (ISTQB, 2018).

- **Prueba de integración de sistemas:** “Estas pruebas se centran en las interacciones e interfaces entre sistemas, paquetes y micro-servicios.

La prueba de integración del sistema se puede realizar después de la prueba o en paralelo con las actividades de prueba del sistema en curso (en desarrollo secuencial y desarrollo iterativo e incremental)”. (ISTQB, 2018).

- **Pruebas de Sistema**

“Las pruebas del sistema se centran en el comportamiento y la funcionalidad de todo el sistema o producto, por lo general considerando las tareas de extremo a extremo que el sistema puede realizar y los comportamientos no funcionales que exhibe mientras realiza esas tareas”. (ISTQB, 2018).

- **Pruebas de Aceptación**

“Las pruebas de aceptación, al igual que la prueba de sistema, generalmente se enfocan en el comportamiento y la funcionalidad de todo el sistema o producto”. (ISTQB, 2018)

Las pruebas de aceptación pueden generar información para evaluar si el sistema está listo para el uso por parte de los clientes (usuarios finales). (ISTQB, 2018).

Las pruebas de aceptación más comunes son:

- Prueba de aceptación de usuario.
- Prueba de aceptación operativa.
- Prueba de aceptación contractual y de regulación.

- Prueba alfa
- Prueba beta.

F) Tipos de Prueba

Un tipo de prueba es un conjunto de actividades de prueba diseñadas para probar una funcionalidad específica de un software o parte de ella basada en un objetivo de prueba específico.

Tomando como referencia el (ISTQB, 2018), “los tipos de pruebas son: pruebas Funcionales, pruebas No Funcionales, pruebas de caja blanca y pruebas asociadas al cambio; a su vez algunos tipos de prueba contiene subtipos de prueba”.

Para el desarrollo del presente trabajo definiremos los siguientes tipos y subtipos de prueba.

- Pruebas Funcionales

De acuerdo a (ISTQB, 2018), “la prueba funcional de un sistema incluye pruebas que evalúan la funcionalidad que debe realizar el sistema. Los requisitos funcionales se pueden describir en productos de trabajo, como las especificaciones del proyecto, requisitos de negocio, épicas, historias de usuario, casos de uso y/o especificaciones funcionales”.

- Pruebas No Funcionales

▪ Pruebas de Rendimiento

Las pruebas de rendimiento son aquellas que someten a un sistema a una carga de trabajo para medir su velocidad, confiabilidad y estabilidad bajo ciertas condiciones de operación.

- Pruebas Asociadas al cambio

▪ Pruebas de Confirmación

Llamadas también retesting, son aquellas pruebas que se realizan para revalidar errores reportados y que han sido corregidos.

De acuerdo a (ISTQB, 2018), “el objetivo de una prueba de confirmación es confirmar que el defecto original se ha solucionado de forma satisfactoria”.

▪ Pruebas de Regresión

El objetivo de estas pruebas es identificar posibles errores no deseados que se presenten al realizar una corrección o cambio en el software que ya había sido ejecutado previamente por un conjunto de casos de prueba.

De acuerdo a (ISTQB, 2018), “los cambios realizados en una parte del código, ya sea una corrección u otro tipo de cambio, pueden afectar inadvertidamente el comportamiento de otras partes del código, ya sea dentro del mismo componente o en otros componentes del mismo sistema, o incluso en otros sistemas”.

3.2.4.6. Herramientas y Tecnologías

A) JIRA

Es un Sistema de gestión para flujos de trabajo que permite realizar el seguimiento de las tareas bajo cualquier escenario.

Jira permite dividir el trabajo en partes manejables, asignarlo a las personas adecuadas y trabajar con flujos de trabajo personalizables hasta su finalización. (Atlassian, 2022)

Con esta herramienta se puede agregar el estado, prioridad, comentarios y archivos adjuntos para asegurarse de tener toda la información correcta para hacer el trabajo. (Atlassian, 2022).

B) POSTMAN

Postman es una herramienta que permite construir y testear APIs como parte de sus principales funcionalidades, sin embargo se ha vuelto una plataforma muy potente que ofrece una variedad de funciones.

La siguiente figura muestra como está estructurada sus diferentes funcionalidades.

Figura 14

Estructura funcional de Postman



Nota. Fuente: (Postman, 2022)

Estas son algunas de las múltiples tareas que se puede realizar con Postman.

- Testing de colecciones o catálogos de API.
- Permite generar documentación de las API.
- Gestión de entornos de prueba.
- Implementación de scripts de validación, pre requisitos.
- Gestión de espacios de trabajo.

C) JAVASCRIPT

Es un lenguaje de programación que permite implementar funciones para el desarrollo de páginas web y aplicaciones.

La implementación de test scripts en Postman está basado en este lenguaje de programación. Postman proporciona una API de JavaScript que se puede usar en scripts de solicitud. Este objeto proporciona funcionalidad para probar datos de solicitud y respuesta.

D) JMETER

De acuerdo a la página oficial (Apache Software Foundation, 2022), Apache JMeter, “es un software de código abierto, una aplicación diseñada al 100% en Java, permite cargar el comportamiento funcional de prueba y medir el rendimiento”.

Inicialmente fue diseñado para probar aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba.

3.2.5. Implementación de las áreas, procesos, sistemas y buenas prácticas

Para iniciar con el desarrollo del proyecto, definiremos algunos términos que se mencionaran recurrentemente.

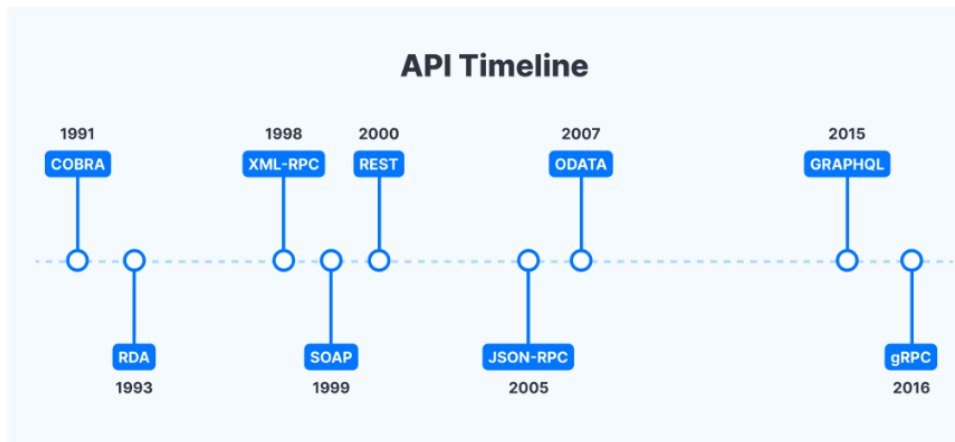
API

“La API o Interfaz de Programación de Aplicaciones proporciona funciones y reglas que permiten la interacción y comunicación entre diferentes aplicaciones. Estas interfaces facilitan la integración de aplicaciones, lo que permite a los desarrolladores crear potentes productos digitales”. (appmaster, 2022)

Una API utiliza varios protocolos y arquitecturas para enviar peticiones y respuestas de un aplicativo a otro. El presente trabajo se desarrolla basadas en API con arquitectura REST. (appmaster, 2022).

Figura 15

Línea de tiempo de la API



Nota. Fuente: (appmaster, 2022)

REST (Transferencia de estado representativo)

API REST y API RESTFUL

REST es un modelo de arquitectura web basado en el protocolo HTTP, utiliza implementaciones cliente-servidor y RESTful es la implementación de dicha arquitectura.

En resumen una API RESTful es un API que fue diseñada usando la arquitectura REST, para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

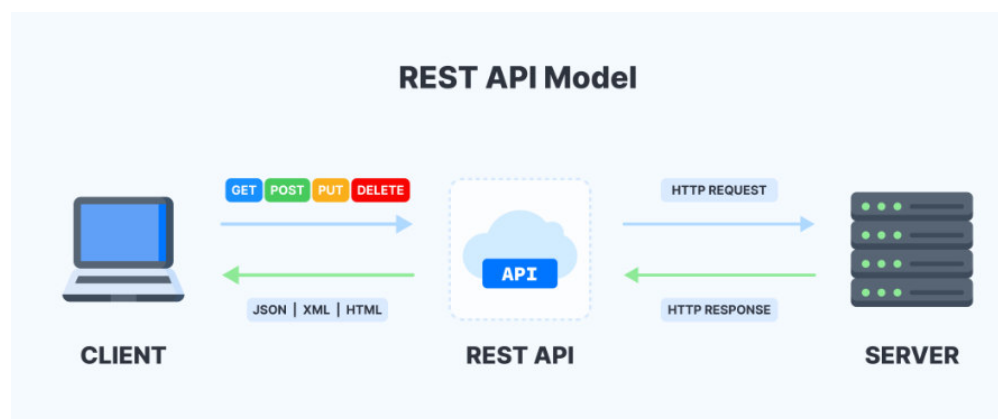
De acuerdo a (appmaster, 2022), los componentes principales de la API REST son:

- **Cliente:** Un programa (de lado del usuario), que lanza la petición desde un dispositivo e inicia la comunicación. Para nuestro caso la herramienta POSTMAN actuara como cliente.
- **Servidor:** Un programa que proporciona un conjunto de servicios y espera peticiones para proporcionar la información solicitada.
- **Recurso:** Cualquier contenido e información (documento, texto, imagen, servicio temporal) que el servidor transmite al cliente.

En la siguiente figura se visualiza el modelo de la API bajo la arquitectura REST.

Figura 16

Modelo del API REST



Nota. Fuente: (appmaster, 2022).

Los métodos y/u operaciones más importantes en cualquier sistema REST y especificación HTTP son:

- **POST:** Método que permite crear un recurso o ejecutar una operación compleja.
- **GET:** Método que permite obtener o recuperar un recurso.
- **PUT:** Método que permite actualizar un recurso.
- **PATCH:** Método que permite actualizar parcialmente un recurso
- **DELETE:** Método que permite eliminar un recurso.

Servicio Multicanal ASO

Servicio de negocio propio de la entidad financiera que pertenece a una API determinada y que puede utilizar métodos GET, POST, PUT, PATCH y/o DELETE para describir qué hacer. Puede ser consumido por cualquier canal que requiera su uso. Un servicio multicanal puede ser de tipo NUEVO, MODIFICACION y REUSO.

- **Nuevo:** Servicios que se construyen desde cero, debido a que ningún servicio del catálogo cubre la necesidad del proyecto interesado.
- **Modificación:** Servicios que requieren de ajustes adicionales para ser consumidos por el proyecto interesado.
- **Reúso:** Servicios que se ajustan a la necesidad o requerimiento del proyecto. Solo se deberá realizar una configuración de enganche hacia el canal del proyecto interesado.

En el Catálogo API de la entidad financiera, un servicio puede hacer referencia a un método HTTP aplicado a un recurso a través de su identificador único.

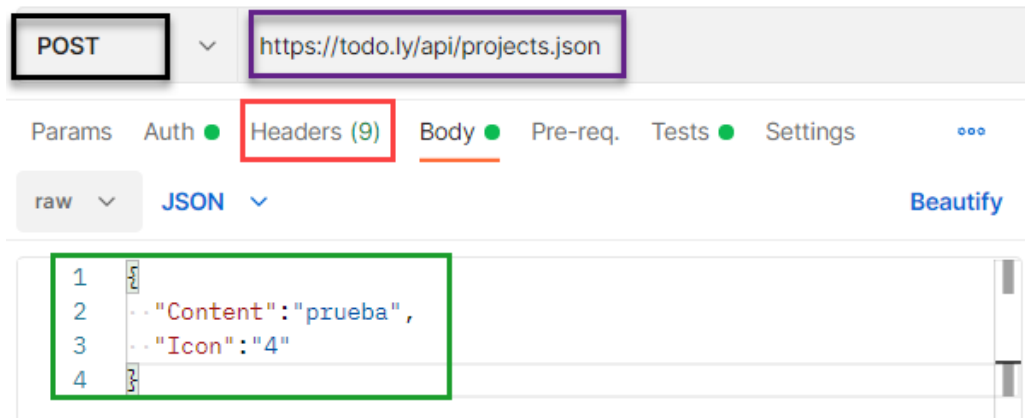
Es preciso mencionar que el cambio de cualquier servicio multicanal impactara en todos los canales que consumen el API al que corresponde el servicio multicanal ASO.

Para acceder al recurso, el cliente debe realizar una petición/solicitud. Al recibirlo, el servidor genera una respuesta que contiene datos codificados sobre el recurso.

La estructura de la petición consta de cuatro partes principales: **método HTTP**, **puntos finales**, **encabezado** (headers) y **cuerpo** (body).

Figura 17

Estructura de una petición/solicitud



Nota. En la figura se puede visualizar un ejemplo de la estructura de una petición que corresponde al recurso de una API de acceso abierto. Fuente: Adaptada de (todo.ly, 2020).

Canal

Se denomina al consumidor del API o servicio, la entidad financiera cuenta con diferentes canales entre las que se encuentra Banca por Internet, Banca Móvil, Oficina Bancaria, ATM entre otras.

El identificador y nombre técnico del canal es denominado AAP. Dependiendo del modelo solución propuesto se gestionara el alta de un nuevo canal para el proyecto.

Backend APX

El backend es la capa de acceso a datos de un software que no es directamente accesible por los usuarios, contiene la lógica de la aplicación. APX es el framework de desarrollo en Java para los aplicativos backend de la entidad financiera.

Backend HOST

Backend desarrollado bajo una arquitectura COBOL.

Arquitectura de Servicios

Área encargada del análisis, diseño, desarrollo y gestión del catálogo de APIs que maneja la entidad financiera. Está estructurada en diferentes equipos.

- **Equipo de Diseño ASO:** Equipo encargado del análisis y diseño del servicio y/o API solicitada. El diseñador asignado recibe el requerimiento y en conjunto con el equipo del proyecto definen los campos y recursos que se van a necesitar para cubrir el requerimiento solicitado. Previamente identifica en el catálogo si existe la API que podría cubrir total o parcialmente las necesidades del proyecto.
- **Equipo de Fabrica ASO:** Equipo encargado de construir o realizar el ajuste a la API. Se basan en los entregables y documentación que se generaron en la fase de análisis y diseño: Publicación en el API Catalogo (plataforma de la entidad financiera donde se encuentran todas las API), Ficha técnica (Documento en la que se detalla la estructura del servicio) y D520 (documento en el que se detalla la estructura del backend que va consumir el servicio ASO) que resulta del análisis y diseño previo.
- **Equipo de Gestores ASO:** Equipo encargado del soporte y gestión de la API, su tarea inicia en el momento que fabrica entrega la construcción y termina cuando la API se encuentra productiva.

Equipos Aplicativo o del Proyecto

Equipo SCRUM que pertenece al proyecto. En un proyecto puede existir más de un equipo. El equipo comúnmente está conformado por el Product owner (representante del negocio), Scrum master (facilitador del equipo) y el equipo de desarrollo (programadores, probadores, analistas BPM). A lo largo del desarrollo del proyecto el equipo interactúa con equipos de otros proyectos y áreas.

1. Etapa inicial del Proyecto

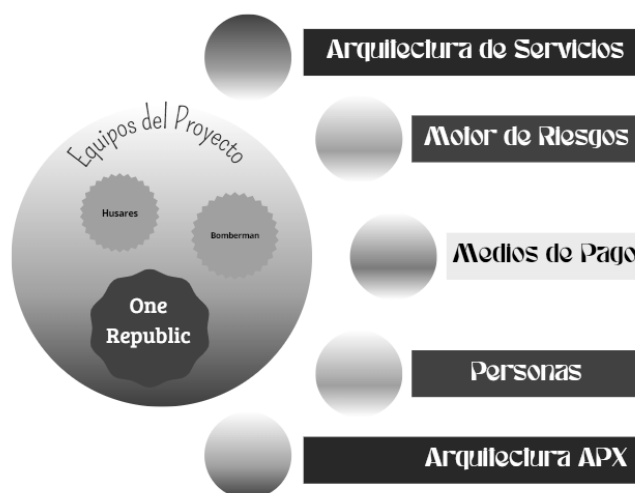
En esta etapa se determina los equipos de trabajo del proyecto que será conformado por tres equipos SCRUM, cada uno liderado por su Scrum Master y un Product Owner asignado a los tres equipos. Así mismo se lleva a cabo el kick off por parte del PO. Republic es el equipo SCRUM a la que pertenece la autora del presente trabajo, conformada de la siguiente manera.

Tabla 6*Equipo Scrum del proyecto*

Nombres	Rol	Observación
Pierre R.R	Product Owner	
Edwin P.Y.	Scrum Master	
Cluny Rojas	Analista QA	
Milton M.O.	Diseñador ASO	
Macg B.R.	Diseñador ASO	
Renzo V.V.	Developer	
Ricardo C.L.	Developer	
Javier P.I.	Developer	
Marcel H.C.	Developer	
Stevens J.M.	Developer	
José R.G.	Gestor ASO	Pertenecen al área de arquitectura de servicios, no participan activamente en las ceremonias de Scrum.
Dany N.O.	Gestor ASO	

Nota. Lista de integrantes del equipo scrum backend. Fuente: Elaboración propia.

Durante el desarrollo del proyecto se requiere de la interacción con otras áreas, ya sea como soporte o dependencia. En la siguiente imagen se muestra a los equipos del proyecto y la relación con otros equipos y Áreas.

Figura 18*El Proyecto y la interacción con otros equipos*

Nota. Fuente: Elaboración Propia.

2. Análisis y Planificación

Teniendo el modelo solución completo se mapea las API que se usaran en el desarrollo del proyecto, se identifican cuantas serán nuevas, modificadas y de reúso. Así mismo se identifica que se realizara la gestión para habilitar un canal nuevo que se denomina Zona Publica Multiproducto. Cada API nueva que se construye incluye el desarrollo de su backend, conocida también en la entidad financiera como Transacción. Este desarrollo lo cubre el equipo Scrum.

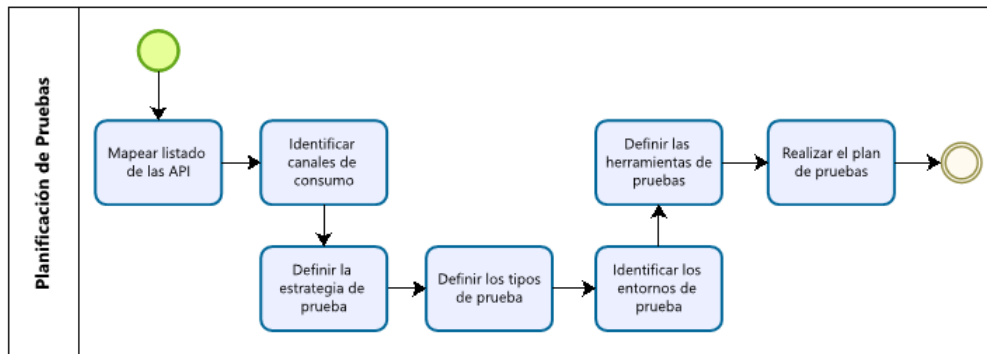
De acuerdo al proceso de pruebas definido previamente, en esta etapa inicia la actividad planificación de pruebas, donde definimos la estrategia y plan de pruebas.

2.1. Planificación de Pruebas

Es la actividad con la que inicia el proceso de pruebas y sigue el siguiente flujo definido.

Figura 19

Flujo de tareas de la Planificación de pruebas



Nota. Fuente: Elaboración Propia.

Para el desarrollo del proceso de pruebas de API REST en el proyecto se aplicara la integración de dos estrategias de prueba.

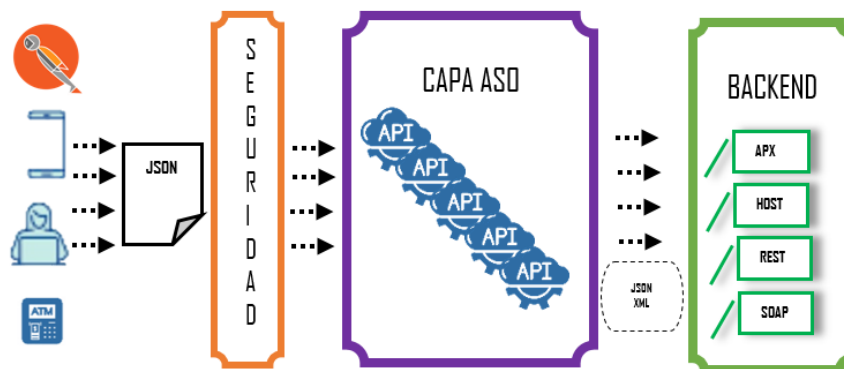
La estrategia Analítica basada en los requerimientos, porque contamos con una base de conocimiento tales como la documentación funcional, documentación técnica (ej. Ficha técnica del API, D520, plataforma del API Catalogo, etc.).

La estrategia de pruebas adversas a la regresión, para la implementación de pruebas de regresión automatizadas.

El alcance de las pruebas cubre la lista de las API asignadas al equipo Scrum Republic, las que dependiendo de su arquitectura pueden consumir los diferentes tipos de backend usados en la entidad bancaria.

Figura 20

Arquitectura del API REST



Nota. Fuente: Elaboración propia.

En la siguiente tabla se lista las API nuevas que se van a desarrollar, es decir que se construyen desde cero.

Tabla 7

Listado de API nuevas

API	Servicio Multicanal ASO	Método
cards	createCardProposal modifyCardProposal listCardProposals	POST PATH GET PUT
customers	listCustomerCivilRegistries	GET
geographicplaces	listGeographicPlaces	GET
images	searchImagesCovers	POST
proposals	createQuestionnairesValidateAccess	POST
riskAdmissions	createExclusionFiltersValidations	POST

Nota. Fuente: Elaboración propia.

En la siguiente tabla se visualizan las API que requieren de un ajuste para cubrir la necesidad del proyecto.

Tabla 8

Listado de API Modificadas

API	Servicio Multicanal ASO	Método
<u>pois</u>	<u>listPoisBranch</u>	GET
<u>products</u>	<u>listCardProducts</u>	GET

Nota. Fuente: Elaboración propia.

En la siguiente tabla se visualizan las API que ya existen y cubren la necesidad del proyecto

Tabla 9

Listado de API de Reúsos

API	Servicio Multicanal ASO	Método
businessCryptography	createCryptoDocument	POST
catalogs	listCatalogRecords	GET
customers	listCustomers	GET
	listCustomerAddresses	GET
	listContactsDetail	GET
riskEvaluationPerson	verifyRiskEvaluationPerson	POST

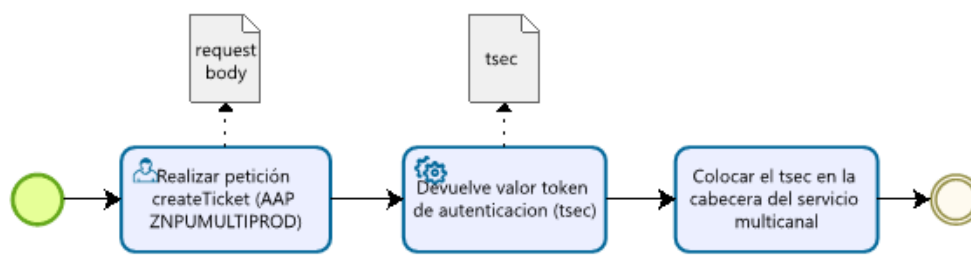
Nota. Fuente: Elaboración propia.

El proyecto usará dos canales de consumo que serán necesarios para la ejecución de las pruebas de cada una de las API, ya que se requiere de un token de autenticación que se invocara a través de un servicio de seguridad. Este servicio cuenta con una configuración propia dependiendo del canal de consumo.

Para obtener el token primero se invoca al servicio de autenticación colocando el request body (usuario, contraseña) que corresponde al canal seleccionado, este devolverá como salida el valor del token, finalmente ese valor se colocara en la cabecera de cada servicio multicanal que se probara.

Figura 21

Flujo para obtener token de autenticación



Nota. Fuente: Elaboración Propia.

Este token de autenticación tiene un tiempo de caducidad de 10 minutos por lo que se tiene que tomar en cuenta en la implementación de los casos de prueba.

En el modelo solución del proyecto se especificaron dos canales de consumo: uno correspondiente al área de arquitectura backend para el uso de servicios inter-backend. Y el segundo, un canal nuevo correspondiente al consumo exclusivo para el desarrollo del proyecto.

Tabla 10

Listado de canales para el proyecto

CODIGO AAP	NOMBRE AAP	TIPO
13000038	APX	EXISTENTE
13000076	ZNPUMULTIPROD (<i>Zona Publica para Contratación multiproducto</i>)	NUEVO

Nota. En esta tabla se visualizan los canales con las que se consumirán los servicios.

Fuente: Elaboración propia.

Las pruebas se aplicaran en el Nivel de Integración.

Las tipos de pruebas que se aplicaran serán:

- Pruebas de Integridad de campos
- Pruebas Funcionales
- Pruebas de Confirmación
- Pruebas de Regresión
- Pruebas de Performance

Los ambientes de prueba con los que se cuenta para la ejecución de pruebas son:

- **Test:** En este ambiente de prueba se realiza las pruebas de integridad de campos, pruebas funcionales y pruebas de confirmación.
- **Calidad:** En este ambiente de prueba se realiza las pruebas de integridad de campos, pruebas funcionales, pruebas de confirmación y pruebas de regresión.
- **Octa:** En este ambiente de prueba se realiza las pruebas de performance.
- **Liga Oculta:** Ambiente pre productivo.
- **Producción:** Ambiente productivo.

Las herramientas que se usarán son:

- **Jira:** Para el seguimiento del ciclo de desarrollo del API, gestión de errores y gestión de tareas del desarrollo del proyecto.

- **Postman:** Para las pruebas de integridad de campos, pruebas funcionales, pruebas de confirmación y automatización de las pruebas de regresión.
- **JMeter:** Para las pruebas de performance, pruebas de integración, pruebas funcionales.
- **C204:** Documento de banco para la definición de casos de prueba.
- **T907:** Documento de banco donde se coloca las evidencias de prueba.

3. Desarrollo e implementación

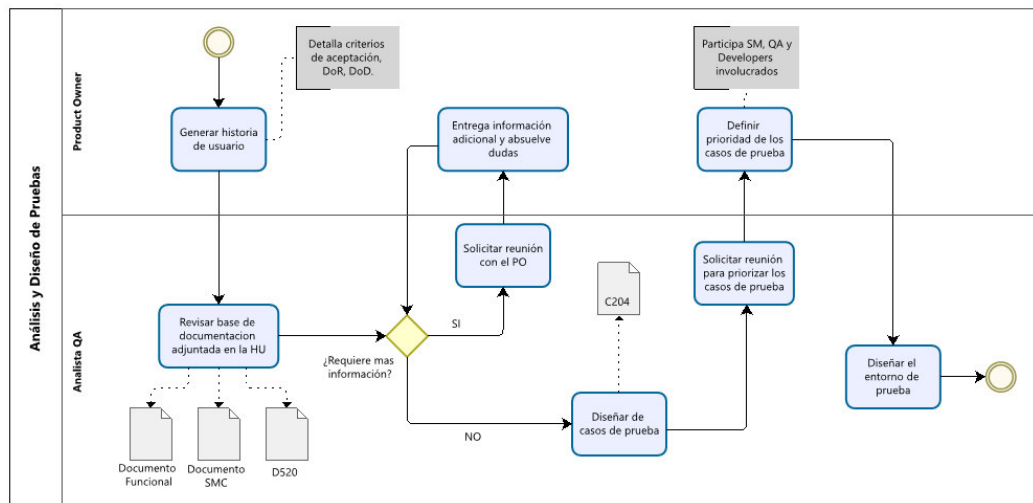
En esta etapa se inicia con las actividades de análisis y diseño de pruebas, así como la implementación y ejecución de pruebas, lo que a su vez se irán desarrollando de forma iterativa e incremental en cada sprint del PI (programa incremental).

3.1. Análisis y diseño de pruebas

En esta etapa del proceso de pruebas se realizara las siguientes actividades

Figura 22

Flujo de tareas del Análisis y Diseño de pruebas



Nota. Fuente: Elaboración propia.

3.1.1. Generación de Historias de Usuario de Prueba

En la reunión de planificación del equipo Scrum se seleccionan, priorizan y estiman las historias de usuario que se trabajara en cada Sprint. Las historias de usuario para diseño de pruebas tienen el siguiente modelo.

Figura 23

Definición de Historia de usuario en JIRA



Nota. En la imagen se visualiza una HU de diseño de casos de prueba donde se indica los criterios de aceptación, DoD, DoR. Fuente: Adaptada de Jira.

3.1.2. Revisión de Base de conocimiento

- **Documento SMC**, es la ficha técnica donde se detalla la estructura de los objetos y campos que tiene el servicio multicanal, indica que campos llevarán valores en cifrado, enmascarado u ofuscado, especifica el tipo del método, endpoint, url, cabeceras y parámetros. Así mismo tiene una sección de Errores donde se detalla si el servicio maneja gestión de errores.

Figura 24

Ficha técnica – Servicio que registra la solicitud de una tarjeta

SERVICIO MULTICANAL (SMC)						
DATOS CATALOGO API						
Ambito	Global	Categorías	PRODUCTS	API cards	Versión	1.162.0
DATOS TÉCNICOS						
Registry ID (SN)	cards	Logical ID (SN)		Registry ID (SMC)	createCardProposal	Logical ID (SMC)
EndPoint	POST	Método	URL Base			Paginación
	/cards/v1/proposals					No
Tipo	Acceso a backends	APX	Identificador de la operación backend			Descripción de la operación backend
						Operación que permite registrar la intención de contratar un producto
HEADERS						
Nombre	Tipo	Subtipo	Entrada / Salida	Obligatorio	Observaciones	
Header Data	ContactId	String	Entrada	Opcional	Esta cabecera se utiliza para canales en el flujo de	
	Location	String	Salida	Obligatorio	Por restricción técnica en APX cuando se crea un	
C. O. + Nombre					Entrada / Salida	Obligatorio Observaciones
- cardType			Entrada	Si		
- cardType.id			Entrada	Si		
- product			Entrada	Si		
- product.id			Entrada	Si		
- physicalSupport			Entrada	Si		
- physicalSupport.id			Entrada	Si		
- grantedCredits[n]			Entrada	No		
- grantedCredits[n].amount			Entrada	Si		
- grantedCredits[n].currency			Entrada	Si		
- contact<SPECIFIC>			Entrada	No		
- contact<SPECIFIC>.contactType			Entrada	Si		

Nota. En la figura se visualiza la ficha técnica de un servicio multicanal que corresponde al API de tarjetas. Fuente: (Empresa, 2020).

- **Documento D520**, donde se detallan los campos de la transacción backend desarrollada para ser consumida por el servicio multicanal. Este documento contiene una breve descripción de cada campo así mismo indica su tipo y longitud.

Figura 25

Documento D520 – Transacción backend del servicio multicanal

D520 - PP - createCardProposal - cards ☆ ⓘ ⓘ

o Editar Ver Herramientas Ayuda [Solicitar](#)

Ámbito	Nº SA	Título de Petición		Código :
APX		Registrar Solicitud		D520
Fecha Creación	Fecha Ult. Actualización	Versión No.	Creado por	País
24-09-2019	30-01-2020	2.1	Renzo Villavisencio	PERÚ

1. DEFINICIÓN

Nombre:	Registrar de Solicitud
Descripción Funcional:	Operación que permite registrar la intención de contratar un producto de la persona identificada

2. DATOS DE CONTEXTO

Nombre ¹	Descripción ²	Tipo ³	Lng ⁴	Obligatorio ⁵	Observaciones ⁶
CONTACTIDENTIFIER	Identificador de Contacto	String		No	

3. DATOS DE ENTRADA

Producto

Nombre ¹³	Descripción ¹⁴	Tipo ¹⁵	Lng ¹⁶	Obligatorio ¹⁷	Observaciones ¹⁸
cardType.id	Tipo de producto financiero identificador.	enum	11	Si	Valores - CREDIT_CARD - DEBIT_CARD - PREPAID_CARD
product.id	Identificador de la tarifa en la que aplica el usuario. (Multibin)	String	2	Si	Hace referencia al multibin de una tarjeta
product.subproduct	Subproducto	Object		No	No-disponible

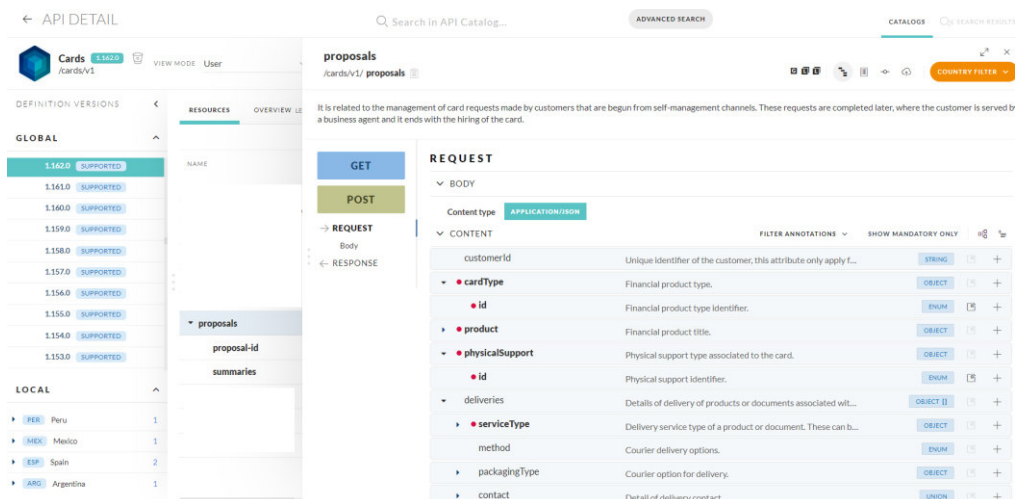
¹³ (Obligatorio) El nombre debe ser descriptivo y contener como máximo 7 caracteres si es host
¹⁴ (Obligatorio) Descripción del campo (que és y para que se usa)
¹⁵ (Obligatorio) Tipo de dato. Host: Alfanumérico(A), Numérico(N), Numérico con Signo(S); Distribuido: Utilizar tipos primitivos.
¹⁶ (Obligatorio) Longitud del dato para numérico con decimales usar la coma Ej 15.2

Nota. En la figura se visualiza el documento técnico del backend que consume el servicio multicanal “Registrar solicitud de la tarjeta”. Fuente: (Empresa, 2020).

- **Catalogo API**, donde está especificado cada recurso del API. Al igual que en la ficha técnica en esta plataforma se podrá revisar cada campo de entrada y salida que contiene el servicio, además ofrece un ejemplo de la estructura del request body (datos de entrada que se le envía al servicio) como también del response body (datos de salida que devuelve el servicio).

Figura 26

Catalogo API Tarjetas

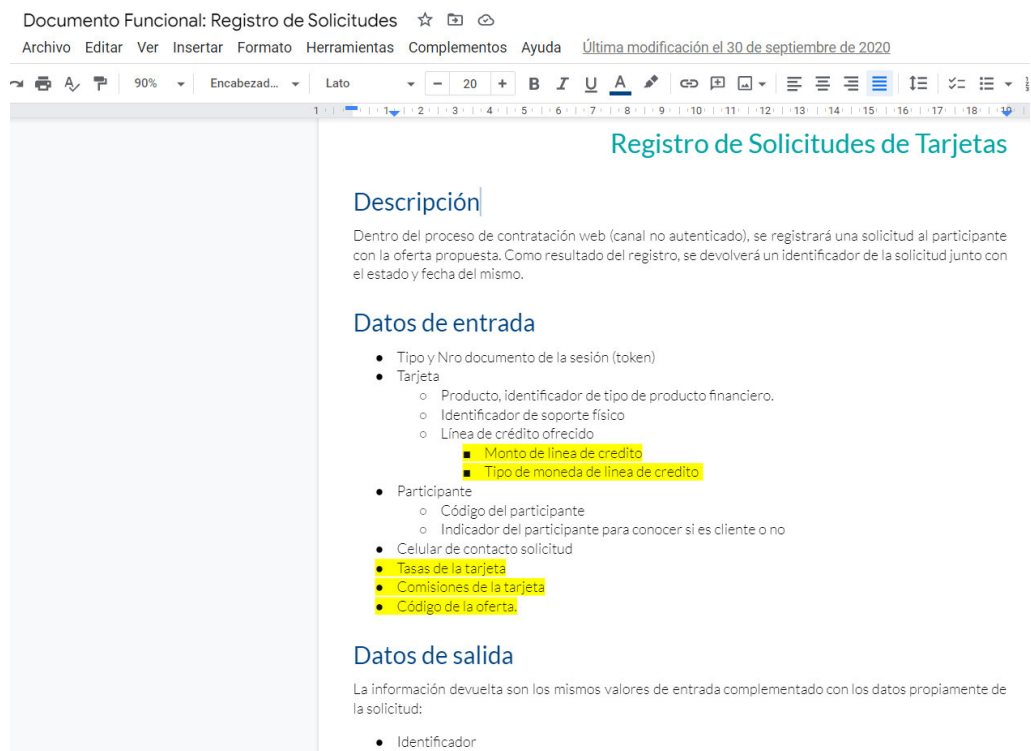


Nota. En la figura se visualiza datos del API tarjetas. Fuente: (Empresa, 2020).

- **Documento funcional**, donde está la especificación funcional del negocio.

Figura 27

Documento Funcional del servicio “Registrar solicitud de la tarjeta”



Nota. En la figura se visualiza el documento funcional realizado para el servicio que se encarga del registro de la solicitud de la tarjeta de crédito. Fuente: (Empresa, 2020).

3.1.3. Diseño de casos de prueba

Revisada la base de conocimiento se diseñan los casos de cada servicio multicanal, tomando en referencia el tipo de API (Nuevo, Modificado, Reúso).

Para el diseño y especificación de casos de prueba se usa el formato denominado C204 y que contiene las siguientes columnas:

- **CodCaso:** El código del caso de prueba está compuesto por CP (Caso de Prueba) + XXX (Numeración correlativa).
- **Caso de prueba:** En esta columna se coloca el nombre del caso de prueba, el cual es definido bajo la siguiente estructura.

Nombre AAP + Descripción corta del caso de prueba + Resultado esperado (OK=Satisfactorio / NOK=Mensaje de Error)

Ejemplo:

ZNPUMULTIPROD-cardType.id campo mandatorio-NOK

ZNPUMULTIPROD-rates objeto opcional-OK

ZNPUMULTIPROD-contact.contact<MOBILE>.number valor Invalido-NOK

- **Prioridad:** Una vez definidos los casos de prueba se determina con el PO y Scrum Master, la prioridad que tendrán cada uno de los casos de prueba. Esta prioridad puede ser ALTA, MEDIA o BAJA.

Estas primeras columnas del documento C204 se completan en la etapa de diseño de las pruebas. Los que vienen a continuación se completan en la etapa de implementación de las pruebas.

- **Paso:** Se define el procedimiento del caso de prueba, indicando el paso a paso para su ejecución.
- **Descripción:** Se describe a detalle el objetivo del caso de prueba.
- **Resultado Esperado:** Indica el valor esperado de la prueba. Puede ser un mensaje de error o el status code de la respuesta esperada.
- **Motivo de Prueba:** Se coloca el valor definido será API Testing.
- **Status:** Indica el estado del caso de prueba, inicialmente todos inician con el status “Pendiente de Ejecución”, luego dependiendo de la ejecución o el tiempo que tome la prueba puede tener un estado de “Satisfactorio”, “Insatisfactorio”, “En curso” o “En espera”.

El documento tiene dos secciones importantes, uno corresponde a la sección de casos de prueba donde se define y especifica cada caso de prueba.

Figura 28

Documento C204, sección de casos de prueba

Caso	Caso de prueba	Prioridad	Paso	Descripción	Resultado Esperado	Ciclo de Prueba	Motivo de Prueba	STATUS
CP0001	ZNPUMULTIPROD-Validación del servicio enviando los campos obligatorios para un h/c Cliente	Alta	- Pasos	Se necesita validar el servicio enviando los campos obligatorios para un	El servicio responde	1	API Testing	Pendiente Ejecución
CP0002	ZNPUMULTIPROD-Validación del servicio enviando los campos obligatorios para un Cliente-OK	Alta	- Pasos	Se necesita validar el servicio enviando los campos obligatorios para un	El servicio responde	1	API Testing	Pendiente Ejecución
CP0003	ZNPUMULTIPROD-Validación del servicio enviando un número de celular con formato	Alta	- Pasos	Se necesita validar el servicio enviando un número de celular con formato	El servicio responde	1	API Testing	Pendiente Ejecución
CP0004	ZNPUMULTIPROD-Validación del servicio enviando una tasa con identificador repetido-NOK	Alta	- Pasos	Se necesita validar el servicio enviando una tasa con identificador	El servicio responde	1	API Testing	Pendiente Ejecución
CP0005	ZNPUMULTIPROD-Validación del servicio enviando una comisión con identificador	Alta	- Pasos	Se necesita validar el servicio enviando una comisión con identificador	El servicio responde	1	API Testing	Pendiente Ejecución
CP0006	ZNPUMULTIPROD-Validación del servicio no enviando el contactid en la cabecera-NOK	Alta	- Pasos	Se necesita validar el servicio NO enviando el contactid en la cabecera	El servicio responde	1	API Testing	Pendiente Ejecución
CP0007	ZNPUMULTIPROD-Validación del servicio enviando un contactid vencido en la cabecera-NOK	Alta	- Pasos	Se necesita validar el servicio NO enviando un contactid vencido en la	El servicio responde	1	API Testing	Pendiente Ejecución
CP0008	ZNPUMULTIPROD-Validación del servicio enviando un tipo de detalle de contacto no	Alta	- Pasos	Se necesita validar el servicio enviando un tipo de detalle de contacto no	El servicio responde	1	API Testing	Pendiente Ejecución
CP0009	ZNPUMULTIPROD-Validación del servicio enviando un tipo de contacto no permitido-NOK	Alta	- Pasos	Se necesita validar el servicio enviando un tipo de contacto no permitido	El servicio responde	1	API Testing	Pendiente Ejecución
CP0010	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de	Alta	- Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0011	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa que posee más	Alta	- Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0012	ZNPUMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee	Alta	- Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0013	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa que posee más	Alta	- Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0014	ZNPUMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto	Alta	- Pasos	Se necesita validar el servicio enviando un monto de línea de crédito que	El servicio responde	1	API Testing	Pendiente Ejecución
CP0015	ZNPUMULTIPROD-Validación del servicio enviando un monto de línea de crédito que	Alta	- Pasos	Se necesita validar el servicio enviando un monto de línea de crédito que	El servicio responde	1	API Testing	Pendiente Ejecución
CP0016	ZNPUMULTIPROD-cardType objeto mandatorio-NOK	Media	- Pasos	Validar la obligatoriedad de cardType	Status Code: 400	1	API Testing	Pendiente Ejecución
CP0017	ZNPUMULTIPROD-cardType id campo mandatorio-NOK	Media	- Pasos	Validar la obligatoriedad de cardType id	Status Code: 400	1	API Testing	Pendiente Ejecución
CP0018	ZNPUMULTIPROD-product objeto mandatorio-NOK	Media	- Pasos	Validar la obligatoriedad de product	Status Code: 400	1	API Testing	Pendiente Ejecución
CP0019	ZNPUMULTIPROD-product id campo mandatorio-NOK	Media	- Pasos	Validar la obligatoriedad de product id	Status Code: 400	1	API Testing	Pendiente Ejecución

Nota. En la figura se visualiza el formato de definición de test cases. Fuente: (Empresa, 2020).

El segundo corresponde a la sección de resumen, donde se coloca el nombre del proyecto, la fecha de inicio y fin de la prueba, una tabla con los resultados de acuerdo al estado de cada caso de prueba y una tabla de versionamiento.

Figura 29

Documento C204, sección de Resumen

Ámbito de las Pruebas		Asignada	Título de la		Código
			Contratación Mutiprodueto Web TC		C204
Fecha Creación	Fecha Ult. Actualización	Versión No.	Creado por:	Pais	
16/06/2020	27/07/2020	1.5	Cluny Rojas	Perù	
Descripción					Inicio de pruebas
					16/06/2020
					Fin de pruebas
					27/07/2020
Resumen Resultados			Nº Casos	%	
Satisfactorio					
Insatisfactorio					
En Curso					
En Espera					
Pendiente Ejecución					
TOTAL			95	100	
Control de versiones					
Versión	Fecha	Autor Modificación	Descripción		
1	16/06/2020		Creación del documento		
1.5	27/07/2020		Actualización documento (+GABY)		

Nota. Fuente: (Empresa, 2020).

Las pruebas de una API nueva incluye un diseño detallado de los casos de prueba que incluye casos de integridad campos y funcionales de todos los campos y objetos especificados en la base de conocimiento.

Las pruebas de una API de modificación incluyen el diseño detallado de los casos de prueba de los campos y objetos que son parte del cambio o ajuste solicitado por el proyecto. Se consideran casos de integridad de campos y funcionales.

Las pruebas de una API de reúso incluyen el diseño de casos de prueba funcionales basados en el requerimiento del proyecto.

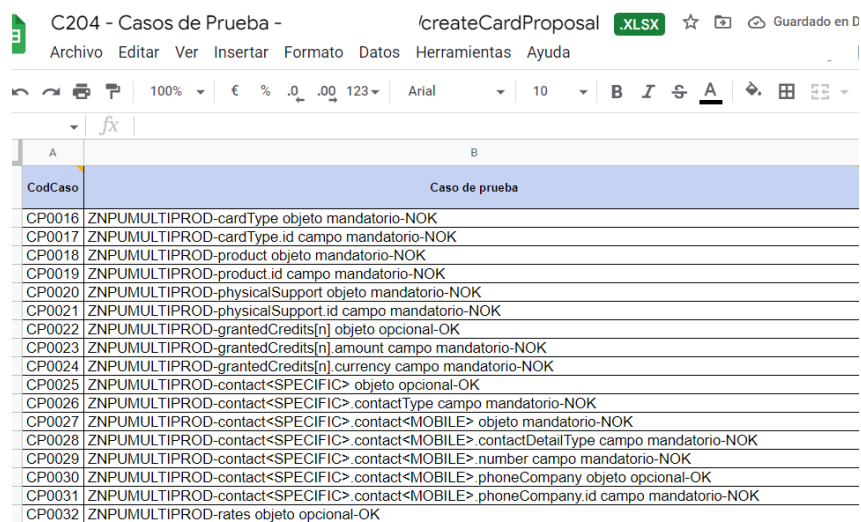
Pruebas de Integridad de Campos:

Los casos de integridad de campos o llamados también casos técnicos incluyen: casos de obligatoriedad, valor inválido y de longitud del campo.

- **Casos de Obligatoriedad:** En base a la especificación de la ficha técnica del servicio, se debe validar que el campo este cumpliendo lo especificado. En caso la columna indique SI, el servicio debe responder con un mensaje de error, validando que el objeto o campo es mandatorio para que el servicio responda satisfactoriamente. En caso la columna indique NO, el servicio deberá responder un status OK, debido a que no es necesario enviar dicho objeto/campo para que el servicio responda correctamente.

Figura 30

Caso de pruebas de Obligatoriedad



CodCaso	Caso de prueba
CP0016	ZNPUMULTIPROD-cardType objeto mandatorio-NOK
CP0017	ZNPUMULTIPROD-cardType.id campo mandatorio-NOK
CP0018	ZNPUMULTIPROD-product objeto mandatorio-NOK
CP0019	ZNPUMULTIPROD-product.id campo mandatorio-NOK
CP0020	ZNPUMULTIPROD-physicalSupport objeto mandatorio-NOK
CP0021	ZNPUMULTIPROD-physicalSupport.id campo mandatorio-NOK
CP0022	ZNPUMULTIPROD-grantedCredits[n] objeto opcional-OK
CP0023	ZNPUMULTIPROD-grantedCredits[n].amount campo mandatorio-NOK
CP0024	ZNPUMULTIPROD-grantedCredits[n].currency campo mandatorio-NOK
CP0025	ZNPUMULTIPROD-contact<SPECIFIC> objeto opcional-OK
CP0026	ZNPUMULTIPROD-contact<SPECIFIC>.contactType campo mandatorio-NOK
CP0027	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE> objeto mandatorio-NOK
CP0028	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.contactDetailType campo mandatorio-NOK
CP0029	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.number campo mandatorio-NOK
CP0030	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.phoneCompany objeto opcional-OK
CP0031	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.phoneCompany.id campo mandatorio-NOK
CP0032	ZNPUMULTIPROD-rates objeto opcional-OK

Nota. En la figura se muestra el diseño de casos de prueba de obligatoriedad para el servicio “Registrar solicitud de la tarjeta”. Fuente: Elaboración propia.

- **Casos de Valor Inválido:** En base a la especificación del tipo de campo, se valida que el servicio responda con un mensaje de error, validando que el campo no acepta valores diferentes al tipo con el que está especificado. En la figura se visualiza el diseño de casos de prueba de valor inválido para cada campo del servicio “Registrar solicitud de la tarjeta”.

Figura 31

Casos de prueba de Valor Inválido

CodCaso	Caso de prueba
CP0052	ZNPUMULTIPROD-cardType.id valor Invalido-NOK
CP0053	ZNPUMULTIPROD-product.id valor Invalido-NOK
CP0054	ZNPUMULTIPROD-physicalSupport.id valor Invalido-NOK
CP0055	ZNPUMULTIPROD-grantedCredits[n].amount valor Invalido-NOK
CP0056	ZNPUMULTIPROD-grantedCredits[n].currency valor Invalido-NOK
CP0057	ZNPUMULTIPROD-contact<SPECIFIC>.contactType valor Invalido-NOK
CP0058	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.contactDetailType valor Invalido-NOK
CP0059	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.number valor Invalido-NOK
CP0060	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.phoneCompany.id valor Invalido-NOK
CP0061	ZNPUMULTIPROD-rates.itemizeRates[n].rateType valor Invalido-NOK
CP0062	ZNPUMULTIPROD-rates.itemizeRates[n].itemizeRatesUnit<PERCENTAGE>.unitRateType valor Invalido-NOK
CP0063	ZNPUMULTIPROD-rates.itemizeRates[n].itemizeRatesUnit<PERCENTAGE>.percentage valor Invalido-NOK
CP0064	ZNPUMULTIPROD-fees.itemizeFees[n].feeType valor Invalido-NOK
CP0065	ZNPUMULTIPROD-fees.itemizeFees[n].itemizeFeeUnit<AMOUNT>.unitType valor Invalido-NOK
CP0066	ZNPUMULTIPROD-fees.itemizeFees[n].itemizeFeeUnit<AMOUNT>.amount valor Invalido-NOK
CP0067	ZNPUMULTIPROD-fees.itemizeFees[n].itemizeFeeUnit<AMOUNT>.currency valor Invalido-NOK
CP0068	ZNPUMULTIPROD-participants[n]<KNOW>.personType valor Invalido-NOK
CP0069	ZNPUMULTIPROD-participants[n]<KNOW>.id valor Invalido-NOK
CP0070	ZNPUMULTIPROD-participants[n]<KNOW>.isCustomer valor Invalido-NOK
CP0071	ZNPUMULTIPROD-participants[n]<KNOW>.participantType.id valor Invalido-NOK
CP0072	ZNPUMULTIPROD-offerId valor Invalido-NOK

Nota. Fuente: Elaboración propia.

- **Casos de Longitud de campo:** En base a la especificación de la longitud de campo, se valida que el servicio responda con un mensaje de error, validando que el campo no acepta ningún valor mayor a la longitud especificada. En la siguiente figura se visualiza el diseño de casos de prueba de longitud de campo del servicio “Registrar solicitud de la tarjeta”.

Figura 32

Casos de prueba de Longitud de campo

C204 - Casos de Prueba -		/createCardProposal	.XLSX	☆	📁	🔗
Archivo Editar Ver Insertar Formato Datos Herramientas Ayuda						
100% € % 0.00 123 Arial 10 B I S A						
fx						
A	B					
CodCaso	Caso de prueba					
CP0073	ZNPUMULTIPROD-cardType.id longitud mayor a 11-NOK					
CP0074	ZNPUMULTIPROD-product.id longitud mayor a 2-NOK					
CP0075	ZNPUMULTIPROD-physicalSupport.id longitud mayor a 14-NOK					
CP0076	ZNPUMULTIPROD-grantedCredits[n].amount longitud mayor a 18-NOK					
CP0077	ZNPUMULTIPROD-grantedCredits[n].currency longitud mayor a 3-NOK					
CP0078	ZNPUMULTIPROD-contact<SPECIFIC>.contactType longitud mayor a 30-NOK					
CP0079	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.contactDetailType longitud mayor a 12-NOK					
CP0080	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.number longitud mayor a 29-NOK					
CP0081	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.phoneCompany.id longitud mayor a 6-NOK					
CP0082	ZNPUMULTIPROD-rates.itemizeRates[n].rateType longitud mayor a 4-NOK					
CP0083	ZNPUMULTIPROD-rates.itemizeRates[n].itemizeRatesUnit<PERCENTAGE>.unitRateType longitud mayor a 10-NOK					
CP0084	ZNPUMULTIPROD-rates.itemizeRates[n].itemizeRatesUnit<PERCENTAGE>.percentage longitud mayor a 10-NOK					
CP0085	ZNPUMULTIPROD-fees.itemizeFees[n].feeType longitud mayor a 25-NOK					
CP0086	ZNPUMULTIPROD-fees.itemizeFees[n].itemizeFeeUnit<AMOUNT>.unitType longitud mayor a 6-NOK					
CP0087	ZNPUMULTIPROD-fees.itemizeFees[n].itemizeFeeUnit<AMOUNT>.amount longitud mayor a 18-NOK					
CP0088	ZNPUMULTIPROD-fees.itemizeFees[n].itemizeFeeUnit<AMOUNT>.currency longitud mayor a 3-NOK					
CP0089	ZNPUMULTIPROD-participants[n]<KNOW>.personType longitud mayor a 6-NOK					
CP0090	ZNPUMULTIPROD-participants[n]<KNOW>.id longitud mayor a 16-NOK					
CP0091	ZNPUMULTIPROD-participants[n]<KNOW>.isCustomer longitud mayor a 4-NOK					
CP0092	ZNPUMULTIPROD-participants[n]<KNOW>.participantType.id longitud mayor a 10-NOK					
CP0093	ZNPUMULTIPROD-offerId longitud mayor a 4-NOK					

Nota. Fuente: Elaboración propia.

Pruebas Funcionales

En base a la especificación del documento funcional, se generan los casos orientados al flujo del negocio. En la siguiente figura se visualiza el diseño de casos funcionales del servicio “Registrar solicitud de la tarjeta”.

Figura 33

Casos de prueba Funcionales

C204 - Casos de Prueba -		/createCardProposal	.XLSX	☆	📁	🔗
Archivo Editar Ver Insertar Formato Datos Herramientas Ayuda						
90% € % 0.00 123 Arial 10 B I S A						
fx						
A	B					
4	CodCaso	Caso de prueba				
5	CP0001	ZNPUMULTIPROD-Validación del servicio enviando los campos obligatorios para un No Cliente en el flujo-OK				
6	CP0002	ZNPUMULTIPROD-Validación del servicio enviando los campos obligatorios para un Cliente-OK				
7	CP0003	ZNPUMULTIPROD-Validación del servicio enviando un número de celular con formato incorrecto-NOK				
8	CP0004	ZNPUMULTIPROD-Validación del servicio enviando una tasa con identificador repetido-NOK				
9	CP0005	ZNPUMULTIPROD-Validación del servicio enviando una comisión con identificador repetido-NOK				
10	CP0006	ZNPUMULTIPROD-Validación del servicio no enviando el contactId en la cabecera-NOK				
11	CP0007	ZNPUMULTIPROD-Validación del servicio enviando un contactId vencido en la cabecera-NOK				
12	CP0008	ZNPUMULTIPROD-Validación del servicio enviando un tipo de detalle de contacto no permitido-NOK				
13	CP0009	ZNPUMULTIPROD-Validación del servicio enviando un tipo de contacto no permitido-NOK				
14	CP0010	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de 3 enteros)-NOK				
15	CP0011	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de 6 decimales)-NOK				
16	CP0012	ZNPUMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee más de 15 enteros)-NOK				
17	CP0013	ZNPUMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee más de 2 decimales)-NOK				
18	CP0014	ZNPUMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto (posee más de 15 enteros)-NOK				
19	CP0015	ZNPUMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto (posee más de 2 decimales)-NOK				

Nota. Fuente: Elaboración propia.

3.1.4. Definición del Nivel de prioridad de los casos de prueba

Una vez definidos los casos de prueba, se programa una reunión con el representante del negocio (PO) y Scrum Master para revisarlos y asignarle un nivel de prioridad (Alta, Media, Baja). Por lo general los casos de prueba funcionales son asignados con la mayor prioridad.

Figura 34

Diseño de casos de prueba priorizados

C204 - Casos de Prueba - /createCardProposal .XLSX ☆

Archivo Editar Ver Insertar Formato Datos Herramientas Ayuda Última modificaci

90% € % .0 .00 123 Arial 10 B I S A

fx Alta

A	B	C
CodCaso	Caso de prueba	Prioridad
CP0001	ZNPUMULTIPROD-Validación del servicio enviando los campos obligatorios para un No Cliente	Alta
CP0002	ZNPUMULTIPROD-Validación del servicio enviando los campos obligatorios para un Cliente-OK	Alta
CP0003	ZNPUMULTIPROD-Validación del servicio enviando un número de celular con formato	Alta
CP0004	ZNPUMULTIPROD-Validación del servicio enviando una tasa con identificador repetido-NOK	Alta
CP0005	ZNPUMULTIPROD-Validación del servicio enviando una comisión con identificador	Alta
CP0006	ZNPUMULTIPROD-Validación del servicio no enviando el contactId en la cabecera-NOK	Alta
CP0007	ZNPUMULTIPROD-Validación del servicio enviando un contactId vencido en la cabecera-NOK	Alta
CP0008	ZNPUMULTIPROD-Validación del servicio enviando un tipo de detalle de contacto no	Alta
CP0009	ZNPUMULTIPROD-Validación del servicio enviando un tipo de contacto no permitido-NOK	Alta
CP0010	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de	Alta
CP0011	ZNPUMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de	Alta
CP0012	ZNPUMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee	Alta
CP0013	ZNPUMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee	Alta
CP0014	ZNPUMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto	Alta
CP0015	ZNPUMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto	Alta
CP0016	ZNPUMULTIPROD-cardType objeto mandatorio-NOK	Media
CP0017	ZNPUMULTIPROD-cardType.id campo mandatorio-NOK	Media
CP0018	ZNPUMULTIPROD-product objeto mandatorio-NOK	Media
CP0019	ZNPUMULTIPROD-product.id campo mandatorio-NOK	Media
CP0020	ZNPUMULTIPROD-physicalSupport objeto mandatorio-NOK	Media
CP0021	ZNPUMULTIPROD-physicalSupport.id campo mandatorio-NOK	Media
CP0022	ZNPUMULTIPROD-grantedCredits[n] objeto opcional-OK	Media
CP0023	ZNPUMULTIPROD-grantedCredits[n].amount campo mandatorio-NOK	Media
CP0024	ZNPUMULTIPROD-grantedCredits[n].currency campo mandatorio-NOK	Media
CP0025	ZNPUMULTIPROD-contact<SPECIFIC> objeto opcional-OK	Media
CP0026	ZNPUMULTIPROD-contact<SPECIFIC>.contactType campo mandatorio-NOK	Media
CP0027	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE> objeto mandatorio-NOK	Media
CP0028	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.contactDetailType campo	Media
CP0029	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.number campo mandatorio-NOK	Media
CP0030	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.phoneCompany objeto opcional-OK	Media
CP0031	ZNPUMULTIPROD-contact<SPECIFIC>.contact<MOBILE>.phoneCompany.id campo	Media

Nota. En la figura se visualiza que cada caso de prueba tiene seleccionado un nivel de prioridad. Fuente Elaboración propia.

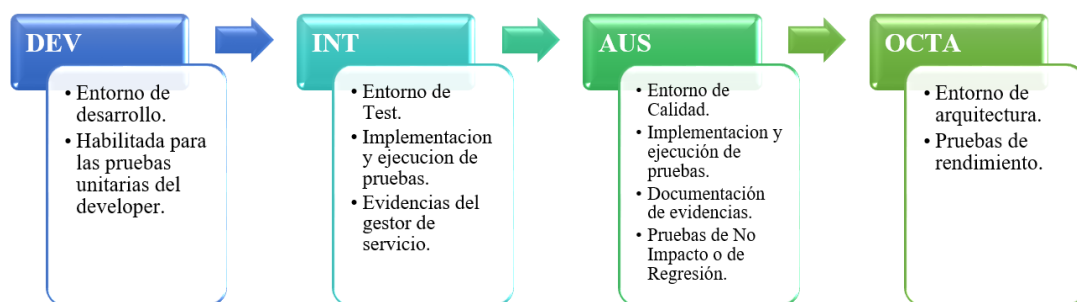
Bajo el mismo flujo de trabajo y en el mismo formato se genera el diseño de casos de prueba y se indica el nivel de prioridad para cada uno de los casos de prueba de las API.

3.1.5. Diseño del entorno de Prueba

Se identifica los ambientes de prueba que se necesita para las pruebas, se solicita los accesos y las URL de cada ambiente de prueba. En la siguiente figura vemos la estructura del entorno de prueba

Figura 35

Entornos de Prueba



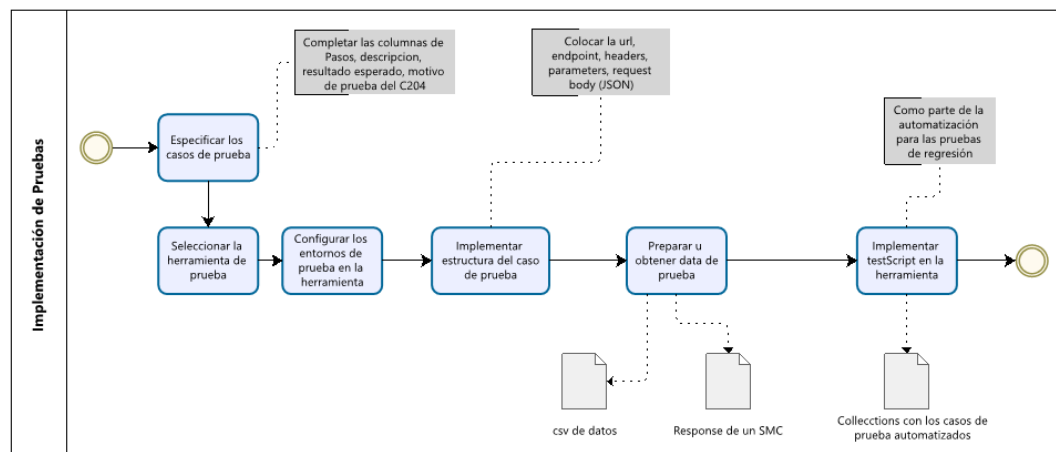
Nota. Actividades que se realizan en cada entorno de prueba. Fuente: Elaboración propia.

3.2. Implementación

En esta etapa del proceso de pruebas realizamos las siguientes actividades:

Figura 36

Flujo de tareas de la implementación de pruebas



Nota. Fuente: Elaboración propia.

3.2.1. Especificación de casos de Prueba

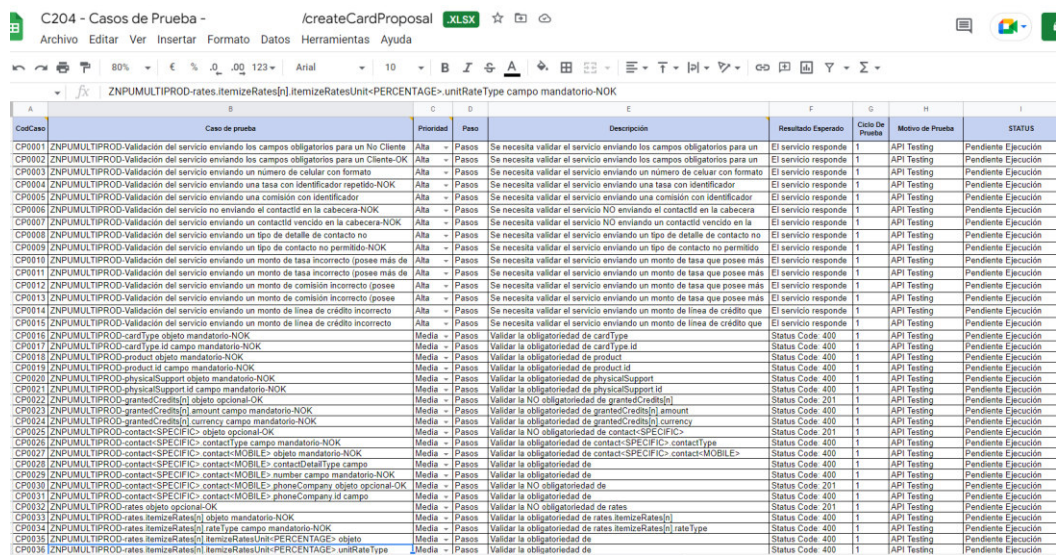
En esta actividad se indica una descripción más detallada de cada caso de prueba, el procedimiento para su ejecución, el resultado esperado y el motivo de la prueba.

De esta manera completamos la especificación de los casos de prueba para cada una de las API.

- **API Tarjetas:** Está API está relacionada con la gestión de las solicitudes de tarjetas realizadas por los clientes. Estas solicitudes se completan posteriormente, siguiendo un flujo de evaluación y finaliza con la contratación de la tarjeta.

Figura 37

Especificación de casos de prueba API tarjetas – registro de solicitud



A	B	C	D	E	F	G	H	I
Caso de prueba	Caso de prueba	Prioridad	Paso	Descripción	Resultado Esperado	Ciclo De Prueba	Motivo de Prueba	STATUS
CP0001	ZNPMULTIPROD-Validación del servicio enviando los campos obligatorios para un No Cliente	Alta	Pasos	Se necesita validar el servicio enviando los campos obligatorios para un	El servicio responde	1	API Testing	Pendiente Ejecución
CP0002	ZNPMULTIPROD-Validación del servicio enviando los campos obligatorios para un Cliente-OK	Alta	Pasos	Se necesita validar el servicio enviando los campos obligatorios para un	El servicio responde	1	API Testing	Pendiente Ejecución
CP0003	ZNPMULTIPROD-Validación del servicio enviando un número de celular con formato	Alta	Pasos	Se necesita validar el servicio enviando un número de celular con formato	El servicio responde	1	API Testing	Pendiente Ejecución
CP0004	ZNPMULTIPROD-Validación del servicio enviando una tasa con identificador repetido-NOK	Alta	Pasos	Se necesita validar el servicio enviando una tasa con identificador	El servicio responde	1	API Testing	Pendiente Ejecución
CP0005	ZNPMULTIPROD-Validación del servicio enviando una comisión con identificador	Alta	Pasos	Se necesita validar el servicio enviando una comisión con identificador	El servicio responde	1	API Testing	Pendiente Ejecución
CP0006	ZNPMULTIPROD-Validación del servicio no enviando el contactid en la cabecera-NOK	Alta	Pasos	Se necesita validar el servicio NO enviando el contactid en la cabecera	El servicio responde	1	API Testing	Pendiente Ejecución
CP0007	ZNPMULTIPROD-Validación del servicio enviando un contactid vencido en la cabecera-NOK	Alta	Pasos	Se necesita validar el servicio NO enviando un contactid vencido en la	El servicio responde	1	API Testing	Pendiente Ejecución
CP0008	ZNPMULTIPROD-Validación del servicio enviando un tipo de detalle de contacto no	Alta	Pasos	Se necesita validar el servicio enviando un tipo de detalle de contacto no	El servicio responde	1	API Testing	Pendiente Ejecución
CP0009	ZNPMULTIPROD-Validación del servicio enviando un tipo de contacto no permitido-NOK	Alta	Pasos	Se necesita validar el servicio enviando un tipo de contacto no permitido	El servicio responde	1	API Testing	Pendiente Ejecución
CP0010	ZNPMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de	Alta	Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0011	ZNPMULTIPROD-Validación del servicio enviando un monto de tasa incorrecto (posee más de	Alta	Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0012	ZNPMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee	Alta	Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0013	ZNPMULTIPROD-Validación del servicio enviando un monto de comisión incorrecto (posee	Alta	Pasos	Se necesita validar el servicio enviando un monto de tasa que posee más	El servicio responde	1	API Testing	Pendiente Ejecución
CP0014	ZNPMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto	Alta	Pasos	Se necesita validar el servicio enviando un monto de línea de crédito que	El servicio responde	1	API Testing	Pendiente Ejecución
CP0015	ZNPMULTIPROD-Validación del servicio enviando un monto de línea de crédito incorrecto	Alta	Pasos	Se necesita validar el servicio enviando un monto de línea de crédito que	El servicio responde	1	API Testing	Pendiente Ejecución
CP0016	ZNPMULTIPROD-cardType objeto mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de cardType	Status Code 400	1	API Testing	Pendiente Ejecución
CP0017	ZNPMULTIPROD-cardType id campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de cardType id	Status Code 400	1	API Testing	Pendiente Ejecución
CP0018	ZNPMULTIPROD-product objeto mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de product	Status Code 400	1	API Testing	Pendiente Ejecución
CP0019	ZNPMULTIPROD-product id campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de product id	Status Code 400	1	API Testing	Pendiente Ejecución
CP0020	ZNPMULTIPROD-physicalSupport objeto mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de physicalSupport	Status Code 400	1	API Testing	Pendiente Ejecución
CP0021	ZNPMULTIPROD-grantedCredits[n] campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de grantedCredits[n]	Status Code 400	1	API Testing	Pendiente Ejecución
CP0022	ZNPMULTIPROD-grantedCredits[n] amount campo mandatorio-NOK	Media	Pasos	Validar la NO obligatoriedad de grantedCredits[n]	Status Code 201	1	API Testing	Pendiente Ejecución
CP0023	ZNPMULTIPROD-grantedCredits[n] amount campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de grantedCredits[n] amount	Status Code 400	1	API Testing	Pendiente Ejecución
CP0024	ZNPMULTIPROD-grantedCredits[n] currency campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de grantedCredits[n] currency	Status Code 400	1	API Testing	Pendiente Ejecución
CP0025	ZNPMULTIPROD-contact-SPECIFIC> objeto optional-OK	Media	Pasos	Validar la NO obligatoriedad de contact-SPECIFIC>	Status Code 201	1	API Testing	Pendiente Ejecución
CP0026	ZNPMULTIPROD-contact-SPECIFIC> contactType campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de contact-SPECIFIC> contactType	Status Code 400	1	API Testing	Pendiente Ejecución
CP0027	ZNPMULTIPROD-contact-SPECIFIC> contact-MOBILE> objeto mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de contact-SPECIFIC> contact-MOBILE>	Status Code 400	1	API Testing	Pendiente Ejecución
CP0028	ZNPMULTIPROD-contact-SPECIFIC> contact-MOBILE> contactDetailType campo	Media	Pasos	Validar la obligatoriedad de	Status Code 400	1	API Testing	Pendiente Ejecución
CP0029	ZNPMULTIPROD-contact-SPECIFIC> contact-MOBILE> number campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de	Status Code 400	1	API Testing	Pendiente Ejecución
CP0030	ZNPMULTIPROD-contact-SPECIFIC> contact-MOBILE> phoneCompany objeto optional-OK	Media	Pasos	Validar la NO obligatoriedad de	Status Code 201	1	API Testing	Pendiente Ejecución
CP0031	ZNPMULTIPROD-contact-SPECIFIC> contact-MOBILE> phoneCompany id campo	Media	Pasos	Validar la obligatoriedad de	Status Code 400	1	API Testing	Pendiente Ejecución
CP0032	ZNPMULTIPROD-rates objeto optional-OK	Media	Pasos	Validar la NO obligatoriedad de rates	Status Code 201	1	API Testing	Pendiente Ejecución
CP0033	ZNPMULTIPROD-rates itemizeRates[n] objeto mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de rates itemizeRates[n]	Status Code 400	1	API Testing	Pendiente Ejecución
CP0034	ZNPMULTIPROD-rates itemizeRates[n] rateType campo mandatorio-NOK	Media	Pasos	Validar la obligatoriedad de rates itemizeRates[n] rateType	Status Code 400	1	API Testing	Pendiente Ejecución
CP0035	ZNPMULTIPROD-rates itemizeRates[n] itemizeRatesUnit=PERCENTAGE> objeto	Media	Pasos	Validar la obligatoriedad de	Status Code 400	1	API Testing	Pendiente Ejecución
CP0036	ZNPMULTIPROD-rates itemizeRates[n] itemizeRatesUnit=PERCENTAGE> unitRateType	Media	Pasos	Validar la obligatoriedad de	Status Code 400	1	API Testing	Pendiente Ejecución

Nota. En la figura se visualiza el detalle de los casos de prueba del servicio multicanal registro de solicitud de tarjeta. Fuente: Elaboración propia.

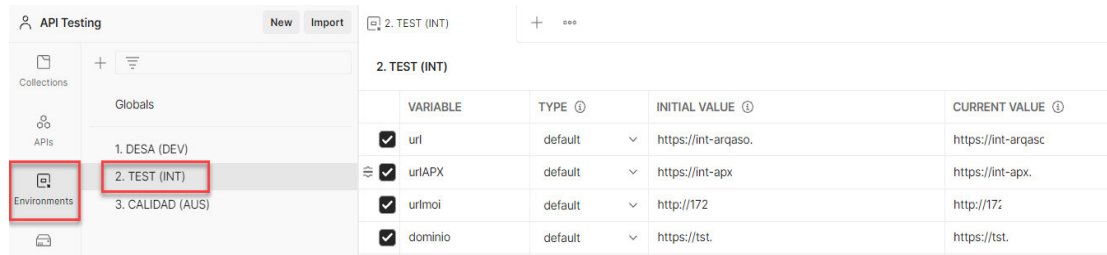
3.2.2. Configuración de entornos de prueba en la herramienta

En la sección entornos (environments) de la herramienta Postman se coloca y parametriza cada una de las URL que se requiere para ejecutar las pruebas. Esto permite gestionar con mayor agilidad los entornos de prueba.

Se prepara las URL de configuración del ambiente de Test y se agrega y configura en la herramienta.

Figura 38

Configuración del ambiente Test



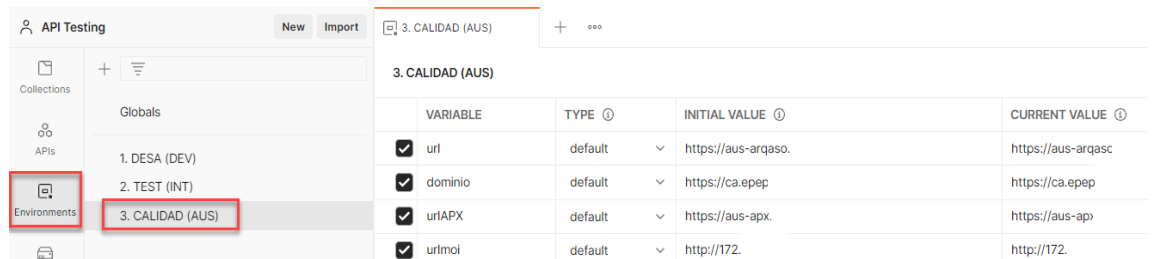
VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE
☒ url	default	https://int-arqaso.	https://int-arqasc
☒ urlAPX	default	https://int-apx	https://int-apx.
☒ urlmoi	default	http://172	http://172
☒ dominio	default	https://tst.	https://tst.

Nota. En la figura se muestra las URL del ambiente de Test. Fuente: Adaptada de Postman.

Se prepara las URL de configuración del ambiente de Test y se agrega y configura en la herramienta.

Figura 39

Configuración del ambiente Calidad



VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE
☒ url	default	https://aus-arqaso.	https://aus-arqasc
☒ dominio	default	https://ca.epep	https://ca.epep
☒ urlAPX	default	https://aus-apx.	https://aus-apx
☒ urlmoi	default	http://172.	http://172.

Nota. En la figura se visualiza las URL del ambiente de Test. Fuente: Adaptada de Postman.

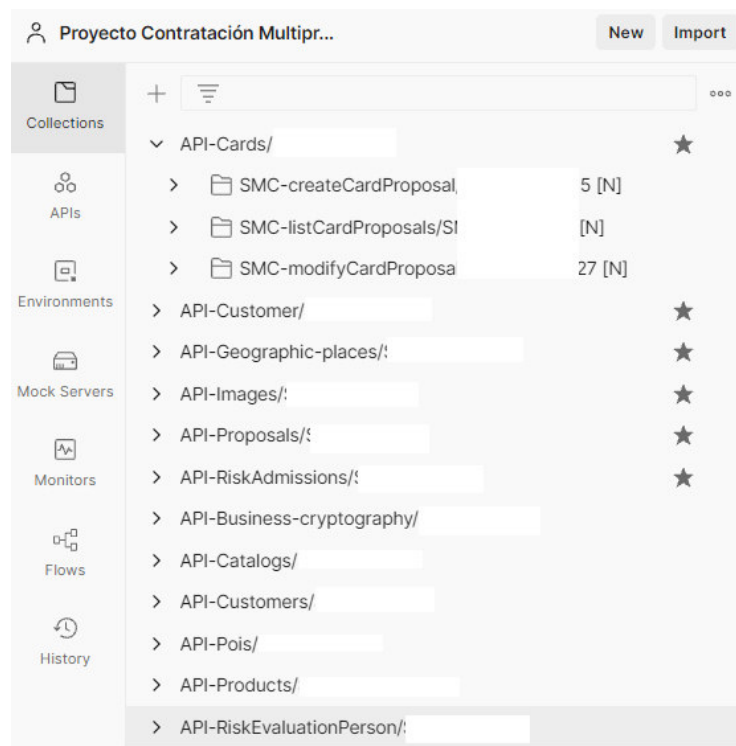
3.2.3. Implementación del caso de prueba

En base al documento SMC que corresponde a la ficha técnica del servicio multicanal y la plataforma del Catálogo API se arma la estructura de cada uno de los servicios y se va desglosando dependiendo de cada caso de prueba definido en el C204.

Se inicia con la creación de la colección en Postman indicando el nombre de la API, tal como se muestra en la siguiente imagen.

Figura 40

Estructura de las colecciones para las pruebas de API

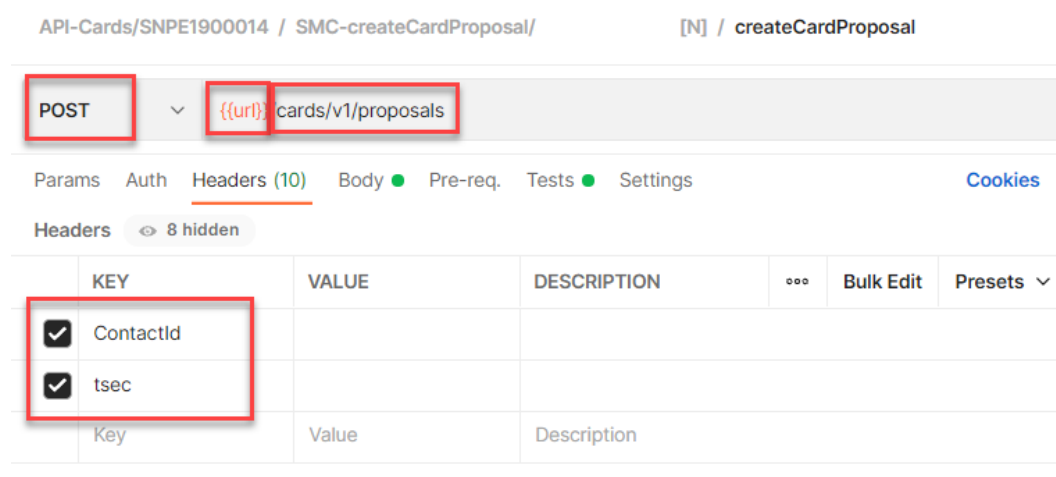


Nota. Fuente: Adaptada de Postman.

Del documento técnico, se obtiene el método, la URL, end-point y cabecera del servicio, para colocar esa información en la herramienta creamos una petición (Request).

Figura 41

Configuración de datos en Postman

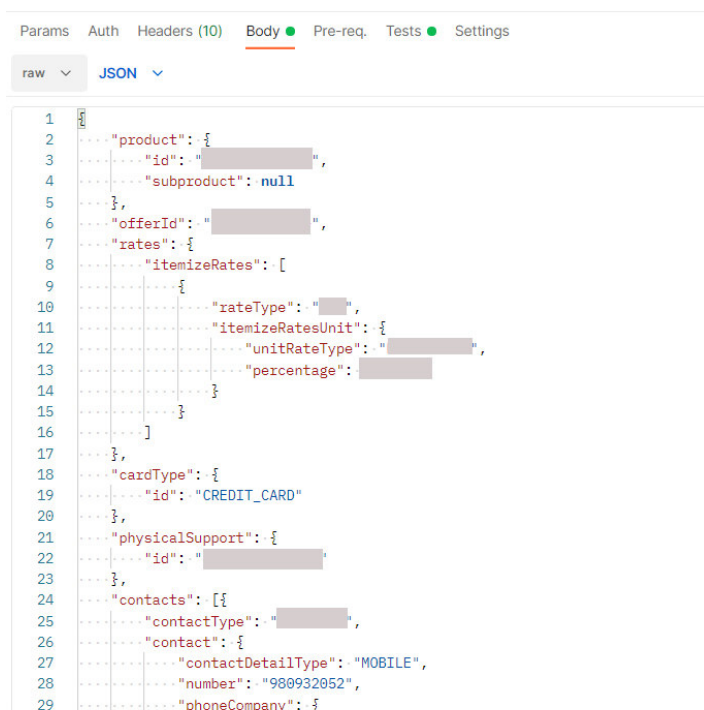


Nota. Fuente: Adaptada de Postman.

De la misma manera y con la guía al documento técnico se arma la estructura de la petición (request body) en formato json y se coloca en la sección Body de la herramienta.

Figura 42

Datos de entrada en Postman



Nota. Fuente: Adaptada de Postman.

Teniendo estos datos completos se genera los request para cada caso de prueba. Los request de cada prueba están en una subcarpeta llamada “Test Cases” y son renombrados bajo el siguiente formato:

Código del caso de prueba + | + Nombre del caso de prueba

3.2.4. Revisión de prerrequisitos y preparación de datos de prueba

Dependiendo del caso de prueba y del pre requisito de la prueba, se invoca a otros servicios.

Un pre requisito mandatorio para todos los servicios es obtener el valor del token de autenticación y enviar como cabecera en todos los servicios multicanales.

En la siguiente imagen se visualiza los dos pre-requisitos del servicio “registrar la solicitud de la tarjeta”. El primero proporciona el token de autenticación y el segundo el valor encriptado del tipo y numero de documento.

Figura 43

Pre requisitos del servicio Registrar la solicitud de la tarjeta.



Nota. Fuente: Adaptada de Postman.

3.2.5. Implementación Test Scripts de validación

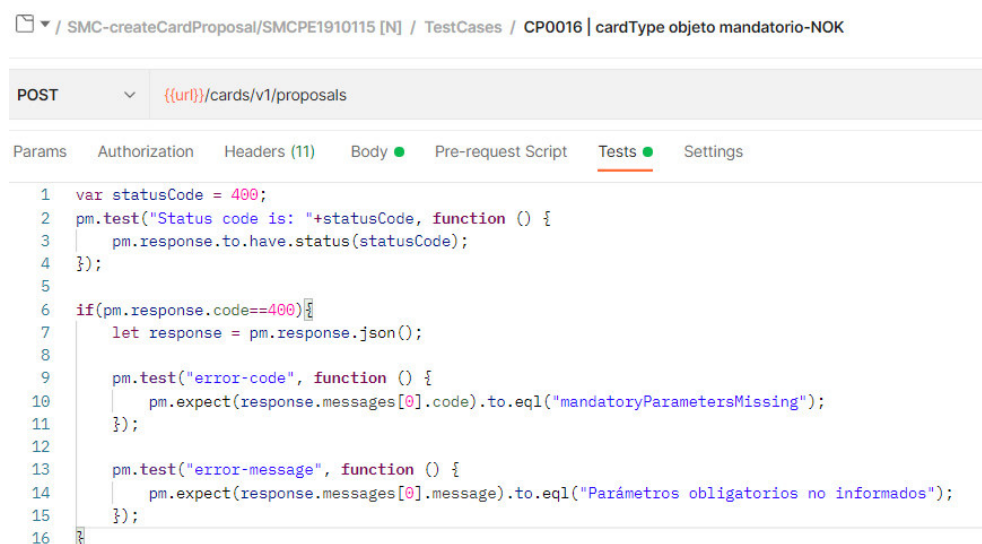
Pruebas de Integridad de Campos

Para cada caso de prueba especificado, se codifica los test scripts de validación dependiendo del tipo de caso de prueba. El desarrollo de cada test script es parte de la automatización para las pruebas de regresión.

El desarrollo del test script de validación para los casos de obligatoriedad consiste en validar primero que el status code corresponda a un error 400 y si cumple la condición continuara con las siguientes sentencias de validación.

Figura 44

Implementación test script de validación - Caso de prueba de Obligatoriedad



Nota. Fuente: Elaboración propia.

En la siguiente figura se muestra un ejemplo de test script para el caso de prueba de valor inválido.

Figura 45

Implementación test script de validación - Caso de prueba de Valor inválido



```
1 var statusCode = 400;
2 pm.test("Status code is " + statusCode, function () {
3   pm.response.to.have.status(statusCode);
4 });
5
6 if(pm.response.code == statusCode){
7
8   var resultError = pm.response.json().messages[0];
9   var expectedErrorCode = "00068011";
10  var expectedErrorMessage = expectedErrorCode + "#" + "CELULAR INCORRECTO";
11
12  pm.test("Error code is " + expectedErrorCode, function () {
13    pm.expect(resultError.code).to.eql(expectedErrorCode);
14  });
15  pm.test("Message is " + expectedErrorMessage, function () {
16    pm.expect(resultError.message).to.eql(expectedErrorMessage);
17  });
18 }
```

Nota. Fuente: Elaboración propia.

En la siguiente figura se muestra un ejemplo de test script para el caso de prueba de longitud de campo.

Figura 46

Implementación test script - Caso de prueba de Longitud de campo



```
1 var statusCode = 400;
2 pm.test("Status code is: "+statusCode, function () {
3   pm.response.to.have.status(statusCode);
4 });
5
6 if(pm.response.code==400){
7   let response = pm.response.json();
8
9   pm.test("error-code", function () {
10     pm.expect(response.messages[0].code).to.eql("badRequest");
11   });
12
13   pm.test("error-message", function () {
14     pm.expect(response.messages[0].message).to.eql("badRequest");
15   });
16 }
```

Nota. Fuente: Elaboración propia.

Pruebas Funcionales

Para la implementación y una ejecución correcta de estas pruebas se requiere preparar data de prueba adecuada, para ello se requiere la invocación de otros servicios que nos proporcionen estos datos. Dependiendo de cada caso de prueba se requerirá los datos o no de uno o más servicios.

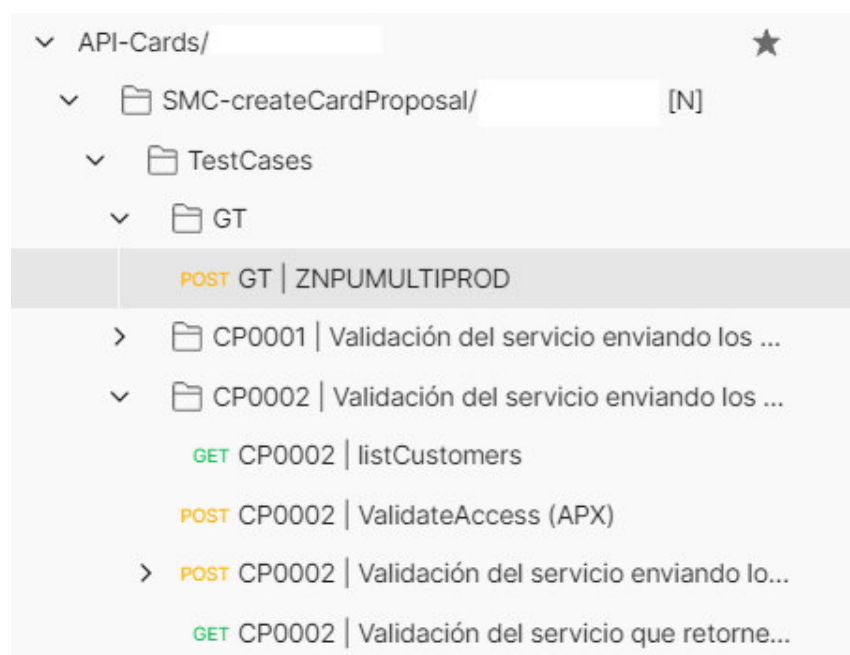
Es el caso de los servicios multicanales que corresponden al API Tarjetas, en la siguiente figura se puede visualizar un caso de prueba de ejemplo que corresponde al servicio multicanal “Registrar solicitud de tarjeta”.

CP0002: Se requiere validar el alta de una tarjeta de crédito para un Cliente.

Para ello se necesita del servicio “listar datos del cliente”, que devuelve información del cliente, lo que a su vez permite que se envíe como dato de entrada el tipo de documento, numero de documento, teléfono y código de cliente en la transacción backend createValidateAccess, y del que se obtendrá el valor del Contact Id (Tipo y Numero de documento encriptado) que se enviara en el servicio “Registrar solicitud del cliente” como una cabecera obligatoria.

Figura 47

Estructura de un caso de prueba

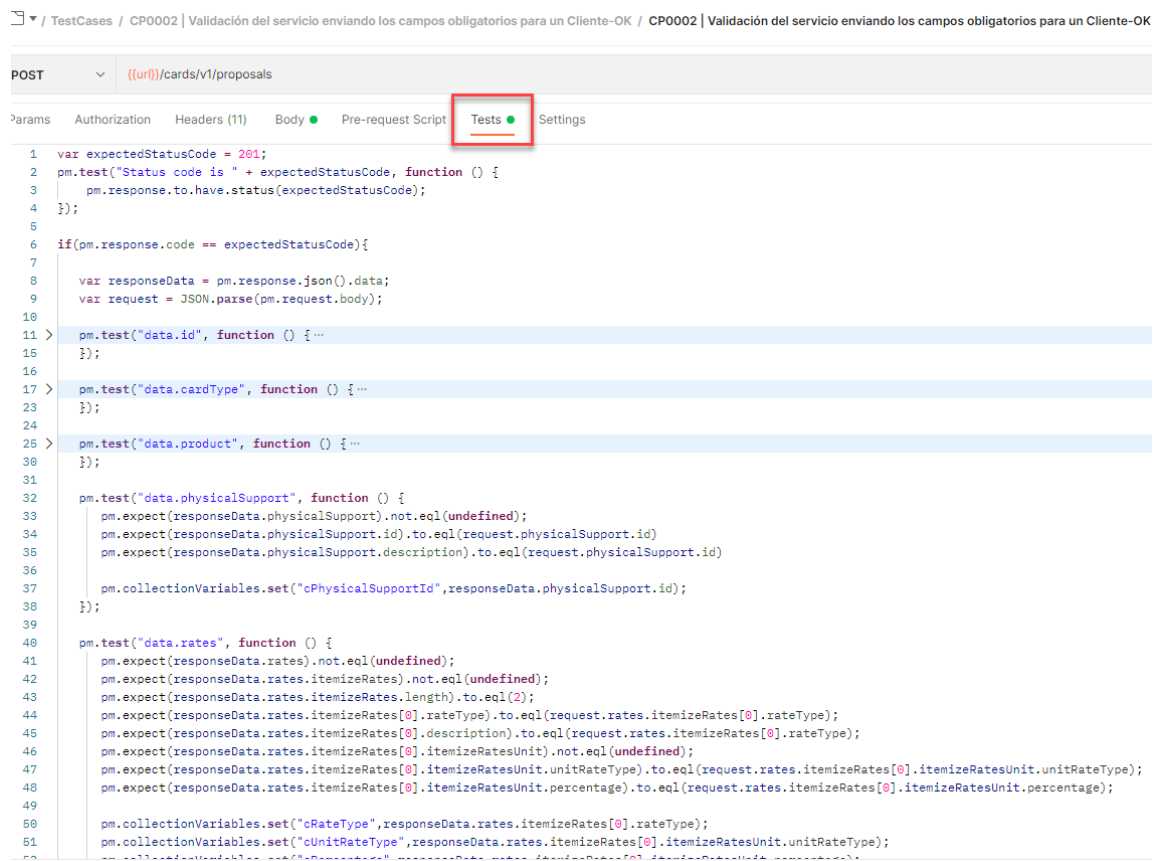


Nota. Fuente: Elaboración Propia.

El desarrollo del test script de validación para los casos funcionales consiste en validar primero que el status code corresponda a un error 201 created, si cumple la condición continuara con las siguientes sentencias de validación. En la siguiente figura se visualiza un ejemplo de test script para un caso de prueba funcional del servicio que genera el registro de una solicitud de tarjeta.

Figura 48

Test script - Caso de prueba funcional “Registrar solicitud de tarjeta”



```
POST {{url}}/cards/v1/proposals

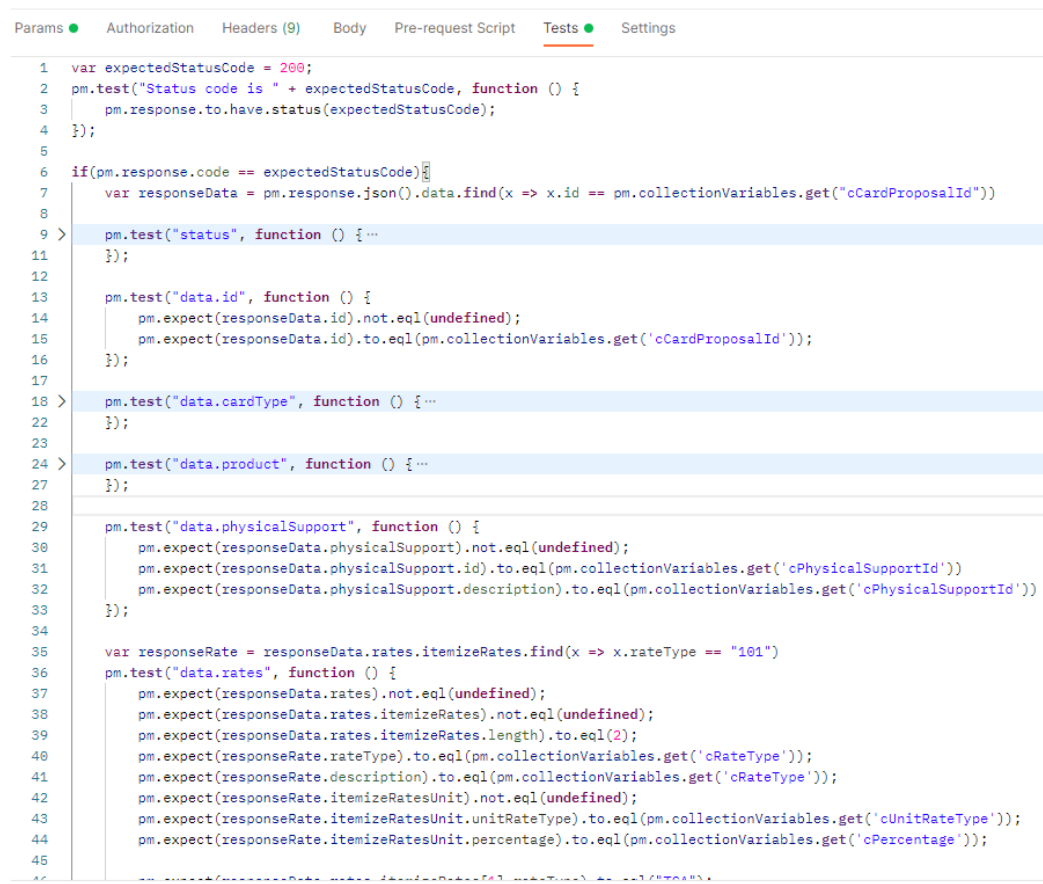
1  var expectedStatusCode = 201;
2  pm.test("Status code is " + expectedStatusCode, function () {
3    pm.response.to.have.status(expectedStatusCode);
4  });
5
6  if(pm.response.code == expectedStatusCode){
7
8    var responseData = pm.response.json().data;
9    var request = JSON.parse(pm.request.body);
10
11 > pm.test("data.id", function () { ...
15   });
16
17 > pm.test("data.cardType", function () { ...
23   });
24
25 > pm.test("data.product", function () { ...
30   });
31
32   pm.test("data.physicalSupport", function () {
33     pm.expect(responseData.physicalSupport).not.eql(undefined);
34     pm.expect(responseData.physicalSupport.id).to.eql(request.physicalSupport.id)
35     pm.expect(responseData.physicalSupport.description).to.eql(request.physicalSupport.id)
36
37     pm.collectionVariables.set("cPhysicalSupportId",responseData.physicalSupport.id);
38   });
39
40   pm.test("data.rates", function () {
41     pm.expect(responseData.rates).not.eql(undefined);
42     pm.expect(responseData.rates.itemizeRates).not.eql(undefined);
43     pm.expect(responseData.rates.itemizeRates.length).to.eql(2);
44     pm.expect(responseData.rates.itemizeRates[0].rateType).to.eql(request.rates.itemizeRates[0].rateType);
45     pm.expect(responseData.rates.itemizeRates[0].description).to.eql(request.rates.itemizeRates[0].rateType);
46     pm.expect(responseData.rates.itemizeRates[0].itemizeRatesUnit).not.eql(undefined);
47     pm.expect(responseData.rates.itemizeRates[0].itemizeRatesUnit.unitRateType).to.eql(request.rates.itemizeRates[0].itemizeRatesUnit.unitRateType);
48     pm.expect(responseData.rates.itemizeRates[0].itemizeRatesUnit.percentage).to.eql(request.rates.itemizeRates[0].itemizeRatesUnit.percentage);
49
50     pm.collectionVariables.set("cRateType",responseData.rates.itemizeRates[0].rateType);
51     pm.collectionVariables.set("cUnitRateType",responseData.rates.itemizeRates[0].itemizeRatesUnit.unitRateType);
52   });
53 }
```

Nota. Fuente: Elaboración propia.

En esta otra figura se visualiza el test script del mismo caso de prueba funcional pero con el servicio que lista la solicitud de la tarjeta. Lo que hace es validar que los datos que se envió al generar la solicitud se hayan guardado correctamente. Por tanto primero se valida que el status code corresponda a un 200 OK, si cumple la condición continuara con las siguientes sentencias de validación.

Figura 49

Test script - Caso de prueba Funcional “Listar solicitud de la tarjeta”



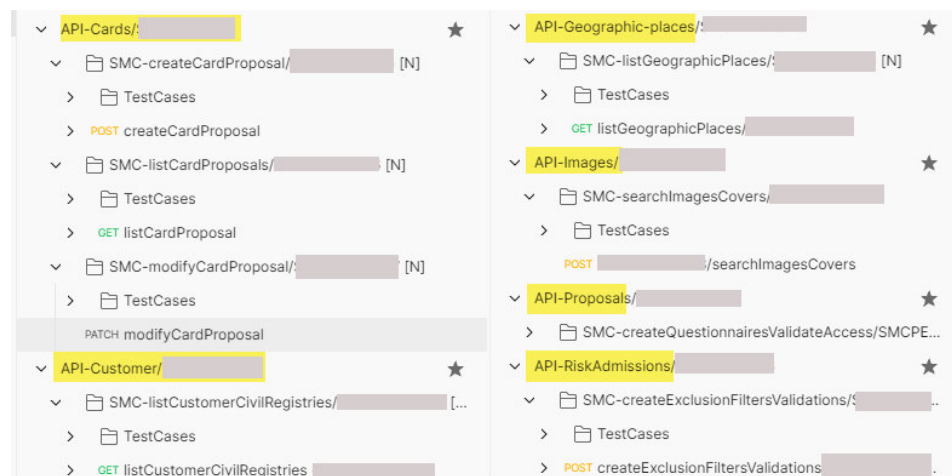
```
Params ● Authorization Headers (9) Body Pre-request Script Tests ● Settings

1 var expectedStatusCode = 200;
2 pm.test("Status code is " + expectedStatusCode, function () {
3   pm.response.to.have.status(expectedStatusCode);
4 });
5
6 if(pm.response.code == expectedStatusCode){
7   var responseData = pm.response.json().data.find(x => x.id == pm.collectionVariables.get("cCardProposalId"))
8
9 > pm.test("status", function () { ...
10  });
11
12  pm.test("data.id", function () {
13    pm.expect(responseData.id).not.eql(undefined);
14    pm.expect(responseData.id).to.eql(pm.collectionVariables.get('cCardProposalId'));
15  });
16
17  pm.test("data.cardType", function () { ...
18  });
19
20  pm.test("data.product", function () { ...
21  });
22
23  pm.test("data.physicalSupport", function () {
24    pm.expect(responseData.physicalSupport).not.eql(undefined);
25    pm.expect(responseData.physicalSupport.id).to.eql(pm.collectionVariables.get('cPhysicalSupportId'));
26    pm.expect(responseData.physicalSupport.description).to.eql(pm.collectionVariables.get('cPhysicalSupportId'));
27  });
28
29  var responseRate = responseData.rates.itemizeRates.find(x => x.rateType == "101")
30  pm.test("data.rates", function () {
31    pm.expect(responseData.rates).not.eql(undefined);
32    pm.expect(responseData.rates.itemizeRates).not.eql(undefined);
33    pm.expect(responseData.rates.itemizeRates.length).to.eql(2);
34    pm.expect(responseRate.rateType).to.eql(pm.collectionVariables.get('cRateType'));
35    pm.expect(responseRate.description).to.eql(pm.collectionVariables.get('cRateType'));
36    pm.expect(responseRate.itemizeRatesUnit).not.eql(undefined);
37    pm.expect(responseRate.itemizeRatesUnit.unitRateType).to.eql(pm.collectionVariables.get('cUnitRateType'));
38    pm.expect(responseRate.itemizeRatesUnit.percentage).to.eql(pm.collectionVariables.get('cPercentage'));
39  });
40
41  pm.expect(responseData.rates.itemizeRates[0].rateType).to.eql("101");
42
43  pm.expect(responseData.rates.itemizeRates[1].rateType).to.eql("102");
44
45  pm.expect(responseData.rates.itemizeRates[2].rateType).to.eql("103");
46
47  pm.expect(responseData.rates.itemizeRates[3].rateType).to.eql("104");
48
49  pm.expect(responseData.rates.itemizeRates[4].rateType).to.eql("105");
50
51  pm.expect(responseData.rates.itemizeRates[5].rateType).to.eql("106");
52
53  pm.expect(responseData.rates.itemizeRates[6].rateType).to.eql("107");
54
55  pm.expect(responseData.rates.itemizeRates[7].rateType).to.eql("108");
56
57  pm.expect(responseData.rates.itemizeRates[8].rateType).to.eql("109");
58
59  pm.expect(responseData.rates.itemizeRates[9].rateType).to.eql("110");
60
61  pm.expect(responseData.rates.itemizeRates[10].rateType).to.eql("111");
62
63  pm.expect(responseData.rates.itemizeRates[11].rateType).to.eql("112");
64
65  pm.expect(responseData.rates.itemizeRates[12].rateType).to.eql("113");
66
67  pm.expect(responseData.rates.itemizeRates[13].rateType).to.eql("114");
68
69  pm.expect(responseData.rates.itemizeRates[14].rateType).to.eql("115");
70
71  pm.expect(responseData.rates.itemizeRates[15].rateType).to.eql("116");
72
73  pm.expect(responseData.rates.itemizeRates[16].rateType).to.eql("117");
74
75  pm.expect(responseData.rates.itemizeRates[17].rateType).to.eql("118");
76
77  pm.expect(responseData.rates.itemizeRates[18].rateType).to.eql("119");
78
79  pm.expect(responseData.rates.itemizeRates[19].rateType).to.eql("120");
80
81  pm.expect(responseData.rates.itemizeRates[20].rateType).to.eql("121");
82
83  pm.expect(responseData.rates.itemizeRates[21].rateType).to.eql("122");
84
85  pm.expect(responseData.rates.itemizeRates[22].rateType).to.eql("123");
86
87  pm.expect(responseData.rates.itemizeRates[23].rateType).to.eql("124");
88
89  pm.expect(responseData.rates.itemizeRates[24].rateType).to.eql("125");
90
91  pm.expect(responseData.rates.itemizeRates[25].rateType).to.eql("126");
92
93  pm.expect(responseData.rates.itemizeRates[26].rateType).to.eql("127");
94
95  pm.expect(responseData.rates.itemizeRates[27].rateType).to.eql("128");
96
97  pm.expect(responseData.rates.itemizeRates[28].rateType).to.eql("129");
98
99  pm.expect(responseData.rates.itemizeRates[29].rateType).to.eql("130");
100
101  pm.expect(responseData.rates.itemizeRates[30].rateType).to.eql("131");
102
103  pm.expect(responseData.rates.itemizeRates[31].rateType).to.eql("132");
104
105  pm.expect(responseData.rates.itemizeRates[32].rateType).to.eql("133");
106
107  pm.expect(responseData.rates.itemizeRates[33].rateType).to.eql("134");
108
109  pm.expect(responseData.rates.itemizeRates[34].rateType).to.eql("135");
110
111  pm.expect(responseData.rates.itemizeRates[35].rateType).to.eql("136");
112
113  pm.expect(responseData.rates.itemizeRates[36].rateType).to.eql("137");
114
115  pm.expect(responseData.rates.itemizeRates[37].rateType).to.eql("138");
116
117  pm.expect(responseData.rates.itemizeRates[38].rateType).to.eql("139");
118
119  pm.expect(responseData.rates.itemizeRates[39].rateType).to.eql("140");
120
121  pm.expect(responseData.rates.itemizeRates[40].rateType).to.eql("141");
122
123  pm.expect(responseData.rates.itemizeRates[41].rateType).to.eql("142");
124
125  pm.expect(responseData.rates.itemizeRates[42].rateType).to.eql("143");
126
127  pm.expect(responseData.rates.itemizeRates[43].rateType).to.eql("144");
128
129  pm.expect(responseData.rates.itemizeRates[44].rateType).to.eql("145");
130
131  pm.expect(responseData.rates.itemizeRates[45].rateType).to.eql("146");
132
133  pm.expect(responseData.rates.itemizeRates[46].rateType).to.eql("147");
134
135  pm.expect(responseData.rates.itemizeRates[47].rateType).to.eql("148");
136
137  pm.expect(responseData.rates.itemizeRates[48].rateType).to.eql("149");
138
139  pm.expect(responseData.rates.itemizeRates[49].rateType).to.eql("150");
140
141  pm.expect(responseData.rates.itemizeRates[50].rateType).to.eql("151");
142
143  pm.expect(responseData.rates.itemizeRates[51].rateType).to.eql("152");
144
145  pm.expect(responseData.rates.itemizeRates[52].rateType).to.eql("153");
146
147  pm.expect(responseData.rates.itemizeRates[53].rateType).to.eql("154");
148
149  pm.expect(responseData.rates.itemizeRates[54].rateType).to.eql("155");
150
151  pm.expect(responseData.rates.itemizeRates[55].rateType).to.eql("156");
152
153  pm.expect(responseData.rates.itemizeRates[56].rateType).to.eql("157");
154
155  pm.expect(responseData.rates.itemizeRates[57].rateType).to.eql("158");
156
157  pm.expect(responseData.rates.itemizeRates[58].rateType).to.eql("159");
158
159  pm.expect(responseData.rates.itemizeRates[59].rateType).to.eql("160");
160
161  pm.expect(responseData.rates.itemizeRates[60].rateType).to.eql("161");
162
163  pm.expect(responseData.rates.itemizeRates[61].rateType).to.eql("162");
164
165  pm.expect(responseData.rates.itemizeRates[62].rateType).to.eql("163");
166
167  pm.expect(responseData.rates.itemizeRates[63].rateType).to.eql("164");
168
169  pm.expect(responseData.rates.itemizeRates[64].rateType).to.eql("165");
170
171  pm.expect(responseData.rates.itemizeRates[65].rateType).to.eql("166");
172
173  pm.expect(responseData.rates.itemizeRates[66].rateType).to.eql("167");
174
175  pm.expect(responseData.rates.itemizeRates[67].rateType).to.eql("168");
176
177  pm.expect(responseData.rates.itemizeRates[68].rateType).to.eql("169");
178
179  pm.expect(responseData.rates.itemizeRates[69].rateType).to.eql("170");
180
181  pm.expect(responseData.rates.itemizeRates[70].rateType).to.eql("171");
182
183  pm.expect(responseData.rates.itemizeRates[71].rateType).to.eql("172");
184
185  pm.expect(responseData.rates.itemizeRates[72].rateType).to.eql("173");
186
187  pm.expect(responseData.rates.itemizeRates[73].rateType).to.eql("174");
188
189  pm.expect(responseData.rates.itemizeRates[74].rateType).to.eql("175");
190
191  pm.expect(responseData.rates.itemizeRates[75].rateType).to.eql("176");
192
193  pm.expect(responseData.rates.itemizeRates[76].rateType).to.eql("177");
194
195  pm.expect(responseData.rates.itemizeRates[77].rateType).to.eql("178");
196
197  pm.expect(responseData.rates.itemizeRates[78].rateType).to.eql("179");
198
199  pm.expect(responseData.rates.itemizeRates[79].rateType).to.eql("180");
200
201  pm.expect(responseData.rates.itemizeRates[80].rateType).to.eql("181");
202
203  pm.expect(responseData.rates.itemizeRates[81].rateType).to.eql("182");
204
205  pm.expect(responseData.rates.itemizeRates[82].rateType).to.eql("183");
206
207  pm.expect(responseData.rates.itemizeRates[83].rateType).to.eql("184");
208
209  pm.expect(responseData.rates.itemizeRates[84].rateType).to.eql("185");
210
211  pm.expect(responseData.rates.itemizeRates[85].rateType).to.eql("186");
212
213  pm.expect(responseData.rates.itemizeRates[86].rateType).to.eql("187");
214
215  pm.expect(responseData.rates.itemizeRates[87].rateType).to.eql("188");
216
217  pm.expect(responseData.rates.itemizeRates[88].rateType).to.eql("189");
218
219  pm.expect(responseData.rates.itemizeRates[89].rateType).to.eql("190");
219
```

De esta manera se realiza la implementación de los test scripts de cada una de las API.

Figura 50

Colecciones de API implementadas



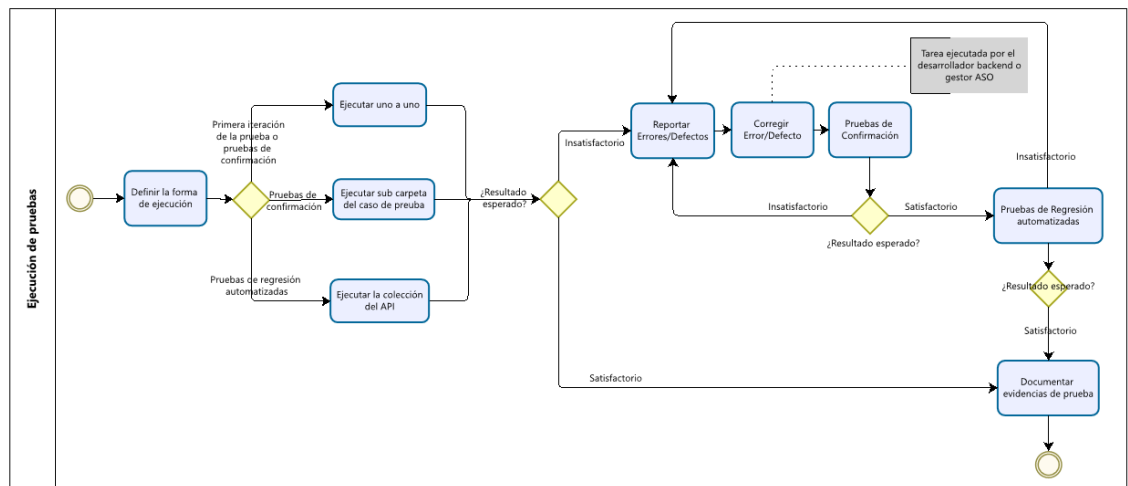
Nota. Fuente: Adaptada de Postman.

3.3. Ejecución de pruebas

En esta etapa se realiza la ejecución de los casos de prueba de cada una de las API, el reporte de incidencias, las pruebas de confirmación, las pruebas de regresión y la documentación de evidencias en el formato definido por la entidad financiera.

Figura 51

Flujo de tareas de la ejecución de pruebas



Nota. Fuente: Elaboración propia.

3.3.1. Ejecución de pruebas

Una vez implementado cada caso de prueba se inicia con la ejecución de estas. La ejecución se puede realizar de uno a uno, por subcarpetas o toda la colección.

3.3.1.1. Ejecución de uno a uno

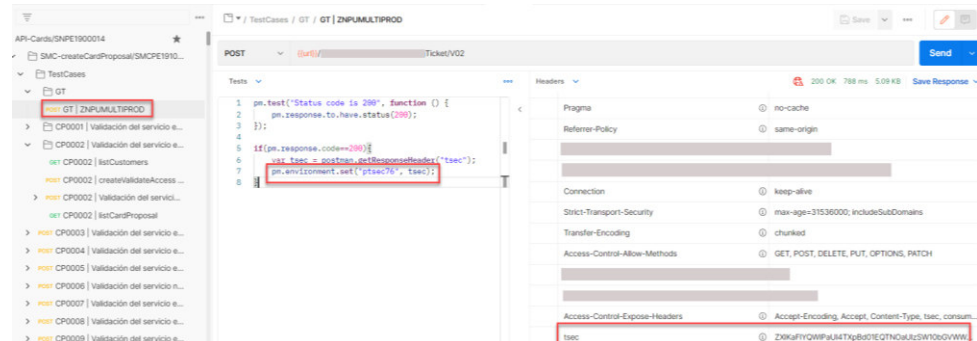
Se toma como ejemplo el caso de prueba: CP0002 | Validación del servicio enviando los campos obligatorios para un Cliente-OK. Este caso de prueba tiene como objetivo validar el alta de una solicitud de tarjeta con los datos básicos para un cliente. Para ejecutar este caso de prueba se seguirá los siguientes pasos:

- 1) Se selecciona el entorno de prueba donde se va ejecutar el caso de prueba.
- 2) Se ejecuta el servicio de autenticación, en la sección test se hizo una implementación para guardar el valor del token. Este valor se envía como cabecera en todos los servicios multicanales y tiene un tiempo de

expiración de 10 minutos a 60 minutos dependiendo del canal al que corresponde.

Figura 52

Ejecución del servicio de autenticación

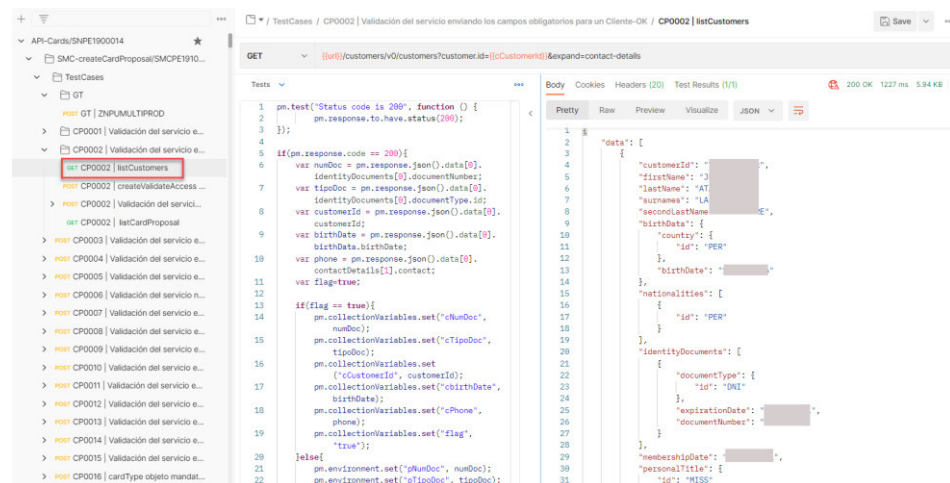


Nota. Fuente: Adaptada de Postman.

- 3) Se ejecuta el servicio que lista los datos de un cliente, en la sección test se realiza una implementación para que permita guardar los siguientes datos: tipo, documento, código, fecha de nacimiento y teléfono.

Figura 53

Ejecución del servicio “listar datos del cliente”

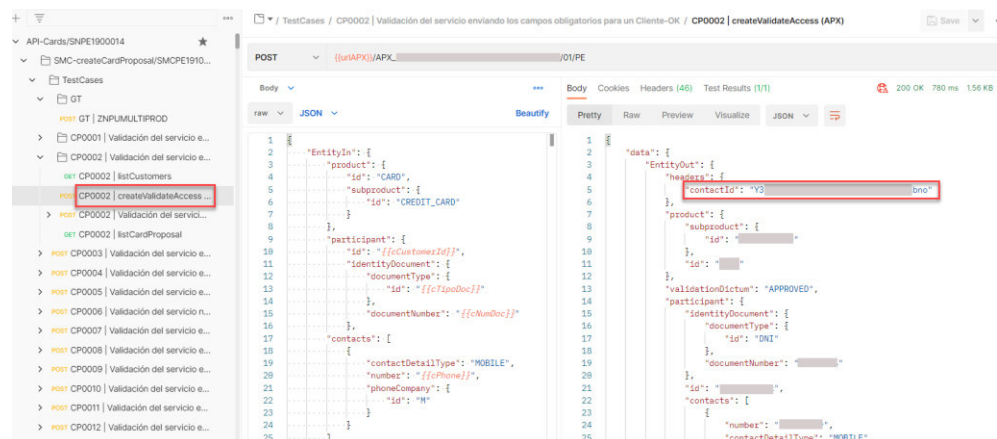


Nota. Fuente: Adaptada de Postman.

- 4) Se ejecuta la transacción backend del servicio que realiza el primer filtro al cliente y genera un valor encryptado del tipo y numero de documento del cliente. Este valor sirve como traza para la ejecución y flujo con las demás API. Aquí se manda los datos obtenidos en la ejecución anterior y de la misma manera en la sección test se hizo una implementación para guardar el valor encryptado que da como respuesta.

Figura 54

Ejecución de la transacción backend

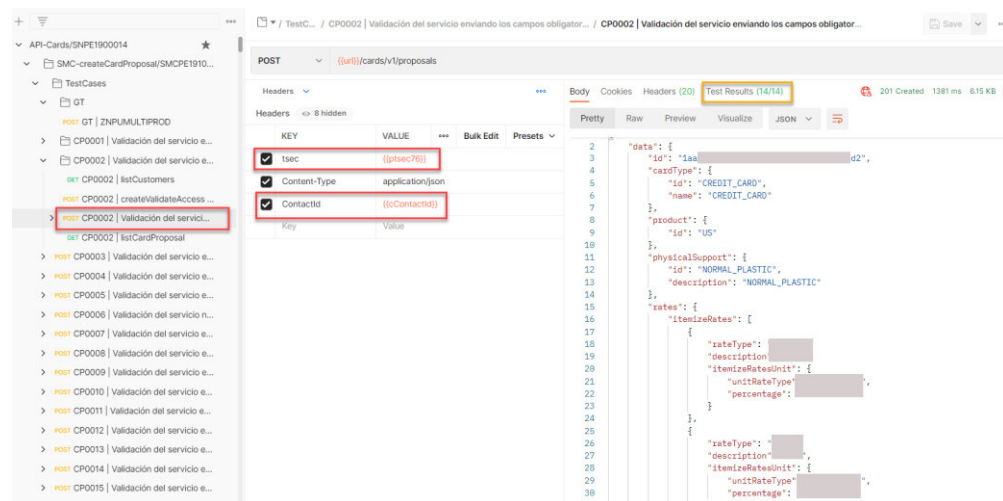


Nota. En la figura se visualiza el JSON de entrada y salida. Fuente: Adaptada de Postman.

5) Se ejecuta el servicio propósito del caso de prueba. En la cabecera se envía el valor del token de autenticación y el valor encriptado generado en las ejecuciones anteriores. En el request body se envía los datos preparados en formato JSON. En la sección test además de implementar los test script de validación se hizo una implementación para guardar los datos que se envían para generar la solicitud.

Figura 55

Ejecución del servicio “Registrar la solicitud de la tarjeta”

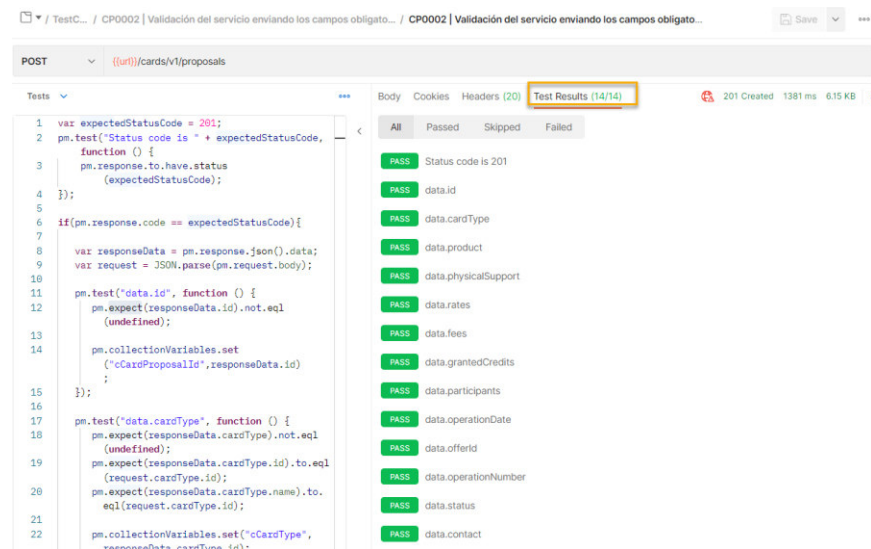


Nota. Fuente: Adaptada de Postman.

- 6) En la sección Test Results se valida el resultado de los test scripts implementados. Si todos los scripts de validación se dieron de forma correcta se muestra PASS, en caso haya error se muestra FAIL.

Figura 56

Resultado de los test scripts de validación del caso de prueba

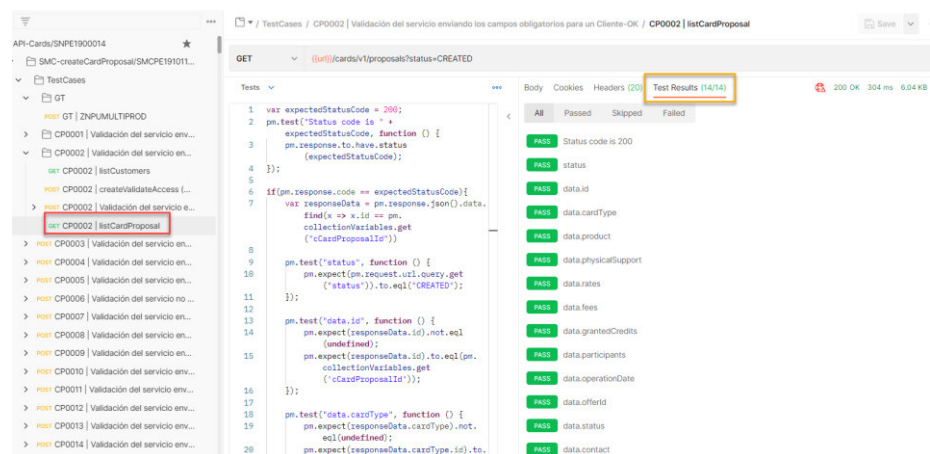


Nota. En la figura se visualiza los test scripts implementados y el resultado exitoso de todos ellos. Fuente: Adaptada de Postman.

- 7) Finalmente se ejecuta el servicio listado de datos de la solicitud. Este servicio nos permite revalidar que los datos enviados en la ejecución anterior (servicio que da de alta la solicitud) se hayan guardado satisfactoriamente.

Figura 57

Ejecución del servicio “Listar solicitud de la tarjeta”



Nota. Fuente: Adaptada de Postman.

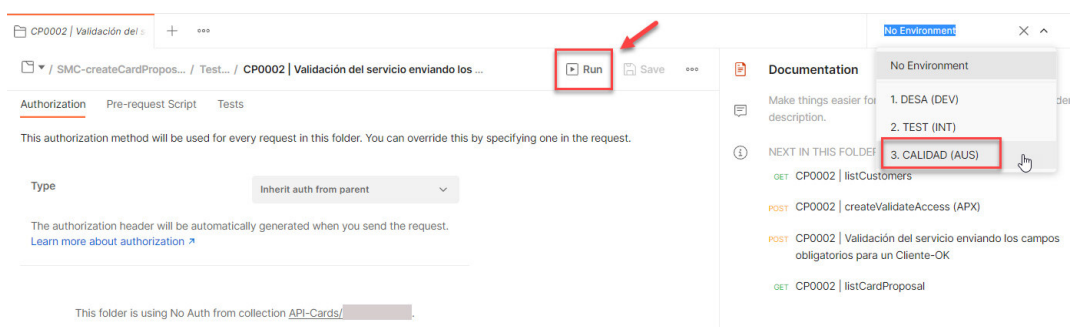
3.3.1.2. Ejecución de la subcarpeta con el caso de prueba implementado

Esta ejecución permite realizar de forma más rápida la ejecución del caso de prueba, para ello se sigue los siguientes pasos:

- 1) Se abre la sub carpeta del caso de prueba a ejecutar, en esta sección se escoge el entorno de prueba y se selecciona el botón Run, tal como se muestra en la siguiente imagen.

Figura 58

Subcarpeta del caso de prueba

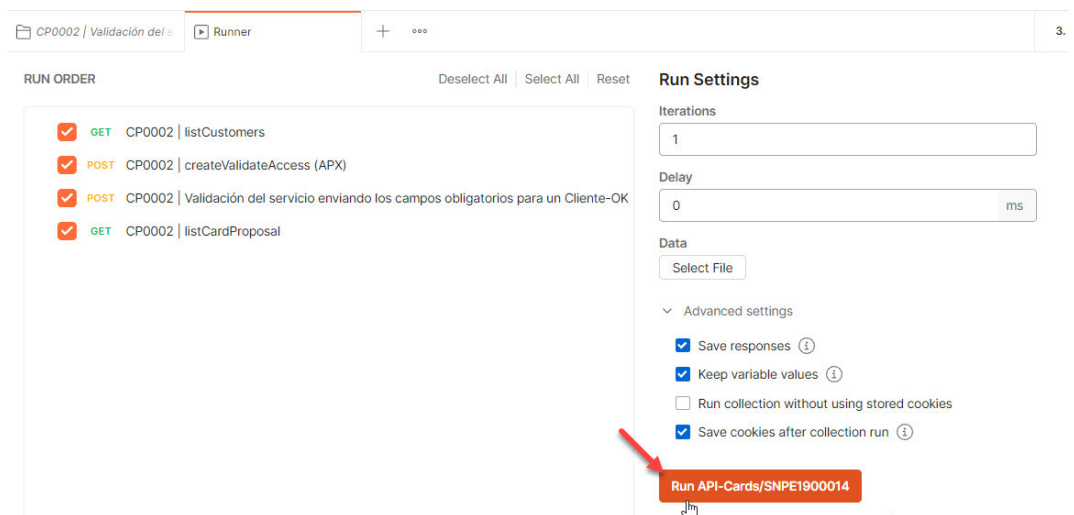


Nota. Fuente: Adaptada de Postman.

- 2) En la siguiente ventana, se verifica algunos datos de configuración adicional y se ejecuta la prueba seleccionando el botón Run.

Figura 59

Ventana de configuración

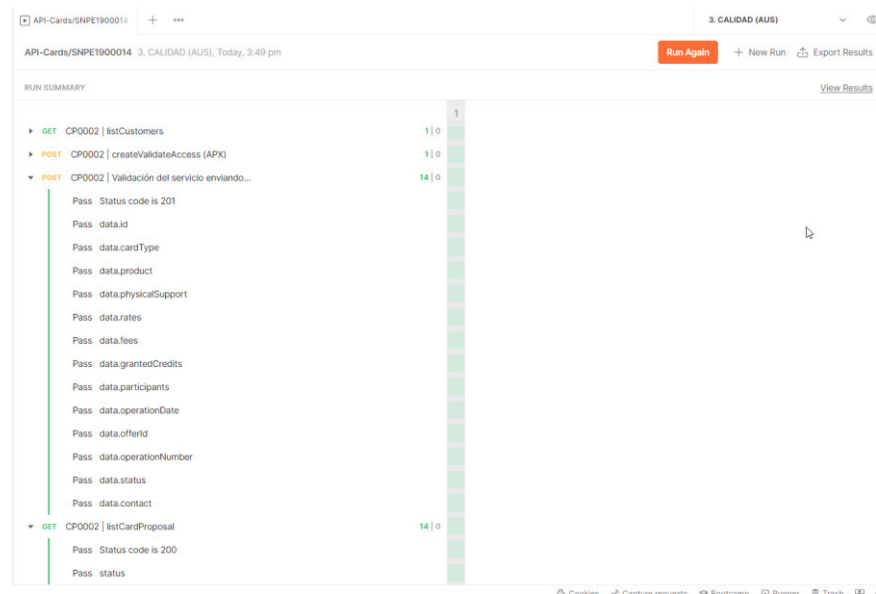


Nota. Fuente: Adaptada de Postman.

- 3) Finalmente se visualiza un resumen con la ejecución del caso de prueba.

Figura 60

Resumen de ejecución



Nota. Fuente: Adaptada de Postman.

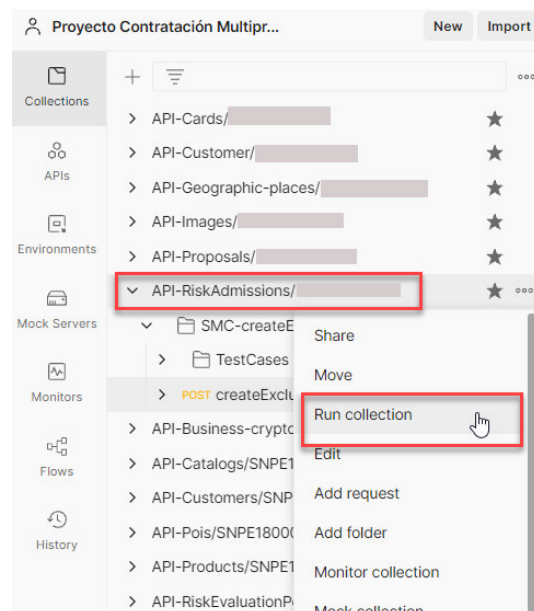
3.3.1.3. Ejecución de colección completa del API

Esta ejecución se realiza para las pruebas de regresión automatizadas.

- 1) Se selecciona con clic derecho la colección de la API y se escoge la opción “Run collection”.

Figura 61

Opción para ejecutar la colección API

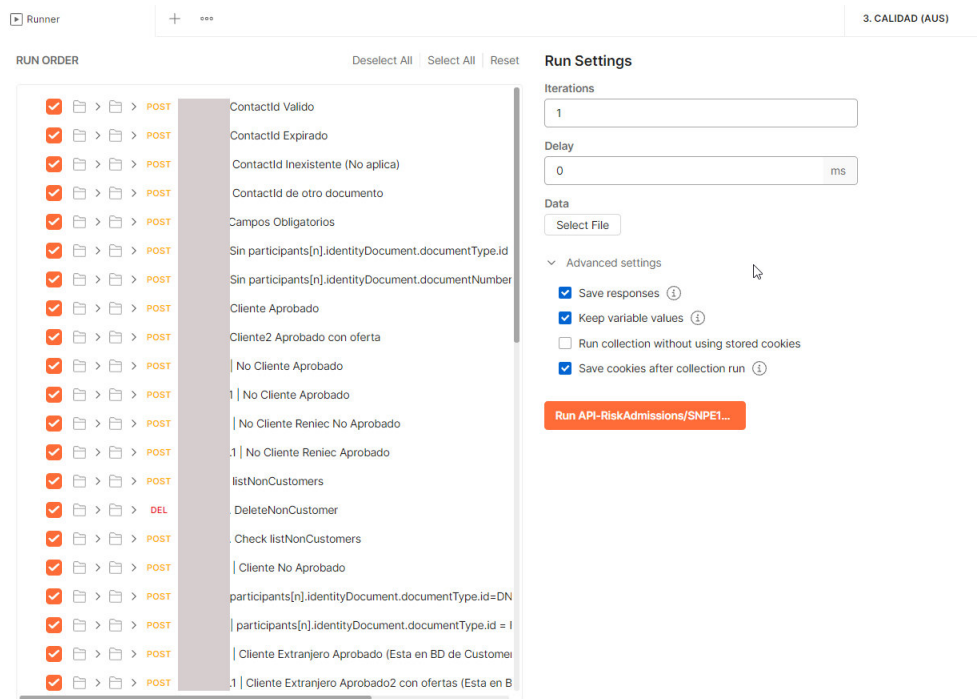


Nota. Fuente: Adaptada de Postman.

- En la siguiente ventana, se verifica algunos datos de configuración adicional y se ejecuta la prueba seleccionando el botón “Run collection API”

Figura 62

Ventana de configuración de la colección API

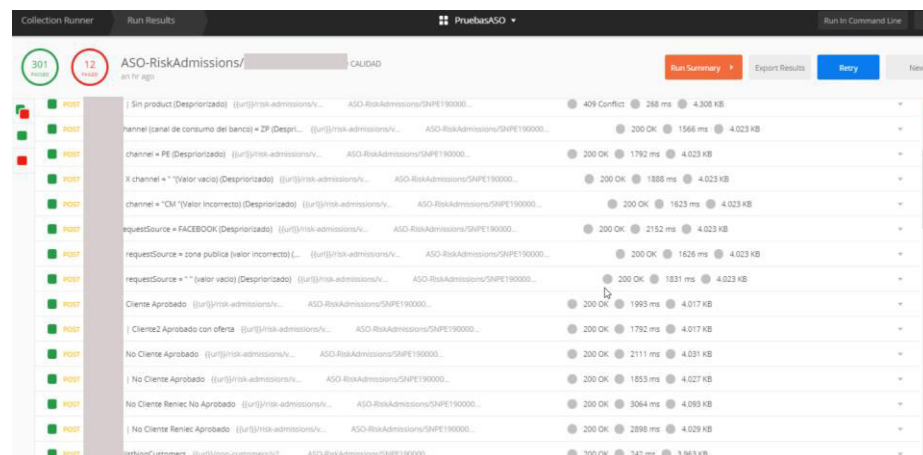


Nota. Fuente: Adaptada de Postman.

- Finalmente se visualiza un resumen con la ejecución de todos los casos de prueba de la API.

Figura 63

Resumen de ejecución de la colección API



Nota. Fuente: Adaptada de Postman.

3.3.2. Gestión de errores y/o defectos

Después de la ejecución de cada caso de prueba se debe actualizar su estado en el documento C204.

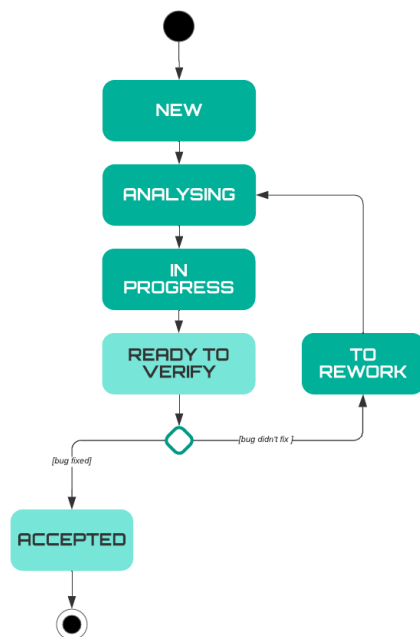
Si la ejecución del caso de prueba no devuelve el resultado esperado, la prueba se considera con estado insatisfactorio y se procede a reportar el error.

Para reportar el error se usa la herramienta JIRA, y seguiremos la siguiente estructura:

- **Título:** El título del error contiene el nombre de la capa en el que se dio el error (capa del servicio multicanal o capa backend), seguido del tipo de prueba (Obligatoriedad, valor invalido, longitud de campo o funcional), una breve descripción y finalmente el nombre del servicio.
- **Tipo:** El error se reporta en una sub tarea generada de la historia de pruebas.
- **Prioridad:** Asignamos la misma prioridad que se completó en el diseño de pruebas.
- **Descripción:** Colocamos el código y título del caso de prueba, una descripción explicando el error, se adjuntan evidencias del documento técnico en caso sea necesario y la ejecución con el error. Finalmente se adjunta el código de la traza de error.
- **Asignación:** Si el error corresponde a la capa del servicio multicanal se le asigna al gestor de servicio, y si el error corresponde al backend se le asigna al desarrollador responsable de la construcción de la transacción backend.
- **Estado:** El error/defecto se crea con estado “Nuevo”, pasa a estado “Analizando” cuando el desarrollador o gestor de servicio inicie con la revisión, luego pasara al estado “En Progreso” cuando se inicie con la corrección. En cuanto es corregido se asigna al analista QA y el estado cambia a “Listo para Verificar”. Finalmente se revalida, si es corregido pasa a estado “Aceptado” y si persiste pasa al estado “Rehacer”.

Figura 64

Ciclo de vida del error/defecto



Nota. Fuente: Elaboración propia.

En la siguiente figura se muestra un error reportado para caso de prueba de obligatoriedad.

Figura 65

Reporte de un error por obligatoriedad

2. [ASO][OBLIGATORIEDAD] Error participants[n].participant.personType - servicio modifyCardProposal

Edit Add comment Assign More Ready To Deploy Ready To Verify Workflow

Details

Type: Sub-task Status: **ACCEPTED** (View Workflow)
Priority: High Resolution: Done
Affects Version/s: None Fix Version/s: None
Labels: None

Description

x CP0050 | participants[n].participant.personType campo obligatorio-NOK
tu apoyo con el siguiente error:
De acuerdo al SMC este campo participants[n].participant.personType, esta considerado como obligatorio, sin embargo el servicio ASO no esta haciendo dicha validación.

SMC **Postman**

• "consumer-request-id": "18a854e4-6b4f-4957-84ed-d9afd0d917f8",

Attachments

Drop files to attach, or browse.

People

Assignee: Assign to me
Reporter: cluny.rojas.4
Votes: 0
Watchers: 3 Stop watching

Dates

Created:
Updated:
Resolved:

Development

Create branch

Agile

Completed Sprint:

Nota. Fuente: Adaptada de Jira.

En el documento C204 se añade una columna “Ticket de error” para registrar el ticket jira de error generado, este excel ayuda a llevar el seguimiento de cada error.

Figura 68

Registro de tickets de error

DETALLE DE CASOS DE PRUEBA									CODIGO
CodCaso	Caso de prueba	Prioridad	Paso	Descripción	Resultado Esperado	Ciclo De Prueba	Motivo de Prueba	STATUS	Ticket de Error
CP001	i) campo obligatorio-NOK	Media	1	se)	Status Code: 400 ("messages": [	1	API Testing	Satisfactorio	
CP002	ii) valor Invalido-NOK	Media	1	envia un valor	Status Code: 400 ("mes	1	API Testing	Satisfactorio	
CP003	iii) valor Invalido-NOK	Media	1	a un valor Invalido	Status Code: 400 ("mes	1	API Testing	Insatisfactorio	PA-1046
CP004	iiii) longitud mayor a 36-NOK	Media	1	cuando sea mayor	Status Code: 400 ("mes	1	API Testing	Satisfactorio	
CP005	v) longitud mayor a 1-NOK	Media	1	ndo sea mayor a	Status Code: 400 ("mes	1	API Testing	Satisfactorio	
CP006	documentos desde BD-OK	Alta	1	'ce=0	Status code: 200 OK	1	API Testing	Satisfactorio	
CP007	documentos desde servicios ent-OK	Alta	1	'ce=1	Status code: 200 OK	1	API Testing	Satisfactorio	
CP008	documentos Generales-OK	Alta	1	Generales, reamiento es	Status code: 200 OK	1	API Testing	Insatisfactorio	PA-1005
CP009	documentos Adicionales-OK	Alta	1	Adicionales.	Status code: 200 OK	1	API Testing	Insatisfactorio	PA-1010
CP010	documentos solicitud Cancelada-OK	Alta	1		Status code: 200 OK	1	API Testing	Insatisfactorio	PA-1043
		Alta	1	Generales.	Status code: 200 OK	1	API Testing	Satisfactorio	

Nota. Fuente: Elaboración propia.

3.3.3. Pruebas de confirmación

Concluida la primera ejecución de cada caso de prueba correspondiente al API y teniendo el registro de los errores encontrados, el desarrollador revisa y corrige el bloque de errores reportados. El analista QA recibe la confirmación y actualización de los tickets jira y procede con la ejecución uno a uno o por subcarpetas de las pruebas de confirmación para revalidar si el error ha sido corregido.

3.3.4. Pruebas de Regresión

Concluida la ejecución de pruebas de confirmación del bloque de errores reportados por cada API, se ejecuta las pruebas de regresión automatizadas a toda la colección del componente para validar que los demás casos de prueba no hayan sido afectados por los cambios que los desarrolladores realizaron.

3.3.5. Documentación de evidencias

Las evidencias de prueba se registraran en el documento T907, estas evidencias son de las pruebas ejecutadas en el ambiente de Calidad. El documento tiene una sección de resumen y otra propiamente para las evidencias.

En la sección de resumen se completa los siguientes datos: código del proyecto, título del proyecto, ámbito del aplicativo, fecha de creación y actualización, tabla con el historial de versiones, nombre y descripción de la API.

Figura 69

Sección de resumen del documento de evidencias

T907 - createCardProposal .DOCX ☆ 📁 ☁

Archivo Editar Ver Insertar Formato Herramientas Ayuda

90% Texto norm... Arial 10 B I U A

T907 - Evidencia de Pruebas

Ámbito	N° Orden de Trabajo	Título de Orden de Trabajo
Distribuido		Contratación Multiproducto Web TC

Fecha creación	Fecha actualización	Creado por
23/06/2020	Ingresar fecha	Cluny Rojas

Objetivo del documento
El objetivo de este documento es registrar las evidencias por cada Caso de Prueba descrito en el C204.

Bitácora de cambios al documento			
Versión	Fecha	Autor Modificación	Descripción del cambio
1	23/06/2020	Cluny Rojas	Creación del documento
1.5	27/07/2020	Cluny Rojas	Actualización documento (+GABY)

Ejecución de los Casos de Prueba del documento C204 y registro de Evidencias

I. Definición

Componente: cards
Descripción: Pruebas unitarias en diferentes escenarios de pruebas

II. Prueba detallada de componente

Prueba de Componente: cards
Descripción: Llamado y retorno de la información del SMC createCardProposal

Ejecución del programa:

DATOS GENERALES:
Método: POST
URL: http /cards/v1/proposals
Header:

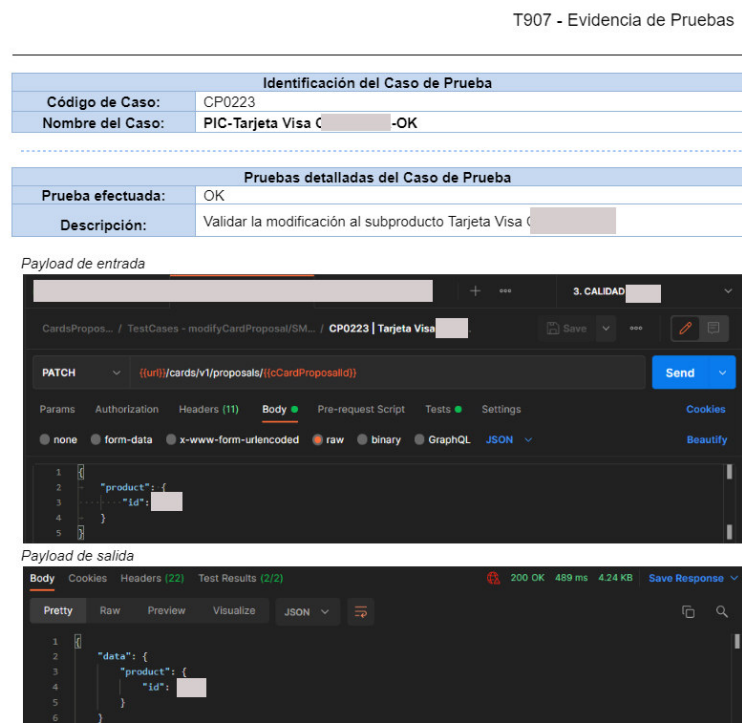
POST {{url}}/cards/v1/proposals

Nota. Fuente: Elaboración propia.

En la sección de la evidencia de pruebas se coloca los datos generales de la prueba del servicio multicanal: Método, URL y cabeceras. Luego tenemos dos tablas que serán registradas por cada caso de prueba, en la primera se registra el código y nombre del caso de prueba y en la segunda se registra la prueba efectuada y la descripción.

Figura 70

Sección de evidencias de la prueba



Nota. Fuente: Elaboración propia.

De este modo se documentan las evidencias de cada una de las API.

Figura 71

Lista de documentos de evidencia

Mi unidad > > Pruebas [C204-T907] > T907 ▾

Nombre ↑	Propietario
T907 - Evidencias de Pruebas - customers-listCustomerCivilRegistries.docx	yo
T907 - Evidencias de Pruebas - businessCryptography-createCryptoDocument.docx	yo
T907 - Evidencias de Pruebas - cards-createCardProposal.docx	yo
T907 - Evidencias de Pruebas - cards-listCardProposals.docx	yo
T907 - Evidencias de Pruebas - cards-modifyCardProposal.docx	yo
T907 - Evidencias de Pruebas - geographicplaces-listGeographicPlaces.docx	yo
T907 - Evidencias de Pruebas - images-searchImagesCovers.docx	yo
T907 - Evidencias de Pruebas - pois- listPoisBranch.docx	yo
T907 - Evidencias de Pruebas - products-listCardProducts.docx	yo
T907 - Evidencias de Pruebas - proposal-createQuestionnairesValidateAccess.docx	yo
T907 - Evidencias de Pruebas - riskAdmissions-createExclusionFiltersValidations.docx	yo
T907 - Evidencias de Pruebas - riskEvaluationPerson-verifyRiskEvaluationPerson.docx	yo

Nota. Fuente: Elaboración propia.

4. Gestión de QA para Producción

Una vez concluida las pruebas en calidad y cerradas las incidencias reportadas se inicia con las gestiones para el pase a producción

El procedimiento establecido para realizar el pase a producción inicia teniendo el visto bueno de las pruebas de no impacto.

4.1. Solicitud de Pruebas de No Impacto

Las pruebas de no impacto consisten en ejecutar el bloque de pruebas que cada canal tiene de acuerdo a la funcionalidad que maneja, el objetivo es asegurar que los cambios efectuados en la API no afecten a los canales que consumen dicho componente.

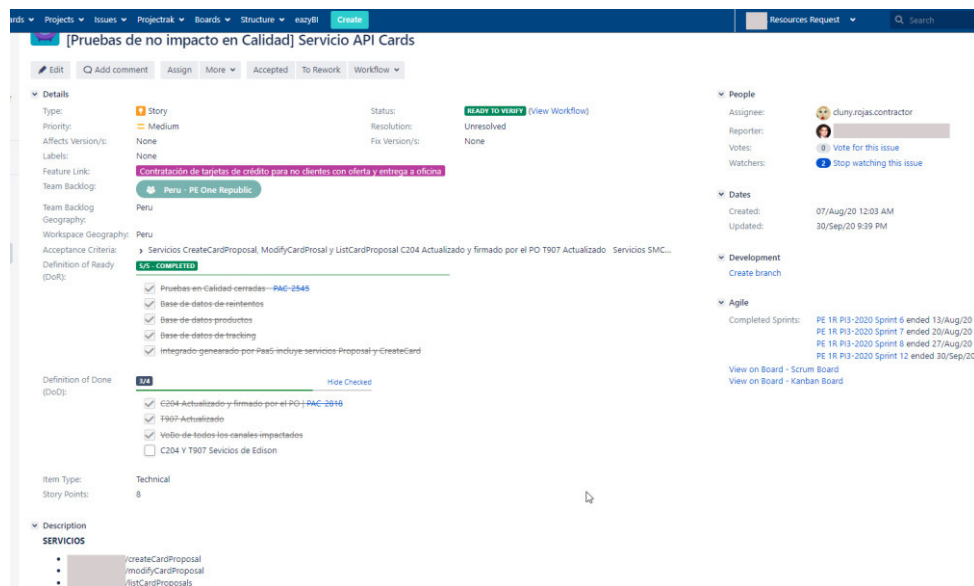
Esta gestión sigue un procedimiento ya definido por el área de Arquitectura de servicios y se coordina con el gestor de servicio.

El gestor de servicio se encarga de preparar el componente a desplegar (versión productiva más el cambio), fijar el día y la hora de las pruebas e identificar a los canales que consumen el API en producción.

El analista QA como parte del equipo scrum y con apoyo del PO y scrum master se encargan de gestionar y solicitar el apoyo de estas pruebas con los responsables de los canales impactados. Así mismo el analista QA realiza pruebas de regresión de la(s) API involucrada(s) como parte del canal solicitante.

Figura 72

Historia de usuario de las pruebas de no impacto API Tarjetas

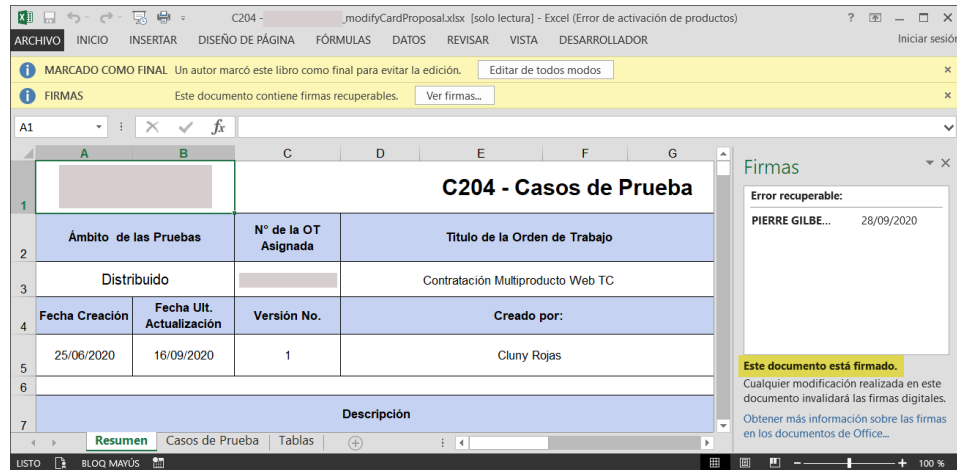


Nota. Fuente: Adaptada de Jira.

Por otro lado el product owner revisa y da la conformidad (firma digital) de los documentos de prueba.

Figura 73

Firma digital del documento de especificación de pruebas



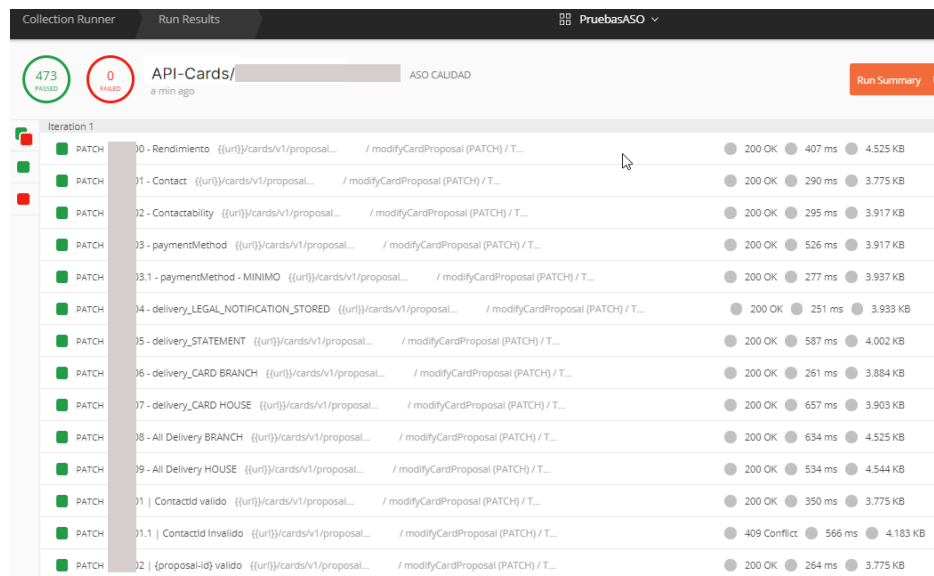
Nota. Fuente: (Empresa, 2020).

El día de las pruebas, el gestor de servicio despliega la versión del componente e informa la hora de inicio de las pruebas, que duran en promedio 2 horas como mínimo.

El analista QA ejecuta la colección del API, es decir realiza una prueba de regresión de todos los casos de prueba.

Figura 74

Pruebas de regresión API Tarjetas



Nota. Fuente: Adaptada de Postman.

El Scrum master coordina con los responsables de los canales impactados para que realicen sus pruebas y den su conformidad.

Figura 75

Visto Bueno de los canales impactados



Nota. Fuente: (Empresa, 2020).

4.2. Certificación Octa y pruebas de rendimiento

Esta etapa, se realiza bajo un procedimiento ya definido por la entidad financiera, donde se considera como uno de los requerimientos importantes para el pase a producción. El objetivo es medir el rendimiento de la API. La conformidad de estas pruebas se coordina y gestiona con el área de arquitectura octa.

Los escenarios de prueba para la ejecución de las pruebas de rendimiento se basan en el resultado de completar la siguiente lista de verificación de análisis brindada por el área de arquitectura.

- Fecha prevista fin de construcción
- Fecha prevista de pase a producción
- Tipo de aplicación backend
- Tipo del componente a certificar
- Descripción de la funcionalidad del componente.
- Volumetría esperada en producción
- Responder si corresponde a una aplicación de arquitectura

- Responder si es una operación crítica
- Responder si considera necesario las pruebas de rendimiento.

El visto bueno de la certificación se debe gestionar a través de un ticket de atención.

Figura 76

Ticket de atención para certificación de pruebas de rendimiento

Ether Platform Help Desk / OTROS / OTRSSD-103

CERTIFICADO DE MEDICIONES, COMPONENTE: [REDACTED] VOLUMETRÍA ESPERADA EN PRODUCCIÓN: Más de 40 peticiones/hora (uso bajo).

Activity

Your request status changed to **Closed**. [REDACTED] **LATEST**

Resolution changed to **Done** [REDACTED]

Administrador [REDACTED]
This issue has been closed automatically.
The issue has been in the "Ready to Verify" for more than 7 days.

Your request status changed to **Ready To Verify** with resolution **Resolved**. [REDACTED]

[User] [REDACTED]
Se adjunta informe técnico: <https://docs.google.com/presentation/d/1DiPh7g6BdXPASdgViuZs8ynlxxFuFA7DU5g3Q4VvOo/edit?usp=sharing>

cluny.ro [REDACTED]
Se realizaron las 25 ejecuciones solicitadas.

Shared with
cluny.rojas.contractor
Creator

Nota. Fuente: (Empresa, 2020)

Teniendo el resultado de la lista de verificación y un técnico asignado para el seguimiento de las pruebas, se realiza la implementación del script en la herramienta JMeter.

Se configura el escenario de prueba, se preparan los datos de prueba y se procede a ejecutar.

Finalmente se obtienen los resultados y se envían al técnico asignado para que realice el monitoreo correspondiente y otorgue la conformidad.

4.3. Cierre de pruebas

Teniendo la conformidad de las pruebas de no impacto y la certificación de las pruebas de rendimiento, se inician las gestiones para el pase a producción.

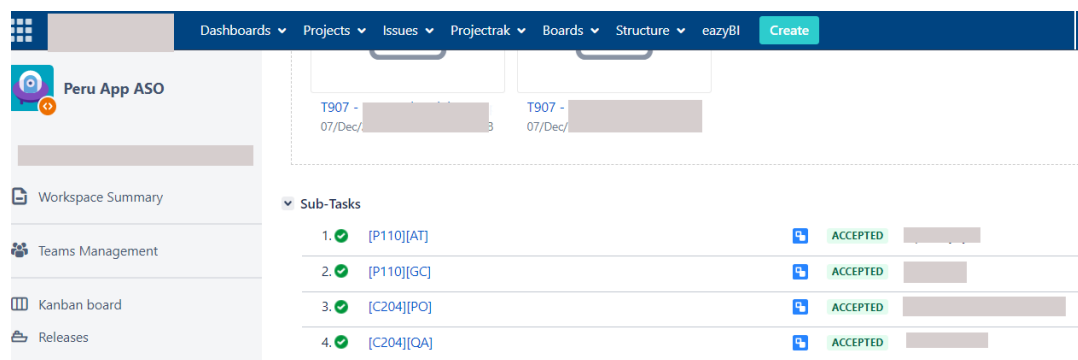
En el ticket de pase se adjunta la ficha de pase con la firma digital del líder técnico o scrum master del equipo y los documentos de prueba con la firma digital del PO.

El ticket de pase pasa por un proceso de vistos buenos para ser considerado “Listo para producción”. El visto bueno se da en las sub tareas del ticket asignándole el estado “Accepted”.

- El primer visto bueno es por parte del líder técnico.
- El segundo visto bueno es por parte de gestión de cambios.
- El tercer visto bueno es por parte del PO.
- El último visto bueno es por parte del líder QA de la entidad financiera.

Figura 77

Vistos buenos para pase a producción.



Nota. Fuente: (Empresa, 2020)

En esta etapa final se valida que todos los reportes de defectos estén cerrados, y se almacena en un solo repositorio, los datos de prueba, scripts de implementación, documentos y todo los entregables generados durante el proceso de pruebas.

Figura 78

Repositorio de entregables de prueba

Mi unidad > Proyectos > Proyecto Contratación de Multiproductos

Los accesos directos ayudan a simplificar Mi unidad
En las próximas semanas, los elementos que se encuentren en más de una carpeta se reemplazarán por accesos directos. El acceso a los archivos y las carpetas no cambiará. [Más información](#)

Nombre	Propietario	Última modificación	Tamaño de archivo
DataExcelPrueba	yo	8 oct 2020 yo	—
DataPrueba	yo	12 feb 2020 yo	—
Evidencias	yo	7 ene 2020 yo	—
Guías	yo	9 feb 2021 yo	—
Jmeter	yo	3 feb 2020 yo	—
Otros	yo	8 jul 2020 yo	—
Postman	yo	9 nov 2020 yo	—
Pruebas [C204-T907]	yo	30 mar 2022 yo	—
PruebasReportes	yo	5 nov 2020 yo	—
Reportes	yo	6 oct 2020 yo	—
C204 - Casos de Prueba V.1 (4).xlsx	yo	15 may 2020 yo	65 kB
T907 - Evidencias de Pruebas v.1.docx	yo	15 may 2020 yo	73 kB

Nota. Fuente: Elaboración Propia.

3.3. Evaluación

3.3.1. Evaluación Económica

La siguiente evaluación económica se basa en un supuesto estimado.

- Recurso Humano

La implementación del proceso de pruebas aplicado al proyecto de software de la entidad bancaria se desarrolló con un Analista de aseguramiento y control de calidad y dos Probadores junior. El analista laboró durante los 12 meses que duró el proyecto mientras que los probadores se incorporaron al proyecto a partir del séptimo mes.

Tabla 11

Evaluación Económica

Recursos	Sueldo Mensual	% Trabajo	Gasto Mensual	Tiempo Trabajo	Gasto Total
Analista QA	S/. 6 000.00	50 %	S/. 3 000.00	12 meses	S/. 36 000.00
Tester Junior 1	S/. 3 000.00	50 %	S/. 1 500.00	6 meses	S/. 9 000.00
Tester Junior 2	S/. 3 000.00	20 %	S/. 600.00	6 meses	S/. 3 600.00
					S/. 48 600.00

Nota. Fuente: Elaboración propia.

- **Herramientas**

Las herramientas con las que se trabajaron como Postman, JMeter y Java JDK son versiones gratuitas. Con respecto a Jira, Git, Butbucket la empresa tiene la licencia contratada, por lo que no representaron un gasto adicional.

3.3.2. *Evaluación Técnica*

La evaluación técnica del presente trabajo está basada en la calidad del proceso de prueba planteado para las pruebas de API REST.

Los modelos, formatos y procedimientos definidos en cada actividad de prueba y la implementación de test scripts de validación para la automatización de las pruebas de regresión, benefician el tiempo de ejecución de cada prueba.

La siguiente tabla muestra una comparación entre haber seguido un proceso de pruebas basado en metodologías y procedimientos definidos y no aplicar ningún proceso.

Tabla 12

Tabla comparativa de evaluación técnica

Proceso de pruebas definido para API	Sin proceso de pruebas para API
Cuenta con una estrategia y plan de pruebas.	No se sigue un plan de pruebas.
El diseño de casos de prueba sigue un formato y procedimiento definido.	No se realiza el diseño de pruebas.
La implementación de pruebas sigue un procedimiento definido y se realiza scripts de automatización.	Usa checklist de verificación.
Gestión de entornos de prueba con variables parametrizadas.	Entornos de prueba configurados con valores estáticos.
Tres opciones de ejecución de pruebas con implementación de test scripts. Esto permite que la validación del resultado esperado sea automático.	Una opción de ejecución, y la validación del resultado esperado se hacen de forma manual.
Pruebas de regresión automatizadas.	Pruebas de regresión manuales.
Se cuenta con un repositorio con todos los documentos, archivos, scripts, datos y demás entregables del proceso de prueba.	Cada miembro del equipo guarda sus entregables.

Nota. Fuente: Elaboración Propia.

CAPÍTULO IV. REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

La autora del presente trabajo cumplió el rol de Analista QA, formando parte de un equipo Scrum del proyecto Contratación de multiproductos, que había recuperado su continuidad en el desarrollo después de haber entrado en una fase de incertidumbre. El proyecto tenía una gran expectativa para la entidad bancaria. El reto de la autora era garantizar las pruebas del desarrollo backend, en específico de las API. Garantizar el correcto funcionamiento de los componentes construidos y cumplir los tiempos establecidos.

La autora aplicó sus conocimientos, experiencia y se apoyó de procedimientos y metodologías definidas en guías de pruebas para definir los lineamientos, procedimientos y realizar la implementación de las pruebas de API. A pesar que las pruebas realizadas fueron satisfactorias para el resultado final del proyecto, la autora considera puntos de mejora basados en la experiencia definida en el presente informe.

- La definición de casos de prueba pueden ser refinadas.
- El formato de la especificación de pruebas puede ser mejorada y rediseñada.
- Realizar guías con los modelos de los test scripts de validación para que otros proyectos puedan usarlos como referencia.
- Establecer el proceso, procedimientos y automatización de estas pruebas en equipos de otros proyectos.
- Mejorar el procedimiento de gestión de errores y defectos.
- Evaluar el uso de herramientas de gestión de pruebas, gestión de errores y defectos.
- Evaluar el uso de otras herramientas para automatización.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Como conclusión general se logró definir e implementar los procedimientos de cada una de las actividades del proceso de pruebas para API REST, así como se automatizaron las pruebas de regresión logrando de esta manera ejecutar dichas pruebas en un tiempo mucho menor a lo que generalmente se demora realizándolas de forma manual.

Como parte de los objetivos específicos definidos se concluye lo siguiente:

- Se logró definir una estrategia de pruebas acorde a la necesidad del proyecto y se elaboró el plan de pruebas.
- Se desarrollaron los procedimientos y lineamientos definidos para cada tarea del análisis y diseño de pruebas. Formatos/modelos de diseño y especificación para casos de prueba de API REST.
- Se desarrollaron los procedimientos y lineamientos definidos para cada tarea de la implementación de pruebas. Test scripts para la validación de resultados esperados y automatización de las pruebas.
- Se ejecutaron las pruebas y se generaron reportes de gestión de errores y defectos.
- Se recopilaron los entregables en un único repositorio del proyecto para su control y gestión de calidad.

5.2. Recomendaciones

En base a la experiencia adquirida la autora considera los siguientes puntos de mejora:

- Alinear el proceso y procedimiento desarrollado en el presente trabajo a los diferentes proyectos planteados por la entidad financiera con la apertura de aceptar propuestas de mejora para que esté acorde a las necesidades de cada proyecto.
- Evaluar el uso de herramientas de gestión de pruebas, gestión de errores y defectos.
- Evaluar el uso de otras herramientas para automatización

5.3. Fuentes de Información

- Apache Software Foundation. (2022). *jmeter.apache.org*. Obtenido de <https://jmeter.apache.org/>
- appmaster. (2022). *appmaster.io*. Obtenido de <https://appmaster.io/es/blog/que-es-la-api-rest-y-en-que-se-diferencia-de-otros-tipos>
- Atlassian. (2022). *atlassian.com*. Obtenido de <https://www.atlassian.com/>
- Fowler, M., Marick, B., Beck, K., & Beedle, M. (2001). *agilemanifesto.org*. Obtenido de <http://agilemanifesto.org/>
- Hamilton, T. (2022). *guru99.com*. Obtenido de <https://www.guru99.com/testlink-tutorial-complete-guide.html>
- Inflectra. (2022). *inflectra.com*. Obtenido de <https://www.inflectra.com/rapise/highlights/api-testing.aspx>
- ISTQB. (2018). *Programa de estudio ISTQB Foundation*.
- Jordán Marcos, D. (2020). *Estudio sobre las metodologías ágiles y metodologías tradicionales para gestión de proyectos de software*. España.
- Niittyviita, S. (2016). The Cost Efficiency of Exploratory Testing: ISTQB Certified Testing Compared with RST.
- Postman. (2022). *postman.com*. Obtenido de <https://www.postman.com/>
- Schwaber, K., & Sutherland, J. (2020). La Guía Definitiva de Scrum: Las Reglas del Juego . En K. Schwaber, & J. Sutherland, *La Guía de Scrum*.
- SCRUMstudy. (2017). *GUÍA SBOKTM*.
- TesteandoSoftware. (2014). *testeandosoftware.com*. Obtenido de <https://testeandosoftware.com/>
- todo.ly. (2020). *todo.ly*. Obtenido de <http://todo.ly/apiwiki/?projects>