



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Sistemas

**Implementación del framework de automatización
Serenity BDD para la ejecución de pruebas de software
para la empresa CreceMas**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Sistemas

AUTOR

José Fernando AYASTA CACHAY

ASESOR

Arturo Alejandro BARTRA MORE

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

Ayasta, J. (2022). *Implementación del framework de automatización Serenity BDD para la ejecución de pruebas de software para la empresa CreceMas*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	JOSÉ FERNANDO AYASTA CACHAY
Tipo de documento de identidad	DNI
Número de documento de identidad	47135000
URL de ORCID	https://orcid.org/0000-0001-5995-2645
Datos de asesor	
Nombres y apellidos	ARTURO ALEJANDRO BARTRA MORE
Tipo de documento de identidad	DNI
Número de documento de identidad	40233946
URL de ORCID	https://orcid.org/0000-0002-3411-7456
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	William MartinEnríquez Maguiña
Tipo de documento	DNI
Número de documento de identidad	06179457
Miembro del jurado 1	
Nombres y apellidos	Raúl MarceloArmas Calderón
Tipo de documento	DNI
Número de documento de identidad	07156168
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 Sistemas

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 20:05 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 Mg. Enríquez Maguiña William Martin (Presidente), Ing. Armas Calderón Raúl Marcelo (Miembro) y el Ing. Bartra More Arturo Alejandro (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/hqu-jkha-bft>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DEL FRAMEWORK DE AUTOMATIZACIÓN SERENITY BDD PARA LA EJECUCIÓN DE PRUEBAS DE SOFTWARE PARA LA EMPRESA CRECEMAS”**, por el Bachiller Ayasta Cachay José Fernando; para obtener el Título Profesional de Ingeniero de Sistemas.

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

El 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, el Bachiller obtuvo la nota de **16 (DIECISÉIS)**.

A continuación el Presidente de Jurados el Mg. Enríquez Maguiña William Martin, declara al Bachiller **Ingeniero de Sistemas**.

Siendo las 20:59 horas, se levantó la sesión.

Presidente

Mg. Enríquez Maguiña William Martin

Miembro

Ing. Armas Calderón Raúl Marcelo

Miembro Asesor

Ing. Bartra More Arturo Alejandro



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA
Escuela Profesional de Ingeniería de Sistemas

INFORME DE EVALUACIÓN DE ORIGINALIDAD

1. Facultad de Ingeniería de Sistemas e Informática
 2. Escuela Profesional de Ingeniería de Sistemas
 3. Autoridad académica que emite el informe de originalidad
Directora(e) de la EPIS
 4. Apellidos y Nombres de la autoridad académica
Dra. Luzmila E. Pró Concepción
 5. Operador del programa informático de similitudes
Dra. Luzmila E. Pró Concepción
 6. Documento evaluado
Título de pregrado: "Implementación del framework de automatización Serenity BDD para la ejecución de pruebas de software para la empresa CreceMas"
 7. Autor del documento
Bach. Ayasta Cachay, José Fernando
 8. Fecha de recepción del documento **10/09/2022**
 9. Fecha de aplicación del programa informático de similitudes **10/09/2022**
 10. Software utilizado
 - Turnitin
 11. Configuración del programa detector de similitudes
 - Excluye textos entrecomillados
 - Excluye bibliografía
 - Excluye cadenas menores a 40 palabras
 12. Porcentaje de similitudes según programa detector de similitudes **6 (seis) %**
 13. Fuentes originales de las similitudes encontradas
Se adjunta en el anexo 1
 14. Observaciones
-
15. Calificación de originalidad
 - Documento cumple criterios de originalidad, sin observaciones ☒
 - Documento cumple criterios de originalidad, con observaciones ☐
 - Documento no cumple criterios de originalidad ☐
 16. Fecha de informe **17/09/2022**



Firmado digitalmente por PRO
CONCEPCION Luzmila Elisa FAU
20148092282 soft
Motivo: Soy el autor del documento
Fecha: 14.11.2022 11:56:18 -05:00

Firma de evaluador
Dra. Luzmila E. Pró Concepción
Directora (e) de la EPIS



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA
Escuela Profesional de Ingeniería de Sistemas

ANEXO 1

Fuentes originales de las similitudes encontradas

1. hdl.handle.net: 4%
2. www.atlassian.com: 1%
3. www.ibm.com: 1%



UNMSM

Firmado digitalmente por PRO
CONCEPCIÓN Luzmila Elisa FAU
20148092282 soft
Motivo: Soy el autor del documento
Fecha: 14.11.2022 11:55:45 -05:00

Firma de evaluador
Dra. Luzmila E. Pró Concepción
Directora (e) de la EPIS

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

**IMPLEMENTACIÓN DE FRAMEWORK SERENITY BDD DE
AUTOMATIZACIÓN PARA LA EJECUCIÓN DE PRUEBAS DE SOFTWARE
PARA LA EMPRESA CRECEMAS**

Autor: Ayasta Cachay José Fernando
Asesor: Bartra More Arturo Alejandro
Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de
Ingeniero de Sistemas
Fecha: Julio 2022

RESUMEN

El presente proyecto detalla la implementación de Serenity-BDD, un framework de automatización de ejecución de pruebas de software, para la empresa CreceMas utilizando metodología ágil Kanban, cuyo objetivo es de reducir el tiempo de ejecución de las pruebas de software enfocada en los aplicativos webs. La metodología Kanban nos permitirá visualizar el estado de las actividades requeridas, así como también el estado general del proyecto. Luego de la implementación de Serenity-BDD, se logró reducir el tiempo de ejecución de pruebas de software, permitiendo a la empresa reducir tiempos de validación de funcionalidades nuevas, existentes y realizar ejecuciones repetitivas en diferentes versiones de las aplicaciones, logrando conocer la estabilidad de sus aplicativos webs en diferentes etapas de desarrollo.

Palabras Clave: Pruebas automatizadas, Metodología ágil, Serenity-BDD

NATIONAL MAJOR UNIVERSITY OF SAN MARCOS

FACULTY OF SYSTEMS ENGINEERING AND INFORMATICS

PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING

**IMPLEMENTATION OF SERENITY BDD AUTOMATION FRAMEWORK TO
EXECUTE SOFTWARE TESTS FOR CRECEMAS COMPANY**

Author: Ayasta Cachay José Fernando
Advisor: Bartra More Arturo Alejandro
Title: Professional Sufficiency Work to opt for the Professional Title of
System Engineer
Date: July 2022

ABSTRACT

This project details the implementation of Serenity-BDD, a software testing automation framework, for the company Crecemas using agile Kanban methodology, whose objectives is to reduce the execution time of software tests for web applications. The Kanban methodology will allow us to visualize the status of the required activities, as well as the general status of the project. After the implementation of Serenity-BDD, the execution time of software tests will be reduced, allowing the company to reduce validation times of new and existing functionalities and by carrying out repetitive executions in different versions of the applications, obtaining to know the stability of its web applications in different stages of development.

Keywords: Automated tests, Agile Methodology, Serenity-BDD

INDICE GENERAL

RESUMEN	ii
ABSTRACT	iii
INDICE GENERAL	5
INDICE DE FIGURAS	8
INDICE DE TABLAS	9
CAPITULO I TRAYECTORIA PROFESIONAL	10
1.1.	11
1.2.	11
1.3.	14
CAPITULO II CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA	14
2.1.	15
2.2.	15
2.3.	15
2.4.	16
2.5.	16
2.6.	17
CAPITULO III ACTIVIDADES DESARROLLADAS	18
3.1.	18
3.1.1. DEFINICIÓN DEL PROBLEMA	18
3.2.	18
3.2.1. OBJETIVOS	19
3.2.2. ALCANCE	19
3.2.3. ETAPAS Y METODOLOGÍA	20
3.2.4. FUNDAMENTOS UTILIZADOS	24
METODOLOGÍA ÁGIL	24
KANBAN	24
PRUEBAS DE SOFTWARE	24
1. Pruebas de Regresión.	25
2. Pruebas de Humo.	25
FRAMEWORKS DE AUTOMATIZACIÓN	25
1. Behavior Drive Development (BDD)	26
2. Serenity-BDD	26
3. Cucumber	26

4.	Cucumber-Reports	27
5.	Selenium	27
6.	Maven	28
	PATRON DE DISEÑO	28
1.	Page Object Model	28
	INTEGRACIÓN CONTINUA (CI)	29
1.	GITLAB	29
2.	GITLAB CI/CD	30
3.	GIT	30
3.2.5.	IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS	30
	ITERACIÓN 1	31
1.	CREACIÓN EL TABLERO DE KANBAN	31
2.	DEFINICIÓN DEL ALCANCE DEL FRAMEWORK	32
3.	IMPLEMENTACIÓN DE SERENITY-BDD	36
4.	SELECCIÓN DE CASOS DE PRUEBA	38
5.	CONFIGURACIÓN DEL AMBIENTE DE TRABAJO	40
	ITERACIÓN 2	40
1.	CREACIÓN DE TARJETAS KANBAN	40
2.	REUNIÓN DIARIA	41
3.	TRIAGE DE RESULTADOS DE EJECUCIONES	41
4.	DESARROLLO DE PRUEBAS SELECCIONADAS	42
5.	EJECUCIÓN DE LA SUITE DE PRUEBAS	48
6.	REUNIÓN DE ESTATUS	48
	ITERACIÓN 3	49
1.	CREACIÓN DE TARJETAS KANBAN	49
2.	REUNIÓN DIARIA	49
3.	TRIAGE DE RESULTADOS DE EJECUCIONES	49
4.	DESARROLLO DE PRUEBAS SELECCIONADAS	49
5.	EJECUCIÓN DE LA SUITE DE PRUEBAS	49
6.	REUNIÓN DE ESTATUS	50
	ITERACIÓN 4	50
1.	CREACIÓN DE TARJETAS KANBAN	50
2.	REUNIÓN DIARIA	50
3.	TRIAGE DE RESULTADOS DE EJECUCIONES	50
4.	DESARROLLO DE PRUEBAS SELECCIONADAS	50
5.	EJECUCIÓN DE LA SUITE DE PRUEBAS	51
6.	REUNIÓN DE ESTATUS	51
	ITERACIÓN 5	51
1.	CREACIÓN DE TARJETAS KANBAN	51
2.	REUNIÓN DIARIA	51
3.	TRIAGE DE RESULTADOS DE EJECUCIONES	51
4.	DESARROLLO DE PRUEBAS SELECCIONADAS	52
5.	EJECUCIÓN DE LA SUITE DE PRUEBAS	52
6.	REUNIÓN DE ESTATUS	52
3.3.	50	
3.3.1.	EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO-BENEFICIO	52
3.3.2.	INTERPRETACIÓN DEL VAN Y DEL TIR	54

CAPITULO IV	REFLEXIÓN CRÍTICA DE LA EXPERIENCIA	56
CAPITULO V	CONCLUSIONES Y RECOMENDACIONES	57
5.1.	55	
5.2.	56	
5.3.	56	

INDICE DE FIGURAS

FIGURA 1. ORGANIGRAMA DEL STARTUP CRECEMAS	15
FIGURA 2. ORGANIGRAMA DEL EQUIPO DE DESARROLLO.	20
FIGURA 3. PROCESO DE IMPLEMENTACIÓN DE LA METODOLOGÍA KANBAN	20
FIGURA 4. TABLERO KANBAN NUEVO CREADO DESDE LA HERRAMIENTA TRELLO	21
FIGURA 5. TABLERO KANBAN CON DEFINICIÓN DE COLUMNAS.	22
FIGURA 6. KANBAN TICKET DEL PROYECTO CRECEMAS.	22
FIGURA 7. TABLERO DE TRABAJO DE CRECEMAS	23
FIGURA 8. TABLA DE SEGUIMIENTO DE AVANCE DEL DESARROLLO DE PRUEBAS AUTOMATIZADAS.	24
FIGURA 9. EJEMPLO DE UNA ESPECIFICACIÓN DE UN ESCENARIO.	26
FIGURA 10. PANTALLAZO DE UN REPORTE DE SERENITY BDD DESPUÉS DE UNA EJECUCIÓN DE PRUEBAS.	27
FIGURA 11. REPRESENTACIÓN DEL MODELO DE PAGE OBJECT.	28
FIGURA 12. PROCESO DE IMPLEMENTACIÓN DEL FRAMEWORK SERENITY-BDD EN EL PROYECTO CRECEMAS.	30
FIGURA 13. TABLERO KANBAN DEL PROYECTO DE IMPLEMENTACIÓN DE FRAMEWORK SERENITY-BDD.	32
FIGURA 14. CREACIÓN DE TAREA EN EL TABLERO PARA SOPORTAR DIVERSOS NAVEGADORES.	33
FIGURA 15. CREACIÓN DE TAREA EN EL TABLERO PARA EJECUTAR PRUEBAS EN LOS AMBIENTES INDICADOS.	34
FIGURA 16. CREACIÓN DE TAREA EN EL TABLERO PARA EJECUTAR MEDIANTE ETIQUETAS.	35
FIGURA 17. EJEMPLO DE ETIQUETAS EN LAS PRUEBAS AUTOMATIZADAS YA IMPLEMENTADAS.	35
FIGURA 18. ESTRUCTURA DEL FRAMEWORK APLICANDO PAGE OBJECT MODEL	36
FIGURA 19. ESTRUCTURA DE CARPETAS DEL FRAMEWORK DE AUTOMATIZACIÓN	37
FIGURA 20. DISTRIBUCIÓN DE LOS CASOS DE PRUEBA Y SUS ESTIMACIONES PARA LA APLICACIÓN MYPE-CRECEMAS.	38
FIGURA 21. DISTRIBUCIÓN DE LOS CASOS DE PRUEBA DE SERVICIOS Y SUS ESTIMACIONES PARA LA APLICACIÓN MYPE-CRECEMAS.	39

FIGURA 22. EJEMPLO DE TARJETA CREADA EN EL PROYECTO CRECEMAS	41
FIGURA 23. PANTALLA DE LA APLICACIÓN MYPE-CRECEMAS	42
FIGURA 24. CREACIÓN DE CLASES JAVA PARA REPRESENTAR LA PÁGINA WEB DE LOGIN DE MYPE-CRECEMAS	42
FIGURA 25. DEFINICIÓN DEL CASO DE PRUEBA LOGIN EN EL ESCENARIO CUCUMBER	43
FIGURA 26. IMPLEMENTACIÓN DE LOS PASOS DEFINIDOS EN EL ESCENARIO DE CUCUMBER EN LOGINSTEPDEFINITION.JAVA	44
FIGURA 27. IMPLEMENTACIÓN DE LOS ELEMENTOS DE LA PÁGINA WEB A LA CLASE LOGINPAGE.JAVA	45
FIGURA 28. IMPLEMENTACIÓN DE LOS MÉTODOS EN LA CLASE LOGINSTEPS.JAVA	45
FIGURA 29. INTEGRACIÓN DE LOGINSTEPDEFINITIOS.JAVA Y LOGINSTEPS.JAVA	46
FIGURA 30. REPORTE DE EJECUCIÓN DEL CASO DE PRUEBA DE LOGIN	47
FIGURA 31. EJEMPLO DE EJECUCIÓN DE UN JOB DEL REPOSITORIO DE AUTOMATIZACIÓN	48

INDICE DE TABLAS

TABLA 1. EXPERIENCIA PROFESIONAL EN EMPRESA TEKTON LABS.	10
TABLA 2. EXPERIENCIA PROFESIONAL EN EMPRESA JALASOFT	11
TABLA 3. EXPERIENCIA PROFESIONAL EN EMPRESA GLOBANT PERÚ	11
TABLA 4. EXPERIENCIA PROFESIONAL EN EMPRESA CIGNIUM TECHNOLOGIES	12
TABLA 5. EXPERIENCIA PROFESIONAL EN EMPRESA BELATRIX SOFTWARE FACTORY	12
TABLA 6. FORMACIÓN ACADÉMICA PROFESIONAL	13
TABLA 7. ALCANCE DE LAS PRUEBAS PARA APLICATIVOS WEBS VS FRAMEWORK DE AUTOMATIZACIÓN.	33
TABLA 8. PRESUPUESTO ESTIMADO POR IMPLEMENTACIÓN DEL FRAMEWORK DE AUTOMATIZACIÓN	52
TABLA 9. TIEMPOS ESTIMADOS CONSIDERANDO LOS TIEMPOS OBTENIDOS EN EL REPORTE DE CUCUMBER (FIGURA 7)	53
TABLA 10. TABLA DE PORCENTAJES EN BASE A 4 HORAS DE TRABAJO	53
TABLA 11. ESTIMACIÓN EN TIEMPO PARA LA IMPLEMENTACIÓN DEL FRAMEWORK DE AUTOMATIZACIÓN	54
TABLA 12. COSTOS DE TECNOLOGÍA UTILIZADA PARA LA IMPLEMENTACIÓN DEL FRAMEWORK DE AUTOMATIZACIÓN.	54
TABLA 13. CANTIDAD DE CASOS DE PRUEBAS AUTOMATIZADOS	55

CAPITULO I TRAYECTORIA PROFESIONAL

1.1. PRESENTACIÓN PROFESIONAL

El autor tiene el grado de Bachiller y es egresado de la carrera de Ingeniería de Sistemas de la Universidad Nacional Mayor de San Marcos.

Inició practicas pre profesionales antes de concluir la carrera universitaria, en el rubro del desarrollo de software en empresas transnacionales.

Se considera como un profesional autodidacta con experiencia de 7 años en el área de calidad del desarrollo de software, iniciando con el rol de manual tester, en donde se resalta la experiencia de ejecución de pruebas para aplicativos móviles, web y servicios de Backend. Posteriormente, obtuvo el nuevo rol de ingeniero automatizador, enfocado a la automatización de pruebas web y de servicios de Backend, así como también configuración e instalación de herramientas de integración continua.

El liderazgo también es parte de la experiencia liderando equipos de hasta 5 integrantes, con actividades más resaltantes: seguimiento de actividades, distribución de tareas, evaluación trimestral y presentación de resultados.

1.2. EXPERIENCIA PROFESIONAL

Tabla 1. Experiencia Profesional en empresa Tekton Labs.

Mayo 2022 - Actualidad	ROL: Senior Software Test Engineer PROYECTO: Globality – Test Ops Team PRINCIPALES RESPONSABILIDADES: - Implementación y Extensión del framework de automatización Cypress. - Desarrollo de scripts automatizados para microservicios
------------------------	--

	- Desarrollo de scripts automatizados para aplicación web. - Manejo de herramientas como: Git, github, Cypress, Karate, AWS, Github Actions, JavaScript. - Metodología Ágil: SCRUM
--	--

Tabla 2. Experiencia Profesional en empresa Jalasoft

Octubre 2021 – Mayo 2022	ROL: Software Test Engineer PROYECTO: IPT Global – Automation Team PRINCIPALES RESPONSABILIDADES: - Desarrollo de scripts automatizados para aplicación desktop. - Manejo de herramientas como: Ranorex, Azure, Git, Tortoise. - Metodología Ágil: SCRUM
--------------------------	---

Tabla 3. Experiencia Profesional en empresa Globant Perú

Junio 2020 – Septiembre 2021	ROL: Semi-Senior Software Test Engineer PROYECTO: AEP Energy – NextStar 2.0 PRINCIPALES RESPONSABILIDADES: - Desarrollo de scripts automatizados para aplicación web. - Validaciones de queries en base de datos Oracle y Mysql. - Liderazgo del equipo de calidad. - Liderazgo del equipo de automatización. - Manejo de herramientas como: Java Customized Framework, Git, GitLab, Gitlab CI/CD, Mysql, Oracle, RestAssured, Selenium, Cucumber, Maven, TestNG, Java.
------------------------------	--

	- Metodología Ágil: SCRUM
--	---------------------------

Tabla 4. Experiencia Profesional en empresa Cignium Technologies

Mayo 2019 – Mayo 2020	ROL: Software Test Engineer PROYECTO: Tranzact PRINCIPALES RESPONSABILIDADES: - Desarrollo de scripts automatizados para aplicación web. - Integración de scripts con herramientas de integración usando TeamCity y Octopus Deploy. - Mantenimiento de scripts automatizados. - Guía de scripts automatizados para los equipos de calidad manual. - Manejo de herramientas como: .NET, C#, Gauge, TeamCity, Octopus Deploy, Git, Github y JMeter. - Metodología Ágil: KANBAN
-----------------------	--

Tabla 5. Experiencia Profesional en empresa Belatrix Software Factory

Diciembre 2015 – Abril 2019	ROL: Manual Tester / Automation Tester PROYECTO: FIS Global PRINCIPALES RESPONSABILIDADES: - Pruebas manuales para aplicaciones móviles - Pruebas manuales para aplicaciones web. - Pruebas manuales para microservicios usando postman. - Miembro activo en reuniones de SCRUM. - Primeros scripts usando Java, Cucumber, Maven, TestNG. Jenkins.
-----------------------------	---

	- Metodología Ágil: SCRUM
--	---------------------------

1.3. FORMACIÓN ACADÉMICA PROFESIONAL

Tabla 6. Formación académica profesional

FORMACIÓN RECIBIDA	INSTITUCIÓN QUE ACREDITA	DOCUMENTO DE ACREDITACIÓN	PERIODO
Grado Académico de Bachiller en Ingeniería de Sistemas	Universidad Nacional Mayor de San Marcos	Grado Académico Bachiller	2011 - 2016
Idioma Inglés	Centro Cultural Británico	Certificate Advanced Phase	2011 - 2015
ISTQB Foundation	ISQTB	16-CTFL-101539-03	2015

CAPITULO II CONTEXTO EN EL QUE SE

DESARROLLA LA EXPERIENCIA

2.1. EMPRESA – ACTIVIDAD QUE REALIZA

CreceMas es un startup cuya idea de negocio nació en el año 2018, pero el desarrollo de las aplicaciones empezó en el 2020, después de iniciada la pandemia del COVID-19. CreceMas busca posicionarse en el mercado de los microempresarios del Perú, promoviendo un ecosistema que permita la conexión entre mypes y usuarios finales, donde sus aplicaciones permiten realizar funcionalidades como: compra, facturación, control de stock, venta y distribución.

La actividad principal es la venta de software a medida en el sector retail de mypes del Perú, entre sus productos más destacados se encuentran los denominados: mype-crecemas (web), pedidos-crecemas (móvil) y distribuidor-crecemas (web).

2.2. VISIÓN

La empresa indica como visión el siguiente contenido:
Facilitamos la gestión de documentos electrónicos tributarios, contables y contractuales, a cada uno de nuestros clientes, a través de una solución confiable, segura y evolutiva, con nuestro conocimiento y experiencia, y con un equipo humano comprometido a entregar el mejor servicio.

2.3. MISIÓN

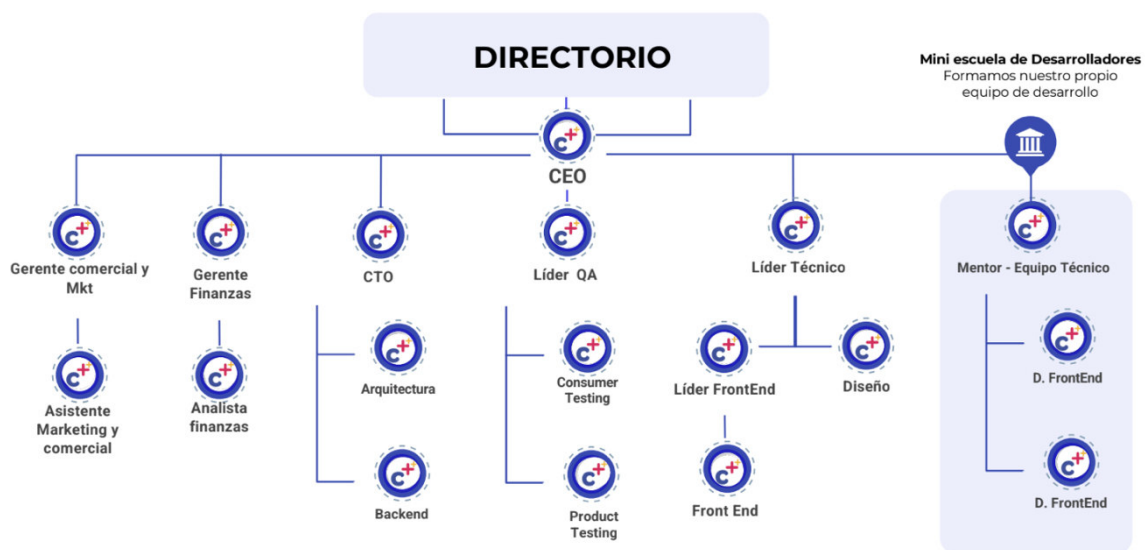
La misión de la empresa es representada por la siguiente descripción:
Proyectar la filosofía de evolución continua de Facele en todos nuestros servicios de gestión electrónica de documentos, construyendo soluciones ágiles para mercados

diversos, dinámicos y globales, en un ambiente de trabajo colaborativo que permita a las personas, que integren nuestros equipos, desarrollar sus talentos y capacidades.

2.4. ORGANIZACIÓN DE LA EMPRESA

La organización del startup se presenta en la siguiente figura:

Figura 1. Organigrama del startup CreceMas



Nota. La figura representa el organigrama de la empresa CreceMas. Información obtenida de la empresa CreceMas (2022).

2.5. AREA, CARGO Y FUNCIONES DESEMPEÑADAS

Área: Calidad de Software

Cargo: Líder QA

Funciones:

1. Supervisar y verificar la calidad del software versión para móviles y web.
2. Crear y ejecutar casos de prueba.
3. Reportar incidencias y seguimiento de los mismos.
4. Participar y guiar en las reuniones diarias.
5. Colaborar con el equipo de UX.
6. Benchmarking de software similares.

7. Formar parte del comité de propuestas e innovación.
8. Liderar y distribuir actividades en el equipo.
9. Implementación del framework Serenity BDD al proyecto.
10. Liderar el proyecto de automatización de pruebas de calidad.
11. Seguimiento y supervisión de avances del equipo.
12. Participar en reuniones de presentación del producto.
13. Entrevistar futuros integrantes en el área de calidad.
14. Trabajar con el área de innovación.

2.6. EXPERIENCIA PROFESIONAL REALIZADA EN LA

ORGANIZACIÓN

CreceMas en el año 2020, inicio la implementación de sus aplicaciones de software. Los presupuestos en proyectos de Startups no son altos, para lo cual se necesita que los integrantes del equipo realicen diversas actividades. El rol que se le asignó al autor es el de Líder de Calidad, que tiene como responsabilidades en el área de calidad de software como: creación y ejecución de casos de prueba para las aplicaciones web y móvil, participación en las reuniones de trabajo, de toma de decisión de mejoras de las aplicaciones, supervisión del personal a cargo y evaluar para futuros ingresos al equipo, mientras que en actividades asociadas a ventas se resalta actividades como: participar en reuniones de presentación de los productos con posibles clientes de inversión, evaluación del mercado competitivo, plantear mejoras y referente técnico de calidad para clientes.

A los 8 meses después de iniciado el proyecto, el autor lideró el proyecto de implementación del framework de automatización de ejecución pruebas de software para los aplicativos webs, los cuales tenían la mayor demanda del mercado. En esta etapa el autor realizó actividades como: el liderazgo del proyecto de automatización, seguimiento del trabajo realizado por los integrantes y distribución de asignaciones, presentación de avance y progreso a la mesa directiva y mantenimiento del framework.

CAPITULO III ACTIVIDADES DESARROLLADAS

3.1. SITUACIÓN PROBLEMÁTICA

3.1.1. DEFINICIÓN DEL PROBLEMA

La empresa CreceMas y el autor detectaron que el proceso de ejecución de pruebas de software de las aplicaciones webs demandaban cada vez más tiempo, ya que se implementaban nuevas funcionalidades y la revisión de las funcionalidades ya implementadas tomaban aún más tiempo, afectando directamente el tiempo de despliegue de una nueva versión de las aplicaciones.

Esto también limitaba los tipos de pruebas a realizar tales como regresión o humo, importantes para conocer la estabilidad de una aplicación en un momento dado. Por otro lado, la poca ejecución de pruebas de software afecta directamente en la cantidad de errores no encontrados, lo que conlleva a un software con bajo nivel de calidad. Finalmente, la empresa CreceMas es afectada, en presentaciones de inversión, cuando las aplicaciones mostradas no funcionan como se espera, generando pérdidas de oportunidades de negocio.

Se identificó la falta de una herramienta que permita la disminución en tiempos de ejecución de pruebas de software para las aplicaciones webs de la empresa CreceMas.

3.2. SOLUCIÓN

La implementación del framework Serenity BDD para la disminución en tiempos de ejecución de las pruebas de software para las aplicaciones webs.

3.2.1. OBJETIVOS

OBJETIVO GENERAL

La disminución del tiempo de ejecución de pruebas de software para las aplicaciones web de la empresa CreceMas utilizando el framework de automatización Serenity-BDD.

OBJETIVOS ESPECÍFICOS

1. Implementación del framework Serenity BDD y configuración para el soporte de ejecuciones de pruebas de software.
2. La automatización del 100% de casos de prueba que dan cobertura a flujos principales de las aplicaciones webs utilizando el framework de Serenity BDD.
3. Ejecución de las pruebas de software automatizadas para cada aplicativo web.

3.2.2. ALCANCE

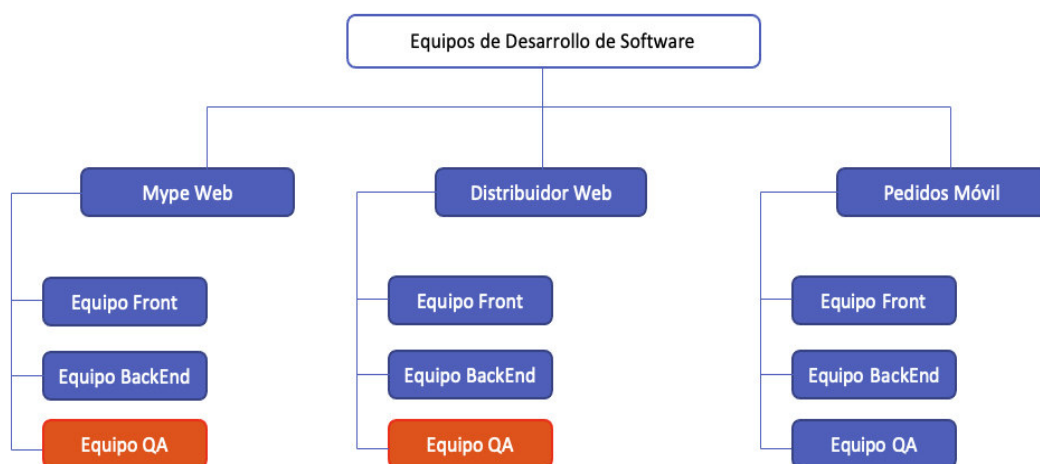
ALCANCE FUNCIONAL

El alcance del proyecto tiene como alcance la implementación del framework Serenity-BDD, creación y ejecución de casos de prueba automatizados de las aplicaciones web: mype-crecemas y distribuidor-crecemas.

ALCANCE ORGANIZACIONAL

Dirigido al equipo QA, del área de calidad de software del proyecto CreceMas para los equipos de mype-web y distribuidor-web.

Figura 2. Organigrama del equipo de desarrollo.



Nota. La figura representa la distribución de los equipos por producto en la empresa CreceMas. Elaboración propia.

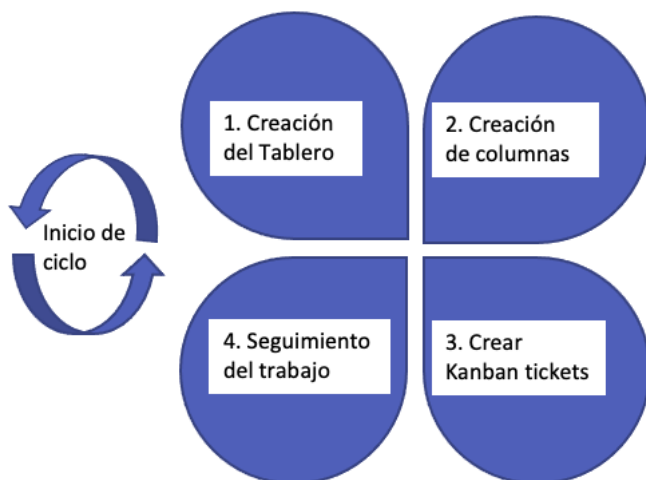
ALCANCE GEOGRÁFICO

El alcance del proyecto es dirigido a la empresa CreceMas y su área de desarrollo de software.

3.2.3. ETAPAS Y METODOLOGÍA

En la metodología, el proyecto decidió utilizar la metodología ágil llamada Kanban, esta metodología nos permite visualizar las actividades pendientes, en progreso y finalizadas. Para la administración y ejecución de las etapas mencionadas previamente, utilizaron la herramienta Trello.

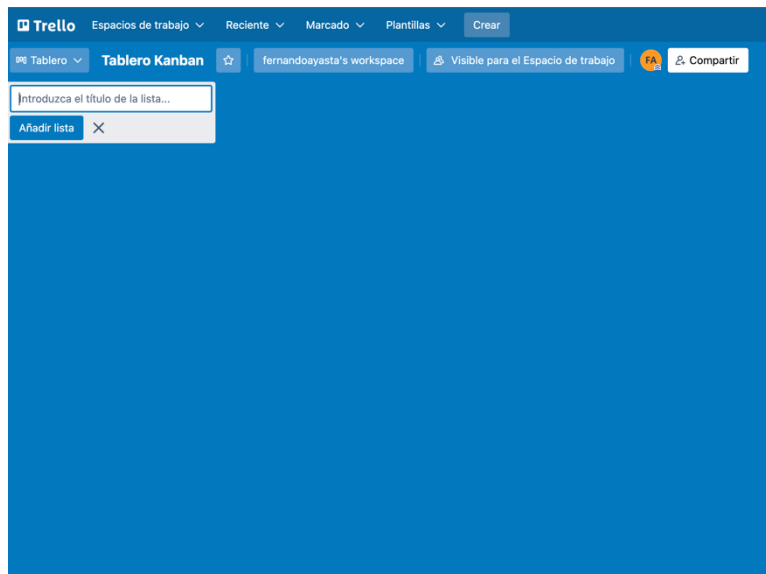
Figura 3. Proceso de implementación de la metodología Kanban



Nota. Elaboración propia.

1. Creación del tablero. La creación de un tablero permitirá visualizar todos las tareas y su estado actual. Para el proyecto se utilizó la herramienta Trello.

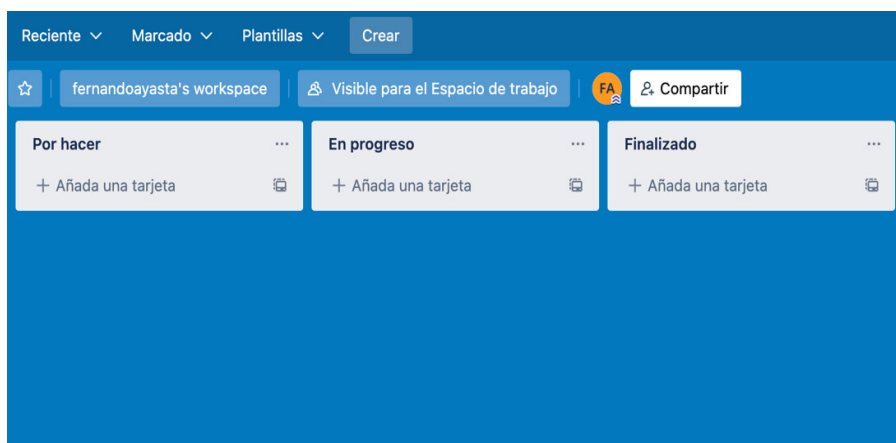
Figura 4. Tablero Kanban nuevo creado desde la herramienta Trello



Nota. Referencia de un tablero nuevo de la herramienta Trello. Elaboración propia.

2. Creación de columnas. Las columnas indicaran las etapas por las cuales las tareas han de pasar.

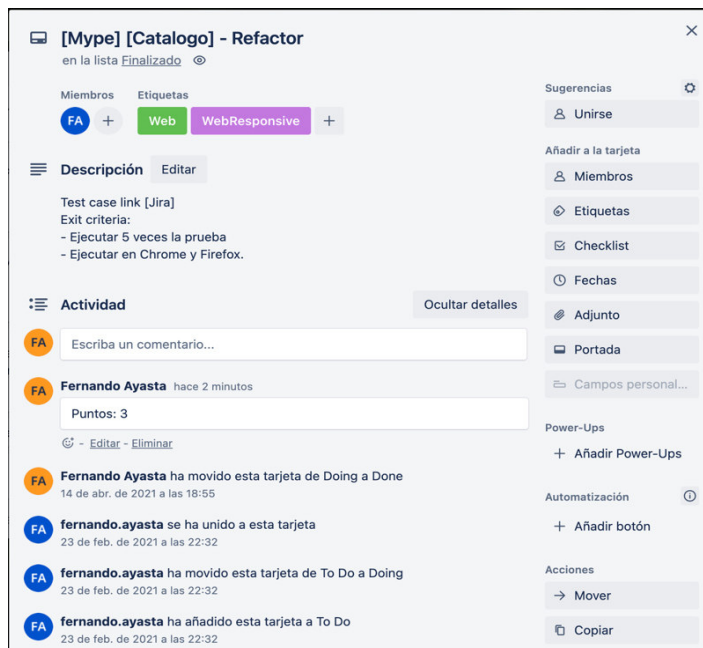
Figura 5. Tablero Kanban con definición de columnas.



Nota. Tablero Kanban con columnas definidas usando la herramienta Trello. Elaboración propia.

3. Crear Kanban Tickets. Los tickets representan las actividades por realizar, estos tickets irán pasando por las diferentes etapas (columnas) en el tablero. A su vez estos tickets, también contienen la descripción del tipo de actividad a realizar.

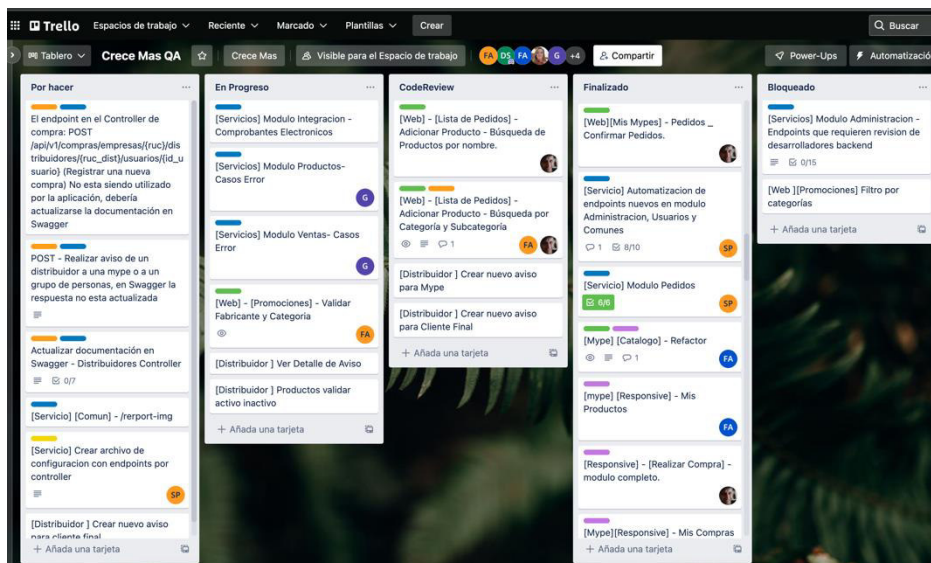
Figura 6. Kanban ticket del Proyecto CreceMas.



Nota. Ticket de trabajo para el proyecto de Automatización de la empresa CreceMas. Elaboración propia

4. Seguimiento del trabajo. Aquí se hace seguimiento que cada tarjeta este en el estado que debe estar. Logrando lo anterior, ayudará a visualizar el estado actual del proyecto.

Figura 7. Tablero de trabajo de CreceMas



Nota. Tablero Kanban del proyecto de Automatización de la empresa CreceMas (2021).

El proyecto adicional requirió de las siguientes actividades:

1. Reuniones Diarias. En estas reuniones, el equipo presentaba su avance personal. El reporte se planteó de la siguiente manera:
 - a. ¿Qué hizo ayer?
 - b. ¿Qué hará hoy?
 - c. ¿Algún impedimento de sus actividades?
2. Reunión de avance del equipo. El equipo decidió tener esta reunión semanalmente, en donde se presentaba los resultados de la semana trabajada y si alcanzaron las metas planteadas como equipo. El autor presentaba los resultados en un archivo Excel, indicando lo siguiente: fecha de revisión, cantidad de pruebas automatizadas finalizadas, el porcentaje de avance, objetivos del equipo. Así como se detalla en la siguiente figura.

Figura 8. Tabla de seguimiento de avance del desarrollo de pruebas automatizadas.

	Fecha de Revision	Web			Servicio		
		Cantidad de pruebas automatizadas	%Cobertura Automatizada	Objetivo del equipo (%)	Cantidad de pruebas automatizadas	Cobertura Automatizada	Objetivo del equipo (%)
Primera Revision	4 de Enero	16	29%	30%	27	12.98%	10%
Segunda Revision	15 de Enero	12	51%	50%	55	25.11%	25%
Tercera Revision	22 de Enero	11	67.27%	75%	30	13.70%	12.50%
Cuarta Revision	29 de Enero	16	100.00%	100%	107	48.86%	
Total		55			219	100.65%	

Nota. Elaboración propia.

3.2.4. FUNDAMENTOS UTILIZADOS

METODOLOGÍA ÁGIL

Las metodologías ágiles surgen ante la necesidad de mejorar el proceso de desarrollo de software permitiendo la flexibilidad del proyecto y la rápida respuesta del mismo ante los constantes cambios que el cliente requiera, proporcionándole así una entrega funcional temprana y continua que le de ventaja competitiva. (Maida & Pacienza, 2015)

KANBAN

Es un sistema de administración de flujos de trabajo y tareas. Kanban mediante el uso de tarjetas permite visualizar el estado actual de las tareas y del proyecto. Las tarjetas Kanban es donde se define el trabajo a realizar, estas irán cambiando dinámicamente en los diferentes estados dependiendo del progreso de la misma. Las columnas son definidas por el proyecto que lo implementa.

PRUEBAS DE SOFTWARE

La prueba de software es el proceso de evaluación y verificación de un producto o aplicación de software para saber si hace lo que se supone que debe hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento. Los Tipos de pruebas de software a tratar en el proyecto son:

1. Pruebas de Regresión.

Las pruebas de regresión son las pruebas que permiten saber si los nuevos cambios, nuevos requerimientos o correcciones afectan las funcionalidades anteriores. En este tipo de prueba se ejecuta todos los casos de pruebas que dan cobertura a todas las funcionalidades, con ello se comprueba que no se introdujeron nuevos errores en el sistema.

2. Pruebas de Humo.

Es el tipo de prueba que permite ejecutar casos de pruebas sin ser exhaustivos, los casos ejecutados son aquellos que verifican funcionalidad de alta criticidad de las aplicaciones, brindando así el resultado de cuan estable esta la aplicación en un momento dado.

FRAMEWORKS DE AUTOMATIZACIÓN

Los frameworks de automatización son marcos de trabajo que ofrecen una estructura sobre la cual se podrá elaborar un proyecto para un objetivo dado. Estos son tomados como plantillas, las cuales serán ajustadas, modificadas, extendidas de acuerdo a las necesidades del proyecto que la implementa.

Estos frameworks de automatización ofrecen funcionalidades, integraciones con plugin y librerías y proveen de configuraciones, lo que utilizarlas disminuye el tiempo de desarrollo. Existen varios frameworks de automatización en el mercado, pero en este proyecto nos enfocaremos en el framework Serenity-BDD (The Serenity BDD Book, n.d)

A continuación, se detallará las tecnologías, herramientas, metodologías requeridas para el mejor entendimiento del framework de automatización.

1. Behavior Drive Development (BDD)

BDD es una estrategia de desarrollo colaborativo, que plantea definir un lenguaje que pueda ser compartido tanto como personas del negocio y personas técnicas con el objetivo del entendimiento de los requerimientos por todo el equipo.

2. Serenity-BDD

Serenity es un framework open-source para la creación de pruebas de software. Esta escrita en el lenguaje de Java. Serenity ayuda a crear pruebas automatizadas de aceptación y regresión, así como también ofrece reportes ilustrados y narrativos que documentan lo que la aplicación bajo prueba hace. Para la implementación de estos reportes narrativos se utiliza la librería de Cucumber y el lenguaje Gherkin, detallados a continuación.

3. Cucumber

Cucumber es herramienta de pruebas de software open-source que soporta BDD, permite leer especificaciones ejecutables escritas en texto plano y valida que el software ejecuta lo que la especificación indica. Cucumber nos ayuda también a tener documentación viva, ya que indica las acciones a seguir en cada paso, con un lenguaje de alto nivel que puede ser entendido y comprendido por cualquier persona del negocio.

Figura 9. Ejemplo de una Especificación de un escenario.

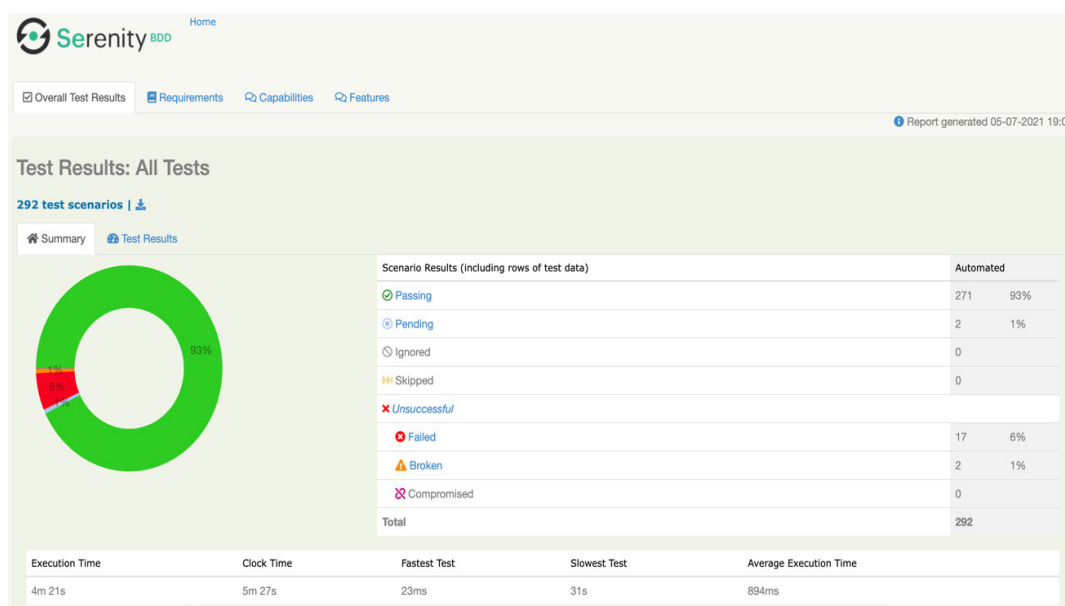
```
@regression @responsive-mobile @responsive-tablet
Escenario: Login a Crece+ utilizando "Correo Electronico y Clave" con datos válidos
Dado que el usuario ingresa a la pagina de login de Crece
Cuando se ingresa en el campo "Correo Electronico" el valor: '{usuario_email}'
Y se ingresa en el campo "Contraseña" el valor: '{usuario_pass}'
Y se presiona el boton Iniciar Sesión
Entonces en la pagina home de Crece+, es mostrado el nombre del usuario: '{usuario_nombre}'
```

Nota. Figura tomada del proyecto de automatización del framework Serenity-BDD, hace referencia al archivo login.feature.

4. Cucumber-Reports

Reportes son necesarias para saber si las pruebas ejecutadas pasaron o fallaron. Para ello el framework Serenity-BDD incluye los reportes de Cucumber. Estos reportes HTML son obtenidos después de una ejecución. Estos reportes contienen información detallada de los pasos ejecutados, tiempos de ejecución, pantallazos, etiquetas y mucho más.

Figura 10. Pantallazo de un reporte de Serenity BDD después de una ejecución de pruebas.



Nota. La figura hace referencia al reporte HTML, el cual muestra el resultado de las ejecuciones de las pruebas automatizadas.

5. Selenium

Selenium es una herramienta open-source utilizada para automatizar pruebas funcionales de software para aplicaciones webs. Se puede escribir tests utilizando diferentes lenguajes tales como: Python, C#, Ruby, Java, etc. Esta herramienta permite simular la interacción humano-software sobre las aplicaciones bajo prueba.

Serenity incluye esta herramienta en su configuración, lo que no se necesita algún proceso de instalación o configuración de la misma.

6. Maven

Es una herramienta que es usada para construir y administrar cualquier proyecto basado en el lenguaje Java. Maven está basado en el concepto de Project Object Model (POM), Maven puede administrar la construcción de un proyecto, reportes y documentación desde una pieza de información central.

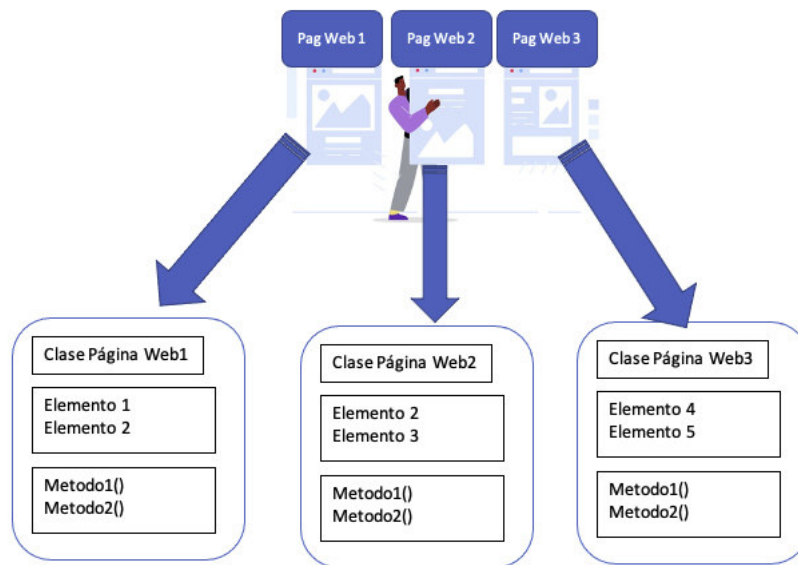
PATRON DE DISEÑO

El patrón es una descripción de un problema recurrente e indica como solucionarlo. Estas soluciones son aceptadas y su implementación varía dependiendo del lenguaje, framework, etc. Hay 3 categorías: Patrones creacionales, Patrones estructurales y Patrones de comportamiento.

1. Page Object Model

En Selenium, este patrón representa cada página web como clase y los elementos contenidos en la página web son clasificados como atributos de esa clase. Así como también, la interacción que realiza el usuario con esos elementos son los métodos de otra clase. Teniendo una distribución de los componentes del sistema web, ayuda a los trabajos de mantenimiento y desarrollo.

Figura 11. Representación del Modelo de Page Object.



Nota. Elaboración propia.

INTEGRACIÓN CONTINUA (CI)

La integración continua (CI por sus siglas en inglés) es la práctica de automatizar la integración de los cambios de código de varios contribuidores en un único proyecto de software. Es una de las principales prácticas recomendadas de DevOps, que permite a los desarrolladores fusionar con frecuencia los cambios de código en un repositorio central donde luego se ejecutan las compilaciones y pruebas. Las herramientas automatizadas sirven para verificar que el nuevo código es correcto antes de la integración.

Un sistema de control de versiones del código fuente es el punto clave del proceso de CI. El sistema de control de versiones también se complementa con otras comprobaciones como las pruebas automatizadas de calidad del código, las herramientas de revisión de estilo de sintaxis y mucho más. (Max Rehkoph, n.d.)

1. GITLAB

Es un repositorio git en la nube, que permite al equipo de desarrollo: almacenar, gestionar y dar el mantenimiento al código fuente.

2. GITLAB CI/CD

Herramienta de Integración Continua, la cual provee de un repositorio en la nube para almacenar los proyectos de software.

3. GIT

Sistema de control de versiones que nos permite tener un historial de los cambios sobre un archivo. Git permite comparar objetos, identificado nuevos cambios, modificaciones o eliminaciones, lo que fomenta el desarrollo, seguimiento del progreso a los desarrolladores.

3.2.5. IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS

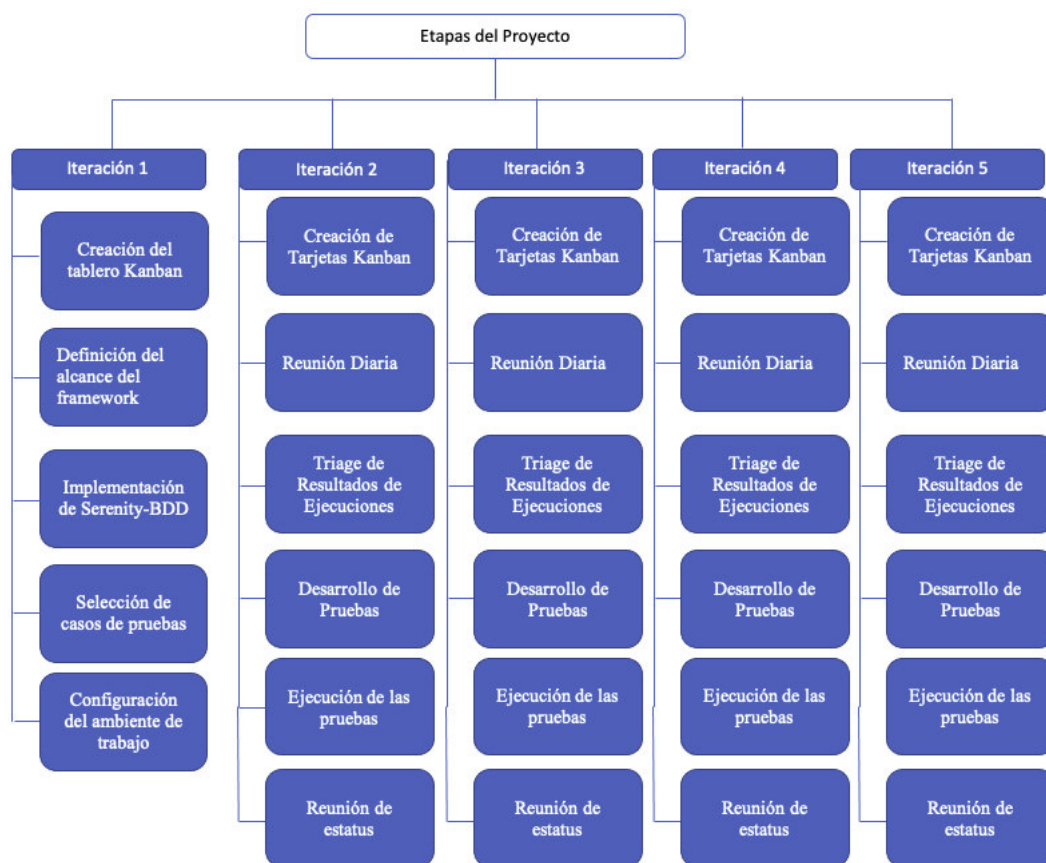
Y BUENAS PRÁCTICAS

El presente proyecto se realizó utilizando la metodología Kanban, el proyecto tuvo una estimación de 5 semanas (a cada semana llamaremos iteración), para implementación del framework Serenity-BDD y de los 39 casos de prueba y 219 endpoints para la aplicación mype-crecemas y distribuidor-crecemas.

En la primera iteración se consideró para el desarrollo de las actividades de: Definición del alcance del framework, Selección de casos de pruebas, Implementación de Serenity-BDD e Implementación de Kanban. se estimó para la implementación del Framework Serenity-BDD y la selección y estimación de los casos de pruebas a ser automatizados.

Las siguientes 4 iteraciones se utilizaron para el desarrollo de los casos de prueba con reuniones semanales de seguimiento con la mesa directiva. A continuación, se detalla las etapas del proyecto de implementación de software.

Figura 12. Proceso de implementación del framework Serenity-BDD en el proyecto CreceMas.



Nota. La figura representa los ciclos iterativos definidos para la implementación del framework Serenity-BDD y la automatización de los casos de prueba. Elaboración propia

ITERACIÓN 1

1. CREACIÓN EL TABLERO DE KANBAN

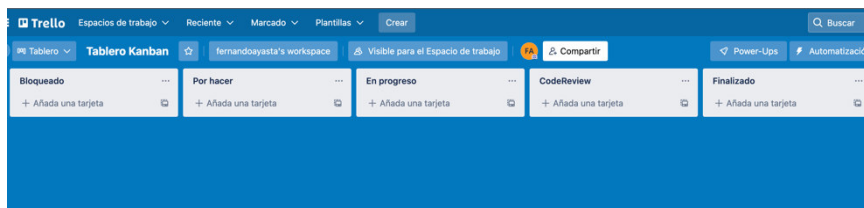
De acuerdo a lo requerido en la metodología Kanban, el tablero Kanban nos permitirá visualizar el trabajo a realizar y el estado de cada uno de las tarjetas creadas.

Esta etapa considera:

- Creación del tablero. El encargado de gestionar el proyecto, creará el tablero Kanban.
- Definición de las columnas. El encargado deberá crear las columnas por la cual las tarjetas Kanban irán pasando hasta ser culminadas. Estas etapas son definidas por el proyecto, en el caso del proyecto CreceMas, se plantearon las siguientes columnas: Bloqueado, Por hacer, En progreso, CodeReview, Finalizado. Donde:

- Bloqueado. Indica que existe un impedimento que no permita el avance de la tarea.
- Por hacer. Tareas que aún no han sido desarrolladas y están listas para ser trabajadas.
- En progreso. Significa que la tarjeta en esta columna, ya ha sido seleccionada por el integrante y está en pleno desarrollo.
- CodeReview. En esta etapa significa que la tarea necesita de revisión de código y se ha generado un Pull Request que está en revisión.
- Finalizado. Indica que tarea ha sido debidamente terminada logrando su objetivo definido.

Figura 13. Tablero Kanban del proyecto de implementación de framework Serenity-BDD.



Nota. Tablero de Kanban utilizando la herramienta Trello, en el cual se define los estados el cual cada ticket pasara durante su desarrollo. Elaboración propia.

2. DEFINICIÓN DEL ALCANCE DEL FRAMEWORK

En esta etapa se enfoca en definir el alcance para cada etapa definida para la implementación del framework de automatización. El objetivo es tener las configuraciones mínimas para el desarrollo de las pruebas. Las actividades que se tienen son las siguientes:

a) Definición del entorno de ejecución de pruebas

Aquí se detalla las configuraciones que el framework de automatización deberá soportar. Estas definiciones se basan en los requerimientos de las aplicaciones bajo pruebas. Para el proyecto de CreceMas, se definió el siguiente alcance:

Tabla 7. Alcance de las pruebas para aplicativos webs vs framework de automatización.

	Automatización
Sistemas Operativos	Windows 10
Navegadores	Chrome Firefox
Resolución de Pantalla	1920x1080

En un inicio el alcance de las ejecuciones automatizadas es limitada, decidiendo iniciar por lo que genere más valor al proyecto. Ahora, con esta definición se deberán crear tareas que verifiquen o implementen estas condiciones en el framework. Estas tareas deben hacerse con alta prioridad y antes de empezar a codificar los nuevos casos de prueba.

Figura 14. Creación de tarea en el tablero para soportar diversos navegadores.



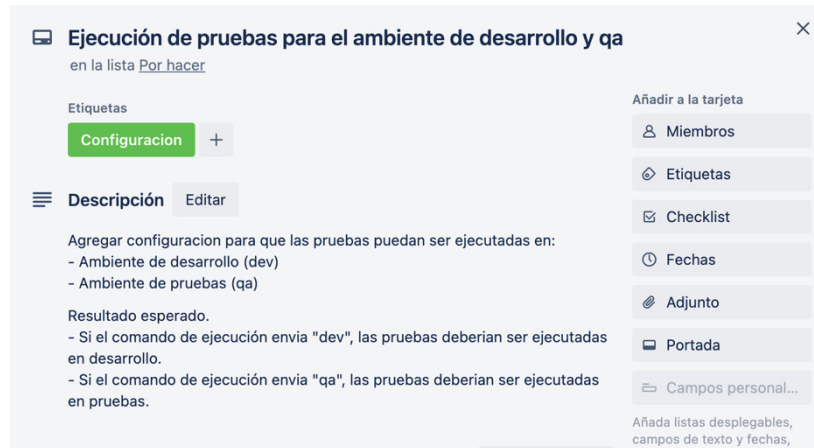
Nota. Creación de tarjeta en el tablero Kanban que ayudará a hacer seguimiento la tarea definida.

b) Definición de los ambientes de ejecución de pruebas

En todo proyecto se requiere definir cual o cuales será los ambientes contra los cuales serán ejecutadas las pruebas automatizadas. En el proyecto de CreceMas se tiene 3 ambientes: desarrollo, qa y producción; de los cuales se definió que la

ejecución se pueda dar en los 2 ambientes (desarrollo y qa).

Figura 15. Creación de tarea en el tablero para ejecutar pruebas en los ambientes indicados.



Nota. Creación de tarjeta en el tablero Kanban que ayudará a hacer seguimiento la tarea definida.

- c) Definición de las configuraciones de integración con la herramienta de Integración Continua.

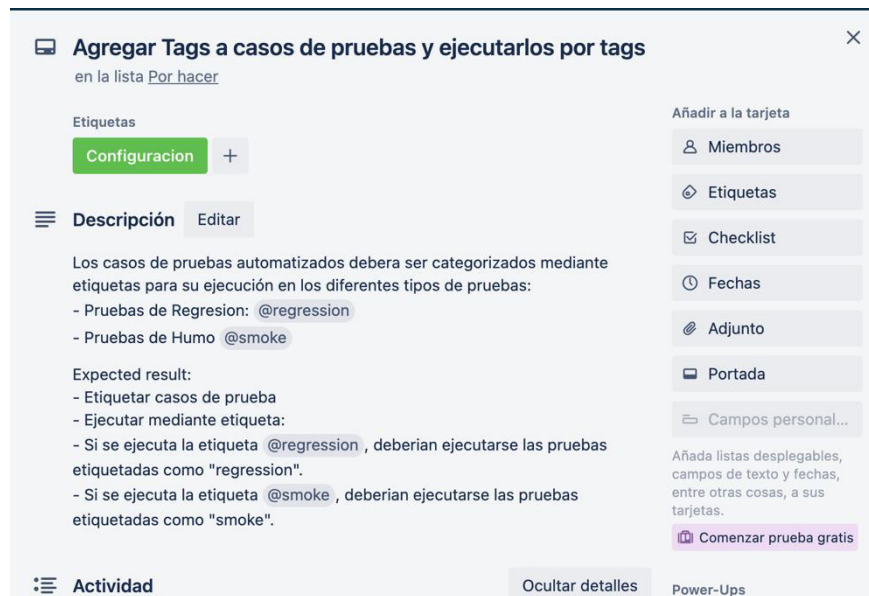
La selección de la herramienta de CI depende de varios factores, como: compartir la herramienta de CI que utiliza el equipo de desarrollo, seleccionar una herramienta de CI que ofrece mayores beneficios para la tecnología utilizada en el framework, sugerir una herramienta por la experiencia de uso, etc.

Considerando que el equipo de desarrollo ya tenía un repositorio privado, convenia utilizar el mismo repositorio para el proyecto de automatización, hacer ello no incrementaba el costo mensual de pago. Por lo tanto, la herramienta seleccionada es GitLab. Para el proceso de creación del nuevo repositorio seguir Anexo 3.2.

- d) Definición de los tipos de pruebas

El proyecto definió 2 tipos de pruebas: pruebas de regresión y pruebas de humo o smoke. Este tipo de pruebas ayudan a saber la estabilidad de la aplicación bajo prueba. Para los casos de pruebas que se categorizaban como pruebas de regresión, se les asignaba una etiqueta regression, mientras que para las pruebas de humo su etiqueta es smoke.

Figura 16. Creación de tarea en el tablero para ejecutar mediante etiquetas.



Nota. Creación de tarjeta en el tablero Kanban que ayudará a hacer seguimiento la tarea definida.

Figura 17. Ejemplo de etiquetas en las pruebas automatizadas ya implementadas.

```
@plataforma:web-mype @regression @smoke
Esquema del escenario: La mype realiza una búsqueda de ordenes de compra
Dado que el usuario ingresa a la pagina de login de Crece
Cuando se realiza el login con los datos Usuario : '{usuario_correo}' y Contraseña: '{usuario_password}'
Entonces se dirige a la pantalla de Mis Compras
Y realizo la búsqueda de una orden de compra por nombre "<OrdenCompraID>"
Y valido que la orden de compra "<OrdenCompraID>" es mostrada
Y el usuario sale de la aplicacion mype
```

Nota. La figura señala un ejemplo de etiquetas definidas y añadidas a casos de prueba automatizados. Estos son definidos por el proyecto.

e) Proceso de Reporte de incidencias

Una consecuencia de las ejecuciones de casos de pruebas, ya sea manual o automatizada, es de encontrar bugs. Estos errores deben ser apropiadamente reportados. El autor planteó agregar una etiqueta especial para hacer referencia al proyecto de automatización, ya que esta ayudará en las métricas de cuantos errores fueron detectados por las pruebas automatizadas. En Jira que es la herramienta de administración de tareas, la etiqueta definida para cuando una incidencia es

encontrada mediante una prueba automatizada es `foundByAutomation`.

f) Definición de DONE

Aquí se define, cuando se considera que el proceso de desarrollar un caso de prueba ha sido completado correctamente. Para el proyecto de CreceMas, el autor planteó para dar por finalizada la tarea se debe considerar:

1. Ejecución de la prueba creada 5 veces seguidas.
2. Ejecución de la rama de trabajo usando la herramienta de CI.
3. Agregar etiquetas por modulo, prueba, tipo de prueba (regression o smoke).
4. Revisión del líder técnico (Code Review).
5. Código reutilizable.

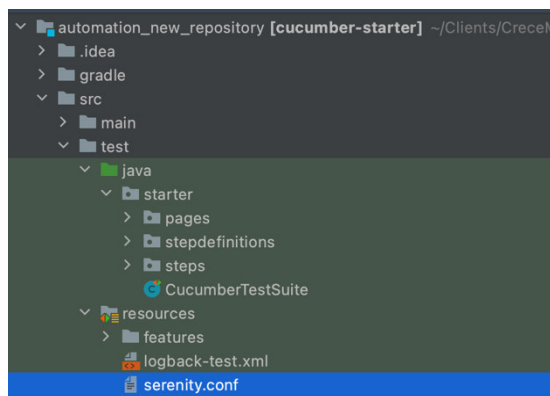
3. IMPLEMENTACIÓN DE SERENITY-BDD

En esta sub etapa tiene las siguientes actividades

- a) Implementación del framework Serenity-BDD en un repositorio privado del proyecto (Véase Anexo 3).
- b) Organización de carpetas en el framework.

Siguiendo la estructura del framework y el Patrón de Object Model, se necesita crear 3 carpetas (page, steps, stepdefinitions).

Figura 18. Estructura del framework aplicando Page Object Model

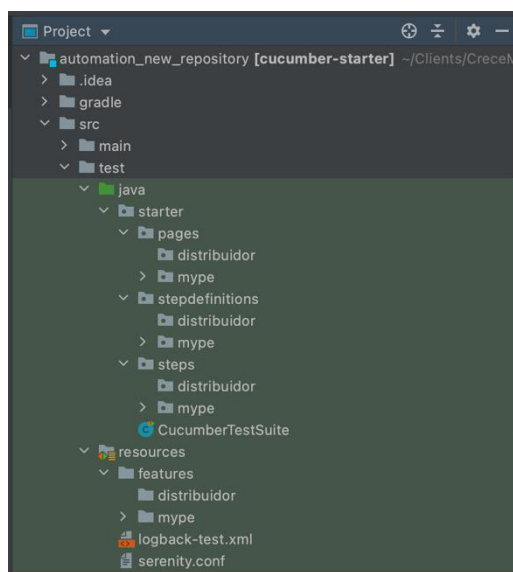


Nota. Figura resalta la estructura de carpetas (pages, stepdefinitions y steps) creadas para organizar las clases java que serán creadas, las cuales seguirán el patrón de Page Object Model.

La carpeta ‘page’, almacenara las clases, siguiendo el patrón de diseño de Page Object, cada clase es la representación de una página web. La carpeta ‘steps’, almacenara las clases, las cuales tienen los métodos para simular la interacción humano-navegador. Y por último, el folder stepdefinitions, es quien implementara los pasos definidos en el archivo .feature.

Para este proyecto se considerará una distribución similar para las aplicaciones de mype-crecemas y distribuidor-crecemas. Tendríamos la siguiente estructura:

Figura 19. Estructura de carpetas del framework de automatización



Esta estructura guiará las pautas para la creación de carpetas, clases y adicionales al framework.

4. SELECCIÓN DE CASOS DE PRUEBA

Para que un framework de automatización retorne el mayor beneficio al proyecto, el equipo se debería enfocar en las funcionalidades que cumplan ciertos requisitos. Para la selección de los casos de pruebas a ser automatizados, se basó en los siguientes criterios.

- Que den cobertura a flujos importantes del negocio.
- Repetitivos y de alto esfuerzo humano.
- Que den cobertura a funcionalidades interconectadas o que son afectadas por otros módulos.

d) Que den cobertura a flujos que no se modifican.

Para un seguimiento del proceso, el autor, una vez seleccionados los casos de prueba, almacenaba estos casos en un Excel para su posterior estimación en horas y su creación en el tablero Kanban. También se clasificaban cuales casos de prueba pertenecen a la iteración 2,3,4 y 5. Este cuadro era actualizado cada día hasta que termine el proceso. Este Excel también formaba parte de la reunión semanal para mostrar el avance del proyecto.

Figura 20. Distribución de los casos de prueba y sus estimaciones para la aplicación Mype-CreceMas.

Mype	Modulo	Funcionalidad	Estimacion	Target	Browser	Assignee	Status
7 Realizar Venta		1. Venta De Producto	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		2. Busqueda de 1 Producto	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		3. Busqueda de Varios Productos	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		4. Venta de Producto en Cantidad	5 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		5. Busqueda por Filtros	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		6. Venta de producto con Descuento	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		7. Venta Producto por Agrupacion	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
9 Realizar Compra		1. Compra De Producto	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		2. Busqueda de 1 Producto	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		3. Compra y Validacion en Distribuidor App	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		4. Compra y Validacion en Mis Compras	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		6. Compra de Producto en Cantidad	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		7. Compra de producto con Descuento	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		8. Compra de producto con Bonificacion	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		9. Busqueda por Categoria/SubCategoria	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
3 Mis Compras		1. Orden de Compra - Ver detalle de pedido	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		2. Orden de Compra - Busqueda por orden de compra	2 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		3. Orden de Compra - Filtro por Estado (Todos)	4 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		4. Orden de Compra - Filtro por Fecha	2 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	ToDo
10 Mis Productos		1. Todos - Busqueda de Producto	2 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		2. Chatbot - Busqueda de Producto	2 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		3. Categorías - Visualizacion de Categorías	3 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		4. Agrupaciones - Validar Agrupacion Existente	3 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		5. Todos - Añadir Nuevo Producto - Create por Mi	3 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
		6. Todos - Añadir Nuevo Producto - Catalogo	3 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done
Considerar el mensaje Not Found		7. Categoría - Buscar producto Activo en una Categoría		Browser Full Screen	Chrome/Firefox	Leonardo	Done
Considerar el mensaje Not Found		8. Categoría - Buscar producto inactivo en una Categoría		Browser Full Screen	Chrome/Firefox	Leonardo	Done
		9. Agrupaciones - Buscar producto activo de una agrupacion		Browser Full Screen	Chrome/Firefox	Leonardo	Done
		10. Agrupaciones - Buscar producto inactivo de una agrupacion		Browser Full Screen	Chrome/Firefox	Leonardo	Done
		11. Chatbot - Validar productos del chatbot tenga activo el switch de Chatbot	TBD	Browser Full Screen	Chrome/Firefox	Leonardo	ToDo
2 Mi Empresa		1. Crear Usuario Vendedor	5 horas	Browser Full Screen	Chrome/Firefox	Leonardo	Done
		2. Crear Usuario Propietario	5 horas	Browser Full Screen	Chrome/Firefox	Leonardo	Done
1 Cerrar Sesión		1. Cerrar sesion	2 horas	Browser Full Screen	Chrome/Firefox	Fernando Ayasta	Done

Nota. Figura representa la distribución de tareas, estimaciones, persona asignada y su estado actual del proyecto de automatización usando Serenity BDD.

Figura 21. Distribución de los casos de prueba de servicios y sus estimaciones para la aplicación Mype-CreceMas.

Trellid	Module	Endpoint	Estimacion	Assignee	Status	Observaciones	Status	Observaciones
IdTrello: 45	Comunes	Post /api/v1/crecemas/templates/repote-img		Gustavo Buhr	ToDo	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/calificaciones		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/lestados		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/filtros		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/medidas		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/monedas		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/peru/ubigeo(codigo_departamento)/(codigo_provincias)/distritos		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/peru/ubigeo(codigo_departamento)/provincias		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/peru/ubigeo/departamentos		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/productos/categorias		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/hubros		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/templates/crecemas-productos-xls		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/templates/distribuidores-productos-xls		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 /api/v1/crecemas/tipos-pago		Gustavo Buhr	Done		ToDo	
IdTrello: 5	Usuarios	8 Get		Gustavo Buhr	Done	Solo Casos 200	ToDo	Casos Negativos
		1 Post		Gustavo Buhr	Done	Solo Casos 200	ToDo	Casos Negativos
		2 Put		Gustavo Buhr	Done	Solo Casos 200	ToDo	Casos Negativos
		2 Delete		Gustavo Buhr	Done	Solo Casos 200	ToDo	Casos Negativos
		2 Patch		Gustavo Buhr	Done	Solo Casos 200	ToDo	Casos Negativos
		28 Get		Santiago	Done	Solo Casos 200	ToDo	
IdTrello: 15	Distribuidores (Parte I)	3 Post		Santiago	Done	Solo Casos 200	ToDo	
		1 Put		Santiago	Done	Solo Casos 200	ToDo	
IdTrello: 24	Distribuidores (Parte II)	4 Get		Santiago	Done	Solo Casos 200	ToDo	
		9 Patch		Santiago	Done	Solo Casos 200	ToDo	
		10 Post		Santiago	Done	Solo Casos 200	ToDo	
IdTrello: 6	Productos	9 Get		Santiago	Done	Solo Casos 200	ToDo	
		2 Post		Santiago	Done	Solo Casos 200	ToDo	
		1 Patch		Santiago	Done	Solo Casos 200	ToDo	
IdTrello: 12	Ventas (*)	19 Get		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		7 Post		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		2 Put		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		2 Patch		Gustavo Buhr	Done	Solo Casos 200	ToDo	
IdTrello: 7	Compras (*)	9 Get		Santiago	Done	Solo Casos 200	ToDo	
		3 Post		Santiago	Done	Solo Casos 200	ToDo	
IdTrello: 48	Administracion	3 Get		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		14 Post		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		7 Patch		Gustavo Buhr	Done	Solo Casos 200	ToDo	
IdTrello: 63	Pedidos	21 Get		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		3 Post		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		2 Delete		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		4 Patch		Gustavo Buhr	Done	Solo Casos 200	ToDo	
		1 Put		Gustavo Buhr	Done	Solo Casos 200	ToDo	

Nota. Figura representa la distribución de tareas de las pruebas automatizadas que dan cobertura a los servicios utilizados por las aplicaciones web.

5. CONFIGURACIÓN DEL AMBIENTE DE TRABAJO

Para poder desarrollar los casos de pruebas utilizando Serenity se debe configurar el ambiente de trabajo local, para ello seguir los anexos 1, 2, 4 y 5.

ITERACIÓN 2

La iteración 2 tiene una duración de 1 semana y contiene las siguientes etapas.

1. CREACIÓN DE TARJETAS KANBAN

De acuerdo a lo requerido en la metodología Kanban, se deberá crear las tarjetas, las cuales representan el trabajo modular por realizar.

Al ser creada cada tarjeta, esta contiene la descripción y un link al caso de prueba almacenado en la herramienta Jira, el nombre de quien esta asignado y etiquetas para reconocer si es un caso de prueba funcional o de servicios, así como también etiquetas a que tipo de prueba pertenecen.

Figura 22. Ejemplo de tarjeta creada en el proyecto CreceMas



Nota. Figura representa un ticket para el proyecto de automatización de CreceMas. En este se vincula el caso de prueba manual para ser automatizado. Adicional de información como: aplicación y etiquetas para añadir.

Este proceso debe ser repetido para cada una de las pruebas seleccionadas para la iteración 2.

2. REUNIÓN DIARIA

Reunión diaria de estado personal para conocer el avance de las actividades asignadas. Ayuda a evitar problemas de incumplimiento y un mejor seguimiento de asignaciones.

3. TRIAGE DE RESULTADOS DE EJECUCIONES

El equipo antes de iniciar cada actividad diaria, debía verificar los resultados de las pruebas automatizadas. Si e caso hubiera una ejecución fallida, el tester debía averiguar si era un falso positivo o un error. Y realizar el proceso adecuado dependiendo del resultado. Las tareas de mantenimiento también son consideradas como parte del desarrollo de pruebas automatizadas.

4. DESARROLLO DE PRUEBAS SELECCIONADAS

En esta sub etapa se implementa los casos de prueba seleccionados para la iteración 2 considerando las horas estimadas.

Para esta primera iteración se definirá el proceso de desarrollo del caso de prueba Login de la aplicación mype-crecemas.

Figura 23. Pantalla de la aplicación mype-crecemas



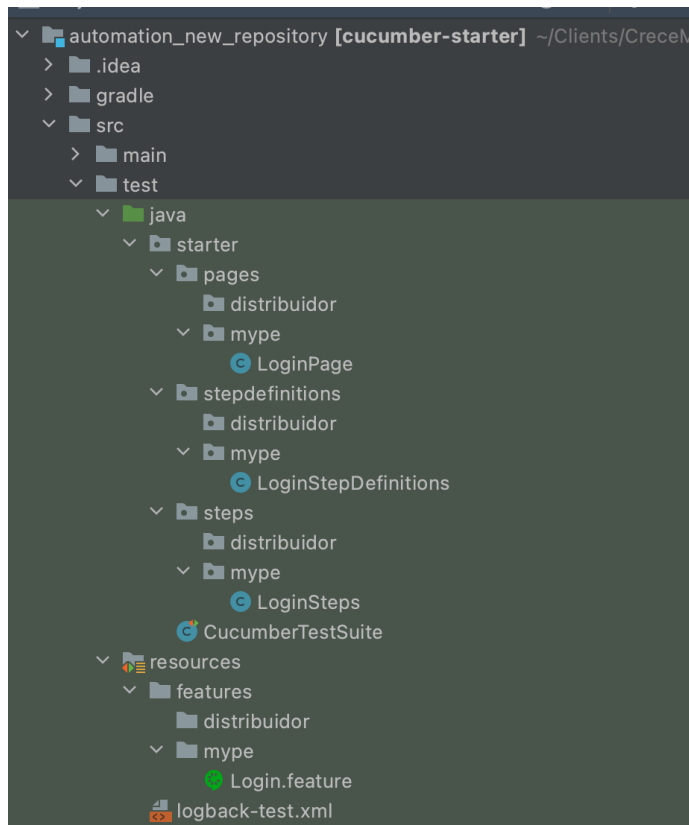
Nota. Figura representa la pantalla de autenticación para el aplicativo mype-crecemas.

Proceso para desarrollar un caso de prueba:

1. Creación de las clases java.

En pasos anteriores ya se creó la estructura de carpetas siguiendo el modelo de Page Object y del framework Serenity.

Figura 24. Creación de Clases Java para representar la página web de Login de mype-crecemas



Nota. Creación de las clases java para la pantalla de Autenticación (Login en inglés).

2. Definición de los pasos del caso de prueba

Como se está utilizando BDD, se define primero lo que la prueba hará y luego su implementación. Para la definición de los pasos, se utiliza el lenguaje Gherkin, estas definiciones se guardan en el archivo Login.feature. Mientras que la implementación de estos pasos será desarrollada en las clases java LoginStepDefinitions.java.

Figura 25. Definición del caso de prueba Login en el escenario Cucumber

```

# language: es
@plataforma:web-mype
@featureWeb:Login
@modulo:mype
Característica: Mype: Login en la pagina de Mype
  En orden de validar que solo clientes de Crece+ pueden ingresar a la pagina

@regression @smoke
Escenario: Login a Crece+ utilizando "Correo Electronico y Clave" con datos válidos
  Dado que el usuario ingresa a la pagina de login de Crece
  Cuando se ingresa en el campo "Correo Electronico" el valor: '{email}'
  Y se ingresa en el campo "Contraseña" el valor: '{contrasena}'
  Y se presiona el boton Iniciar Sesión
  Entonces en la pagina home de Crece+, es mostrado el nombre del usuario: '{usuario_nombre}'
  Y el usuario sale de la aplicacion mype

```

Nota. Implementación del caso de prueba de autenticación usando lenguaje Gherkin.

Figura 26. Implementación de los pasos definidos en el escenario de Cucumber en LoginStepDefinition.java

```

package starter.stepdefinitions.mype;

import io.cucumber.java.es.Cuando;
import io.cucumber.java.es.Dado;
import io.cucumber.java.es.Entonces;
import io.cucumber.java.es.Y;

public class LoginStepDefinitions {
    @Dado("que el usuario ingresa a la pagina de login de Crece")
    public void queElUsuarioIngresaALaPaginaDeLoginDeCrece() {
    }

    @Cuando("se ingresa en el campo {string} el valor: {string}")
    public void seIngresaEnElCampoElValorEmail(String arg0) {
    }

    @Y("se presiona el boton Iniciar Sesión")
    public void sePresionaElBotonIniciarSesion() {
    }

    @Entonces("en la pagina home de Crece+, es mostrado el nombre del usuario: {string}")
    public void enLaPaginaHomeDeCreceEsMostradoElNombreDelUsuarioUsuario_nombre() {
    }

    @Y("el usuario sale de la aplicacion mype")
    public void elUsuarioSaleDeLaAplicacionMype() {
    }
}

```

Nota. Implementación de los pasos definidos en lenguaje Gherkin en métodos java.

3. Identificar los elementos de la página web.

La página web de Login tiene 3 elementos importantes: el campo de email, el campo de contraseña y el botón de Iniciar Sesión. Estos 3 elementos serán representados como atributos de la clase LoginPage.java.

Figura 27. Implementación de los elementos de la página web a la clase LoginPage.java

```

package starter.pages.mype;

import net.serenitybdd.core.annotations.findby.FindBy;
import net.serenitybdd.core.pages.WebElementFacade;

public class LoginPage {

    @FindBy(name = "email")
    private WebElementFacade emailTextField;

    @FindBy(name = "password")
    private WebElementFacade passTextField;

    @FindBy(css = "button.button.button--xl.button--accent")
    private WebElementFacade loginButton;

    public WebElementFacade getEmailTextField() {
        return emailTextField;
    }

    public WebElementFacade getPassTextField() {
        return passTextField;
    }

    public WebElementFacade getLoginButton() {
        return loginButton;
    }
}

```

Nota. Clase java LoginPage, la cual almacena la representación de los elementos HTML de la pantalla de Autenticación de la aplicación mype-crecemas.

4. Creación de métodos de los pasos.

Los métodos son la simulación que realiza el usuario con la aplicación. Para el ejemplo de la funcionalidad de Login, el usuario debe completar los campos obligatorios de email y contraseña y luego clic en el botón de Iniciar Sesión. Conociendo lo que debe hacer el usuario, se procede a crear los métodos en la clase LoginSteps.java.

Figura 28. Implementación de los métodos en la clase LoginSteps.java

```

package starter.steps.mype;

import net.thucydides.core.annotations.Step;
import starter.pages.mype.LoginPage;

public class LoginSteps {

    private LoginPage loginPage;

    @Step
    public void openLoginPage(){
        loginPage.open();
    }

    @Step
    public void completeEmailField(String email){
        loginPage.getEmailTextField().sendKeys(email);
    }

    @Step
    public void completePassField(String password){
        loginPage.getPassTextField().sendKeys(password);
    }

    @Step
    public void clickIniciarSesion(){
        loginPage.getLoginButton();
    }

}

```

Nota. Clase java LoginSteps almacena los métodos, los cuales son las representaciones de las simulaciones del usuario con los elementos HTML.

5. Integración de los métodos con las implementaciones de los StepDefinitions.

Lo que se busca es la integración de los pasos definidos en el paso b) con los métodos del paso d).

Figura 29. Integración de LoginStepDefinitios.java y LoginSteps.java

```

import starter.steps.mype.LoginSteps;

public class LoginStepDefinitions {

    @Steps
    LoginSteps loginSteps;

    @Dado("que el usuario ingresa a la pagina de login de Crece")
    public void queElUsuarioIngresaALaPaginaDeLoginDeCrece() {
        loginSteps.openLoginPage();
    }

    @Cuando("se ingresa en el campo {string} el valor: {string}")
    public void seIngresaEnElCampoElValorEmail(String email) {
        loginSteps.completeEmailField(email);
    }

    @Y("se presiona el boton Iniciar Sesion")
    public void sePresionaElBotonIniciarSesion() {
        loginSteps.clickIniciarSesion();
    }

    @Entonces("en la pagina home de Crece+, es mostrado el nombre del usuario: {string}")
    public void enLaPaginaHomeDeCreceEsMostradoElNombreDelUsuarioUsuario_nombre(String password) {
        loginSteps.completePassField(password);
    }

}

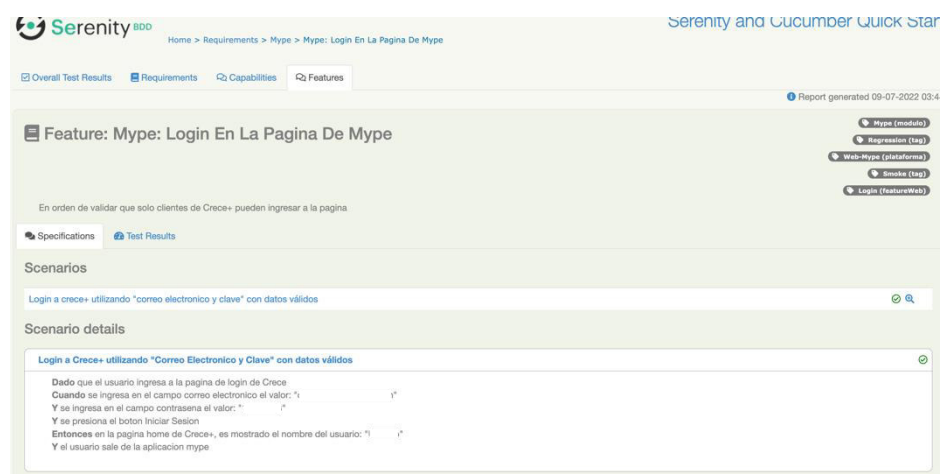
```

Nota. Clase LoginStepDefinitions es la conexión entre los pasos definidos en lenguaje Gherkin con los métodos creados en la clase LoginSteps.

6. Ejecutar la prueba.

Para la ejecución de la prueba, ir al Anexo 5. Una vez terminada la prueba, se podrá obtener el reporte de la ejecución en la ruta /target/site/Serenity/index.html. Con los pasos definidos en LoginStepDefinitions.java

Figura 30. Reporte de ejecución del caso de prueba de Login



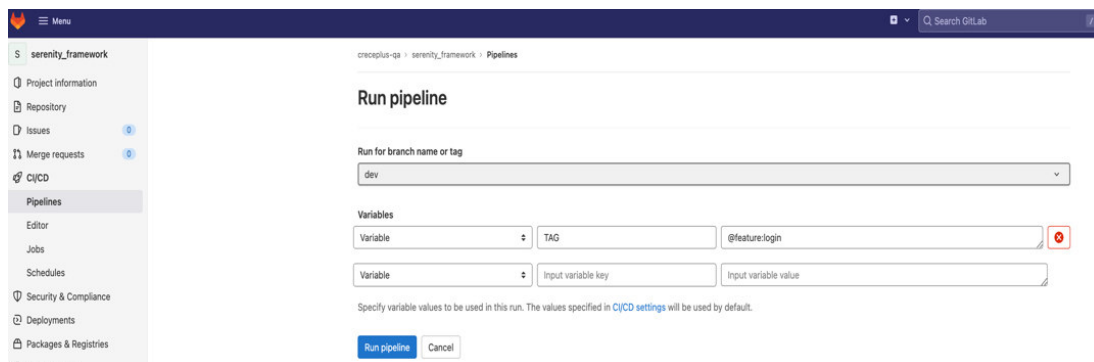
Nota. La figura representa el reporte HTML obtenido de la ejecución de los pasos anteriores y como los pasos escritos en el la figura 25 son mostrados.

Este flujo es recurrente para todos los casos de prueba seleccionados para las iteraciones 2, 3, 4 y 5.

5. EJECUCIÓN DE LA SUITE DE PRUEBAS

Al final de cada día se debía ejecutar la suite de pruebas al final del día utilizando GitLab CI/CD, con el objetivo de que cada inicio del día, el equipo sepa la estabilidad del software. Si en caso hubiera fallas, en el proceso de Triage evaluará e identificará la causa de la falla.

Figura 31. Ejemplo de ejecución de un job del repositorio de Automatización



Nota. Representación de la herramienta de GitLab CI/CD para la ejecución de un pipeline.

Cuando la ejecución termina, se guardarán el reporte de la ejecución. Y podrán servir al equipo para conocer la estabilidad de la aplicación, tiempos de ejecución, si las pruebas automatizadas están bien desarrolladas, etc.

6. REUNIÓN DE ESTATUS

Al final de cada iteración se tiene una reunión en la cual se indica:

- a) Si se logró la meta definida, es decir automatizar todos los casos definidos para la iteración.
- b) En caso de que no se logre a la meta, se renegociara las metas para la siguiente iteración.

ITERACIÓN 3

La iteración 3 tiene una duración de 1 semana y contiene las siguientes etapas.

1. CREACIÓN DE TARJETAS KANBAN

Se crearán las tarjetas de los casos de prueba seleccionados para la Iteración 3.

2. REUNIÓN DIARIA

Reunión diaria de estado personal para conocer el avance de las actividades asignadas. Ayuda a evitar problemas de incumplimiento y un mejor seguimiento de asignaciones.

3. TRIAGE DE RESULTADOS DE EJECUCIONES

El equipo antes de iniciar cada actividad diaria, debía verificar los resultados de las pruebas automatizadas. Si e caso hubiera una ejecución fallida, el tester debía averiguar si era un falso positivo o un bug. Y realizar el proceso adecuado dependiendo del resultado. Las tareas de mantenimiento también son consideradas como parte del desarrollo de pruebas automatizadas.

4. DESARROLLO DE PRUEBAS SELECCIONADAS

En esta sub etapa se implementa los casos de prueba seleccionados para la iteración 3 considerando las horas estimadas. También se podría considerar trabajo no realizado de la iteración anterior.

5. EJECUCIÓN DE LA SUITE DE PRUEBAS

Al final de cada día se debía ejecutar la suite de pruebas al final del día utilizado la herramienta de CI, con el objetivo de que cada inicio del día, el equipo sepa la estabilidad del software. Si en caso hubiera fallas, en el proceso de Triage evaluará e identificará la causa de la falla.

6. REUNIÓN DE ESTATUS

Al final de cada iteración se tiene una reunión en la cual se indica:

- a) Si se logró la meta definida, es decir automatizar todos los casos definidos para la iteración.
- b) En caso de que no se logre a la meta, se renegociara las metas para la siguiente iteración.

ITERACIÓN 4

La iteración 4 tiene una duración de 1 semana y contiene las siguientes etapas.

1. CREACIÓN DE TARJETAS KANBAN

Se crearán las tarjetas de los casos de prueba seleccionados para la Iteración 4.

2. REUNIÓN DIARIA

Reunión diaria de estado personal para conocer el avance de las actividades asignadas. Ayuda a evitar problemas de incumplimiento y un mejor seguimiento de asignaciones.

3. TRIAGE DE RESULTADOS DE EJECUCIONES

El equipo antes de iniciar cada actividad diaria, debía verificar los resultados de las pruebas automatizadas. Si e caso hubiera una ejecución fallida, el tester debía averiguar si era un falso positivo o un bug. Y realizar el proceso adecuado dependiendo del resultado. Las tareas de mantenimiento también son consideradas como parte del desarrollo de pruebas automatizadas.

4. DESARROLLO DE PRUEBAS SELECCIONADAS

En esta sub etapa se implementa los casos de prueba seleccionados para la iteración 4 considerando las horas estimadas. También se podría considerar trabajo no realizado de la iteración anterior.

5. EJECUCIÓN DE LA SUITE DE PRUEBAS

Al final de cada día se debía ejecutar la suite de pruebas al final del día utilizado la herramienta de CI, con el objetivo de que cada inicio del día, el equipo sepa la estabilidad del software. Si en caso hubiera fallas, en el proceso de Triage evaluará e identificará la causa de la falla.

6. REUNIÓN DE ESTATUS

Al final de cada iteración se tiene una reunión en la cual se indica:

- a) Si se logró la meta definida, es decir automatizar todos los casos definidos para la iteración.
- b) En caso de que no se logre a la meta, se renegociara las metas para la siguiente iteración.

ITERACIÓN 5

La iteración 5 tiene una duración de 1 semana y contiene las siguientes etapas.

1. CREACIÓN DE TARJETAS KANBAN

Se crearán las tarjetas de los casos de prueba seleccionados para la Iteración 5.

2. REUNIÓN DIARIA

Reunión diaria de estado personal para conocer el avance de las actividades asignadas. Ayuda a evitar problemas de incumplimiento y un mejor seguimiento de asignaciones.

3. TRIAGE DE RESULTADOS DE EJECUCIONES

El equipo antes de iniciar cada actividad diaria, debía verificar los resultados de las pruebas automatizadas. Si e caso hubiera una ejecución fallida, el tester debía averiguar si era un falso positivo o un bug. Y realizar el proceso adecuado dependiendo del resultado. Las tareas de mantenimiento también son consideradas como parte del desarrollo de pruebas automatizadas.

4. DESARROLLO DE PRUEBAS SELECCIONADAS

En esta sub etapa se implementa los casos de prueba seleccionados para la iteración 5 considerando las horas estimadas. También se podría considerar trabajo no realizado de la iteración anterior.

5. EJECUCIÓN DE LA SUITE DE PRUEBAS

Al final de cada día se debía ejecutar la suite de pruebas al final del día utilizando la herramienta de CI, con el objetivo de que cada inicio del día, el equipo sepa la estabilidad del software. Si en caso hubiera fallas, en el proceso de Triage evaluará e identificará la causa de la falla.

6. REUNIÓN DE ESTATUS

Al final de cada iteración se tiene una reunión en la cual se indica:

- a) Si se logró la meta definida, es decir automatizar todos los casos definidos para la iteración.
- b) En caso de que no se logre a la meta, se renegociara las metas para la siguiente iteración.

3.3. EVALUACIÓN

3.3.1. EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO-

BENEFICIO

La evaluación económica para el proyecto de implementación del framework de automatización se puede calcular en base a los recursos humanos, tiempo y herramientas utilizadas.

Tabla 8. Presupuesto estimado por implementación del framework de automatización

Recurso	Tiempo (semanas)	Costo/Mensual(soles)	Subtotal(soles)
Líder técnico de calidad	5	S/ 2000	S/ 2500
			Total: S/2500

Tomando como referencia que 1 tester se demora en promedio 2 minutos para ejecutar la prueba de servicios de un endpoint. Lo que significa que 1 tester en 4 horas (horario part-time) podría ejecutar 120 pruebas de servicio.

Ahora tomando de referencia la Figura 7, para ejecutar 120 pruebas de servicio utilizando las pruebas automatizadas obtendríamos los siguientes resultados.

Tabla 9. Tiempos estimados considerando los tiempos obtenidos en el reporte de Cucumber (Figura 7)

	El tiempo más rápido	El tiempo más lento	El tiempo promedio
Tiempo	23ms	31s	894ms
Tiempo para ejecutar 120 pruebas	0.046 min (2.76 s)	62 min	1.8 min
Nota. Elaboración propia			

Ahora si calculamos qué porcentaje de 4 horas es cada una de las ejecuciones tendríamos.

Tabla 10. Tabla de porcentajes en base a 4 horas de trabajo

	El tiempo más rápido	El tiempo más lento	El tiempo promedio
Tiempo ejecución	0.046 min	62 min	1.8 min
Porcentaje de 4 horas	~0.02%	~25.8%	~0.75%
Nota. Elaboración propia			

Con esto podríamos deducir que el tiempo de ejecución de pruebas se redujo entre un **74.2% a 99.8%**. A su vez, considerando que el día trabajado de 1 tester es de S/100 soles (monto inferido de la tabla 8), se podría inferir que el costo de ahorro por día para 120 ejecuciones varia desde un **S/74.2 a S/99.8**.

Adicional al ahorro económico, la ejecución usando pruebas automatizadas permite al tester utilizar ese tiempo para actividades en paralelo, lo que aún aumenta más el ahorro.

3.3.2. INTERPRETACIÓN DEL VAN Y DEL TIR

El presupuesto estimado de la primera etapa del proyecto se basa en el costo de recursos y la cantidad de empleados para su desarrollo.

Tabla 11. Estimación en tiempo para la implementación del framework de automatización

Recurso	Semana	Hora/Dia	Cantidad de horas semanales
Líder técnico de calidad	1	4	20
Líder técnico de calidad	2	4	20
Líder técnico de calidad	3	4	20
Líder técnico de calidad	4	4	20
Líder técnico de calidad	5	4	20
			Total: 100 horas

Tabla 12. Costos de tecnología utilizada para la implementación del framework de automatización.

Tecnología	Descripción	Subtotal (soles)
Serenity-BDD	Open source	S/0
Maven	Open source	S/0
Java	Open source	S/0
GitLab	Repositorio privado	S/366.3
		Total: S/366.3

Nota: Elaboración propia

Tabla 13. Cantidad de casos de pruebas automatizados

Producto	Casos automatizados	Porcentaje
Mype-CreceMas	55	20%
Distribuidor-CreceMas	50	24%
Endpoints/Servicios	292	34.25%

Esto indica que para la implementación del framework de automatización Serenity-BDD y desarrollo de la implementación de los casos de prueba asignados para la primera etapa tuvo un costo total a S/2866.3 nuevos soles.

CAPITULO IV REFLEXIÓN CRÍTICA DE LA

EXPERIENCIA

La metodología ágil Kanban y su forma de trabajo se ajustó a las necesidades requeridas por el proceso de implementación del framework, permitiendo la realización del trabajo de la implementación a tiempo.

La experiencia del autor en implementaciones de framework de automatización ayudo a que se definieran las actividades requeridas a realizar con anticipación, así estas tareas eran apropiadamente estimadas y podrían ser incluidas como parte del trabajo. Evitando en gran parte, añadir más tareas de las que ya se habían planificado.

El autor planteó Serenity como framework de automatización basándose en la experiencia previa de haber trabajado con el framework, sabiendo sus beneficios y limitantes. También considerando en la expansión de casos de prueba, tales como: pruebas para web responsive, pruebas en móviles y ejecuciones usando herramientas como granja de dispositivos como BrowserStack o SauceLabs.

El seguimiento adicional al trabajo en reuniones semanales y el planteamiento de objetivos a lograr, ayudo a que los entregables (casos de pruebas automatizados) se desarrollen en el tiempo estimado.

El bajo número de integrantes de pruebas en el equipo de desarrollo afecto directamente en el tiempo disponible para el desarrollo de las actividades de automatización, ya que, si el equipo de calidad necesitaba hacer pruebas de funcionalidades no automatizadas, el equipo debía tomar esas pruebas como prioridad y poner en espera cualquier otra actividad. Esto ocasionaba demoras o esfuerzo extra para el logro de los objetivos de automatización.

CAPITULO V CONCLUSIONES Y

RECOMENDACIONES

5.1. CONCLUSIONES

El framework Serenity-BDD provee de configuraciones para poder ejecutar las pruebas de software automatizadas en diferentes ambientes para cada una de las aplicaciones webs.

La automatización del 100% de casos de prueba que dan cobertura a flujos principales de las aplicaciones web permite tener un conjunto de pruebas que al ejecutarse dan como resultado el estado actual de las aplicaciones webs.

La implementación del framework Serenity-BDD permitió automatizar las pruebas de software, y el tiempo de ejecución de estas pruebas se lograron reducir sustancialmente entre un **74% - 99%**. Como consecuencia las pruebas de humo o regresión que son necesarias en el desarrollo de una aplicación toman menos tiempo.

La utilización de los reportes generados automáticamente por el framework de Serenity-BDD ayudaron en visibilidad de los resultados obtenidos. Estos reportes almacenan la información de las ejecuciones, tales como: tasa de casos exitosos, tasa de casos fallidos, tiempo de ejecución, capturas de pantallas, etc. Añadiendo al proyecto una información actualizada de la estabilidad de sus aplicaciones.

5.2. RECOMENDACIONES

El autor basado en el análisis y la implementación del framework de automatización recomienda lo siguiente:

1. Se sugiere que la selección del framework de automatización se seleccione basado en las funcionalidades del proyecto, antes de iniciar, hacer la correcta investigación y evaluación de posibles opciones ayudará a la empresa a evitar gastos y tiempo en desarrollar un framework que no provee del soporte y funcionalidades mínimas para el desarrollo de las pruebas.
2. Los reportes ayudan a la toma de decisiones, lo que tener una en tu framework de automatización es imprescindible.
3. Se recomienda automatizar los casos de prueba que verifiquen funcionalidades principales de las aplicaciones o que son constantemente repetidas manualmente, estas generaran un gran beneficio al ser ejecutadas.
4. La implementación de un patrón de diseño ayudara al mantenimiento y extensión de nuevas clases en el framework. Implementarlo desde un inicio ahorrara retrabajos cuando el framework almacene un gran número de casos automatizados y el mantenimiento y seguimiento de errores de código sean insostenibles.
5. Se recomienda integrar el framework de automatización con alguna herramienta de Integración Continua (CI). Estas herramientas proveen de configuraciones para poder ejecutar las pruebas de software en ambientes que cumplan con las condiciones o especificaciones definidas para la aplicación.

5.3. FUENTES DE INFORMACIÓN

(n.d.). Maven – Welcome to Apache Maven. Retrieved from <https://maven.apache.org/>

Ferguson, J. (2014, December 16). *An Introduction to BDD Test Automation with Serenity and Cucumber-JVM*. Serenity BDD. Retrieved from <http://thucydides.info/docs/articles/an-introduction-to-serenity-bdd-with-cucumber.html>

Introduction. (n.d.). Cucumber. Retrieved from <https://cucumber.io/docs/guides/overview/>

Maven – Introduction. (n.d.). Apache Maven. Retrieved from <https://maven.apache.org/what-is-maven.html>

Nguyen, T. (2019, October 29). *Page Object Model Pattern*. Automated. Retrieved from <https://www.automatedtestingwithtuyen.com/post/page-object-model-pattern>

Pujari, C. (2021, May 16). *Page Object Model and Page Factory in Selenium*. BrowserStack. Retrieved from <https://www.browserstack.com/guide/page-object-model-in-selenium>

¿Qué Es Kanban? Una Descripción General Del Método Kanban. (2021, August 18). Digite. Retrieved from <https://www.digite.com/es/kanban/que-es-kanban/>

¿Qué son las pruebas de software y cómo funcionan?. IBM. Retrieved from <https://www.ibm.com/ar-es/topics/software-testing>

Radigan, D. (n.d.). *Kanban: una breve introducción*. Atlassian. Retrieved from <https://www.atlassian.com/es/agile/kanban>

Rehkopf, M. (n.d.). *¿En qué consiste la integración continua?* Atlassian. Retrieved from <https://www.atlassian.com/es/continuous-delivery/continuous-integration>

Selenium: Definition, How it works and Why you need it. (n.d.). BrowserStack. Retrieved from <https://www.browserstack.com/selenium>

Maida, E. G., & Pacienza, J. (2015). *Metodologías de desarrollo de software*. Biblioteca Central San Benito Abad. (n.d.).

Introduction. (n.d.). The Serenity BDD Book. Retrieved July 8, 2022, from <https://serenity-bdd.github.io/theserenitybook/latest/index.html>

ANEXOS

ANEXO DE INSTALACIÓN Y CONFIGURACIÓN

1. Instalación de Java – Maven – Git

1.1. Java

Paso 1) Instalar el jdk 1.8 desde esta url

<https://www.oracle.com/java/technologies/downloads/#jdk18-windows>, seleccionar la adecuada dependiendo del sistema operativo del ordenador

Java 18Java 17

Java SE Development Kit 18.0.1.1 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

LinuxmacOSWindows

Product/file description	File size	Download
x64 Compressed Archive	172.8 MB	https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.zip (sha256 [?])
x64 Installer	153.38 MB	https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.exe (sha256 [?])
x64 MSI Installer	152.26 MB	https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.msi (sha256 [?])

Paso 2) Instalar el jre desde la siguiente url <https://www.java.com/es/download/>

Paso 3) Proseguir ambos con la instalación por defecto de los instaladores.

1.2. Maven

Paso 1) Ir a la siguiente url <https://maven.apache.org/download.cgi>

Paso 2) Descargar la opción Source zip archive

Downloading Apache Maven 3.8.6

Apache Maven 3.8.6 is the latest release and recommended version for all users.

The currently selected download mirror is <https://dlcdn.apache.org/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup mirrors* (at the end of the mirrors list) that should be available. You may also consult the [complete list of mirrors](#).

Other mirrors:

System Requirements

Java Development Kit (JDK)	Maven 3.3+ require JDK 1.7 or above to execute - they still allow you to build against 1.3 and other JDK versions by Using Toolchains
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

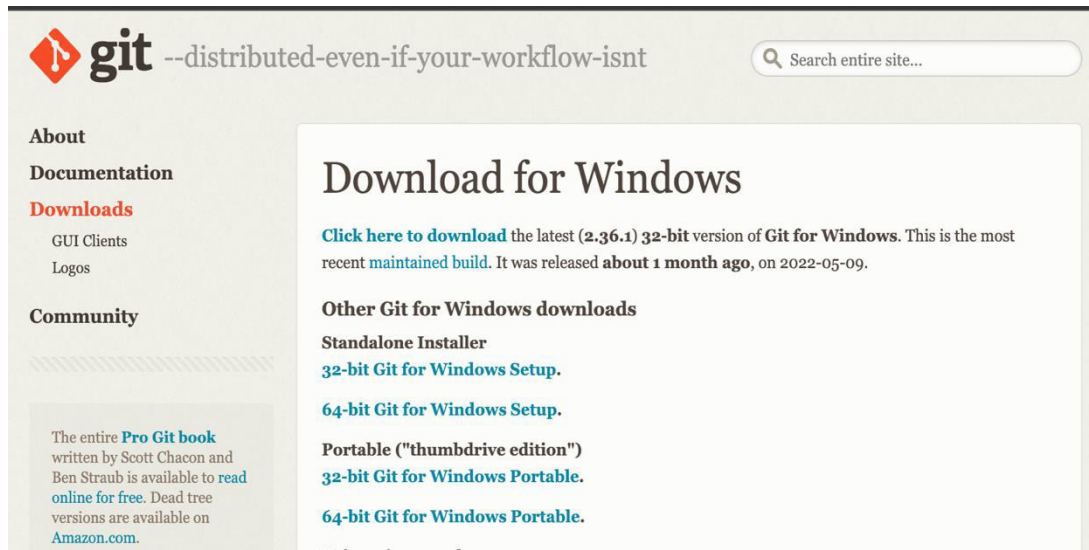
In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.8.6-bin.tar.gz	apache-maven-3.8.6-bin.tar.gz.sha512	apache-maven-3.8.6-bin.tar.gz.asc
Binary zip archive	apache-maven-3.8.6-bin.zip	apache-maven-3.8.6-bin.zip.sha512	apache-maven-3.8.6-bin.zip.asc
Source tar.gz archive	apache-maven-3.8.6-src.tar.gz	apache-maven-3.8.6-src.tar.gz.sha512	apache-maven-3.8.6-src.tar.gz.asc
Source zip archive	apache-maven-3.8.6-src.zip	apache-maven-3.8.6-src.zip.sha512	apache-maven-3.8.6-src.zip.asc

Paso 3) Descomprimir y guardar en una carpeta local

1.3. Git

Paso 1) Ir a la siguiente url <https://git-scm.com/download/win>



Paso 2) Seguir los pasos dependiendo del sistema operativo del ordenador y versión de bits en caso de sistema operativo windows.

2. Configuración de las variables de entorno

2.1. Variables de entorno de Java

Paso 1) Abrir una terminal y ejecutar el comando: **vi .bash_profile**

Paso 2) Agregar las siguientes líneas:

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home
export PATH=${JAVA_HOME}/bin:$PATH
```

2.2. Variables de entorno de Maven

Paso 1) Abrir una terminal y ejecutar el comando: **vi .bash_profile**

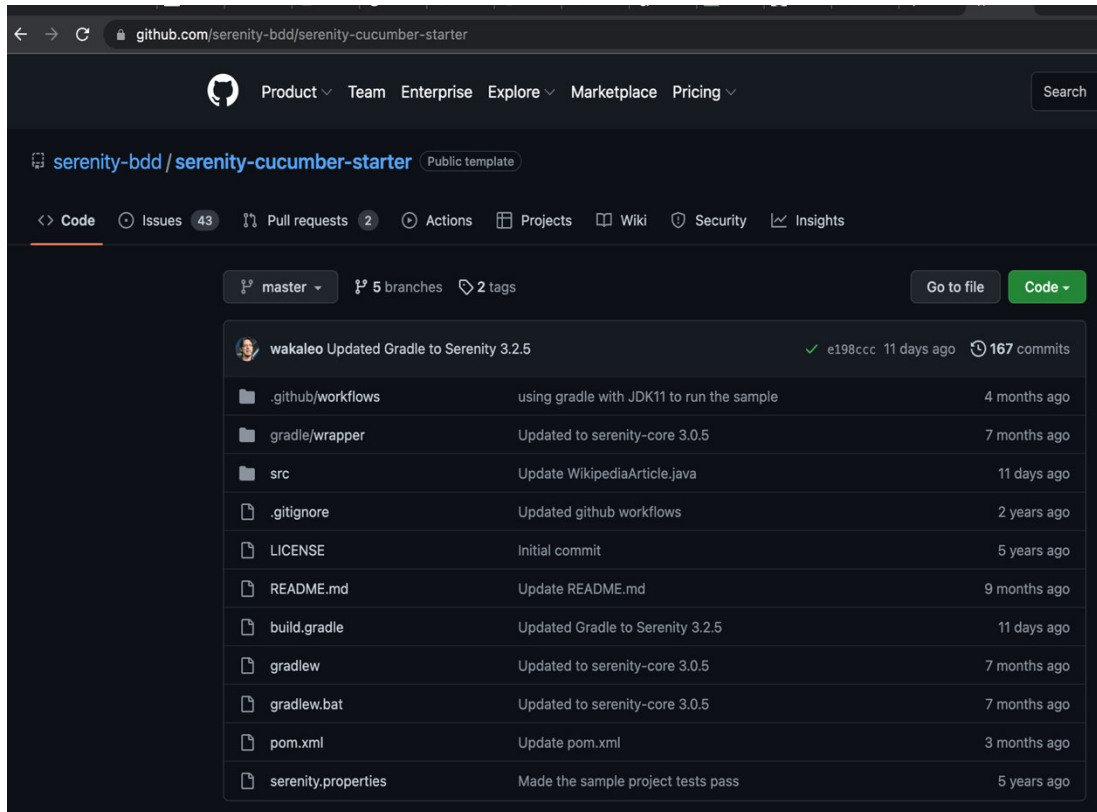
Paso 2) Agregar las siguientes líneas:

```
# MAVEN
export MAVEN_HOME=/Applications/Maven/apache-maven-3.8.4
export PATH=$PATH:$MAVEN_HOME/bin
```

3. Clonación del Framework open-source Serenity BDD

3.1. Clonación del framework Serenity BDD

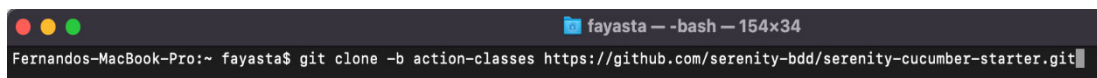
Paso 1) Ir al repositorio del framework <https://github.com/serenity-bdd/serenity-cucumber-starter>



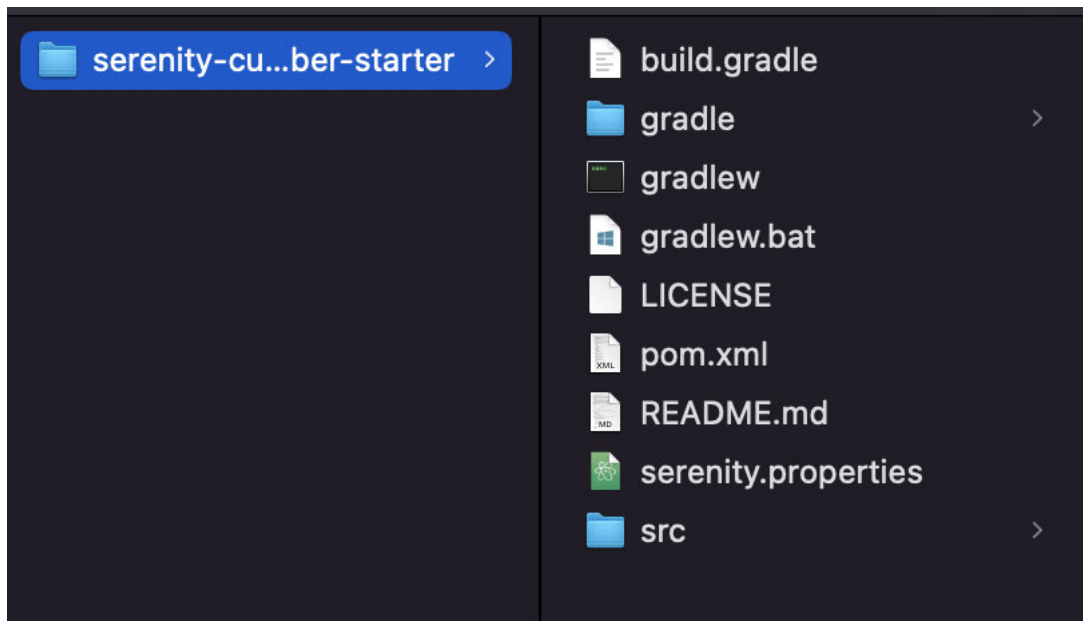
Paso 2) Para nuestro proyecto se implementó la versión que se guía bajo el patrón de Page Object, entonces se debe seleccionar la rama: **action-classes**

Para ello crear una carpeta local y luego abrir un terminal al nivel del folder creado y ejecutar el siguiente comando:

```
git clone -b action-classes https://github.com/serenity-bdd/serenity-cucumber-starter.git
```

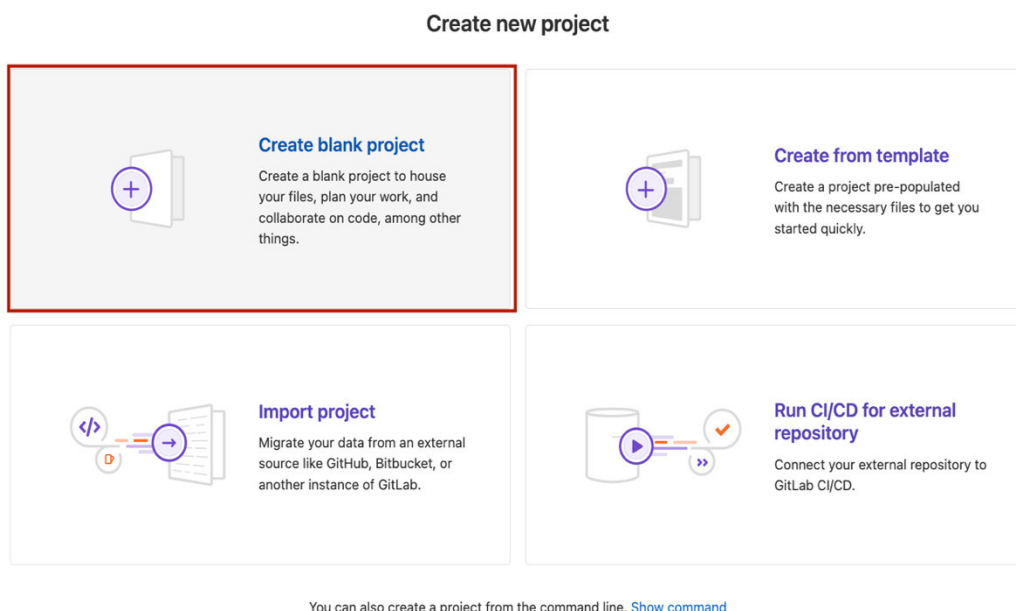


Paso 3) Verificar que el Proyecto de Serenity se haya descargado correctamente, debería tener una estructura similar a la siguiente imagen.



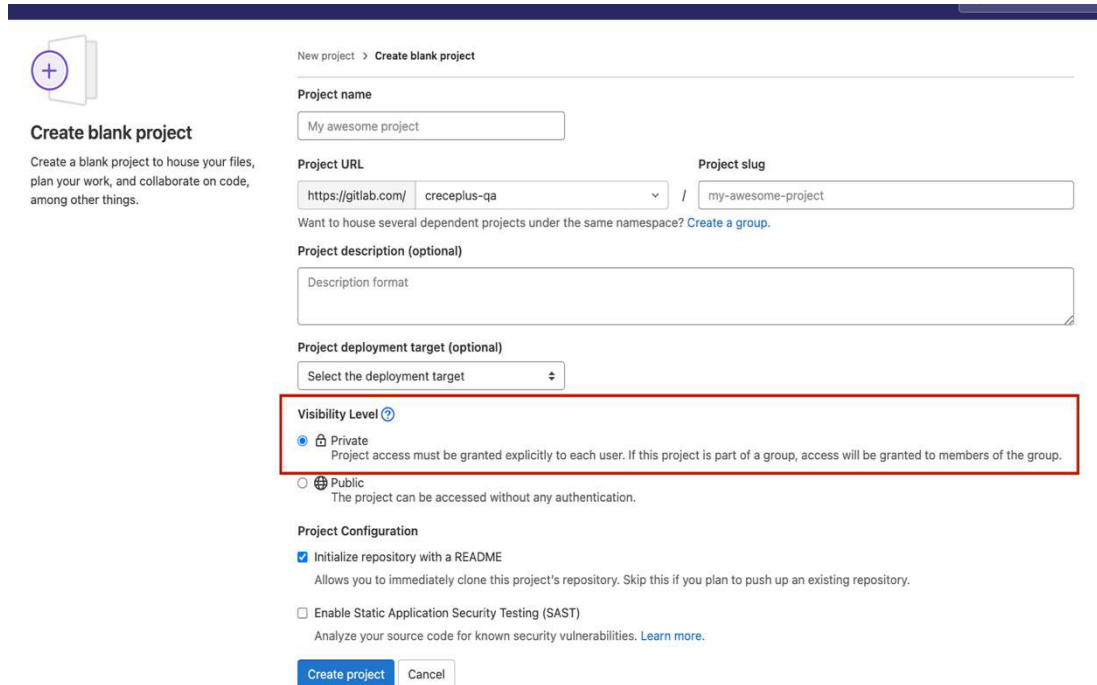
3.2. Crear un repositorio privado en la organización

Paso 1) Ir al repositorio del proyecto que se quiere implementar, en el caso del proyecto CreceMas se utilizó el repositorio GitLab y crear un proyecto en blanco.



Paso 2) Completar los campos requeridos para poder crear un proyecto nuevo. Luego clic en Crear Proyecto.

Se recomienda seleccionar la opción Privada, ya que esto evitara que usuarios no pertenecientes a la empresa puedan visualizar el código y tener vulnerabilidad de seguridad en el sistema.



New project > Create blank project

Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

Project name

My awesome project

Project URL

https://gitlab.com/ creceplus-qa

Project slug

my-awesome-project

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Project deployment target (optional)

Select the deployment target

Visibility Level

☒ Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☐ Public
The project can be accessed without any authentication.

Project Configuration

☒ Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more](#).

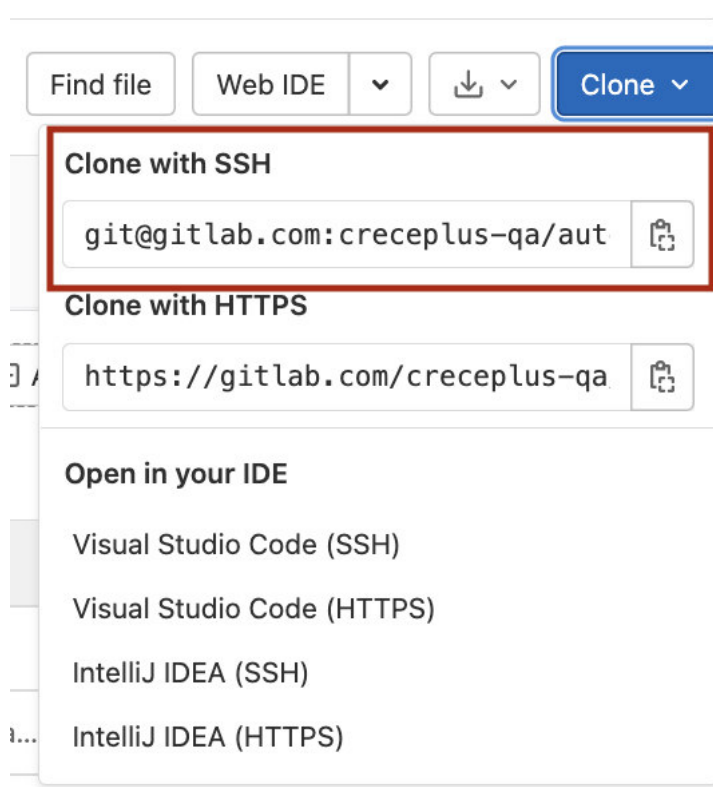
Create project Cancel

3.3. Mover un repositorio local a un repositorio remoto.

Paso 3) La seguridad de GitLab recomienda tener configurada una ssh key para cada usuario perteneciente a la organización. Lo cual se debe configurar siguiendo los siguientes pasos definidos en la documentación de GitLab <https://docs.gitlab.com/ee/user/ssh.html>

Paso 4) Una vez configurada la ssh key, procederemos a crear una nueva carpeta local y clonar el repositorio creado en el paso 2. Una vez creado la nueva carpeta, abrir una terminal.

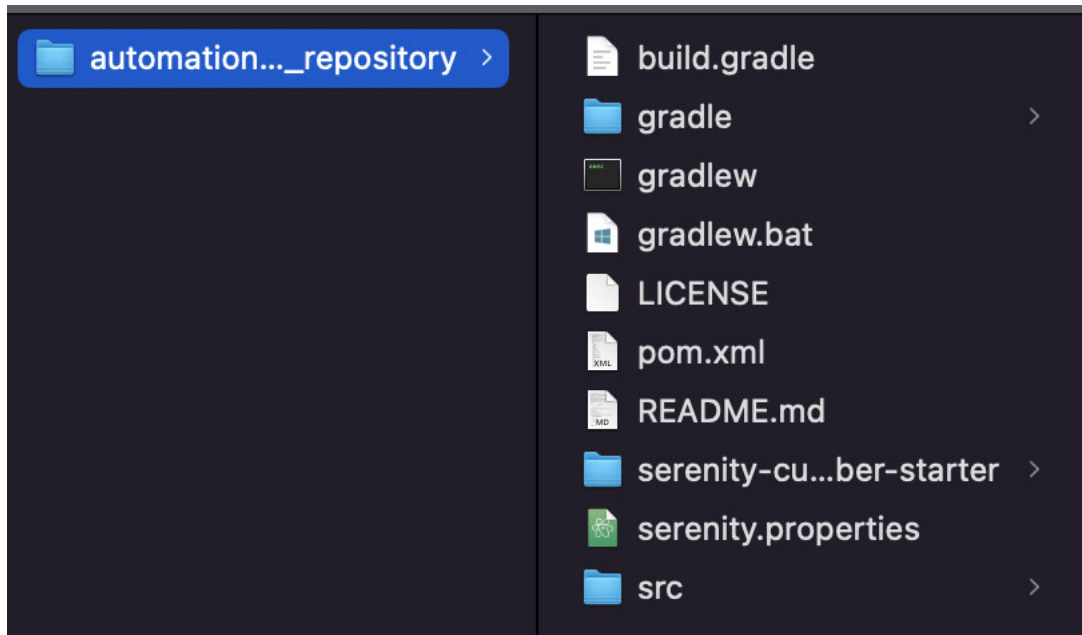
Paso 5) En el repositorio creado en el paso 2, en la opción Clonar, podrás visualizar dos opciones, copia el contenido de la opción Clonar con ssh.



Paso 6) En la terminal, ejecutar el comando `git clone repositorio_url`, donde **repositorio_url** es el contenido copiado en el paso 5. Por ejemplo, para el caso del proyecto el comando seria: `git clone git@gitlab.com:repositorio`

Paso 7) Verificar que se haya clonado correctamente el repositorio en blanco. (solo debería contener el archivo readme).

Paso 8) Copiar el contenido de la carpeta **serenity-cucumber-starter** creado en el paso 3.1. *Paso 2* dentro de la carpeta creado en el paso 4.



Paso 9) Abrir un terminal al nivel de la nueva carpeta y ejecutar el comando **git status**. Se deberá visualizar una imagen como la siguiente.

```
automation_new_repository — -bash — 112x31
Fernandos-MacBook-Pro:automation_new_repository fayasta$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        LICENSE
        build.gradle
        gradle/
        gradlew
        gradlew.bat
        pom.xml
        serenity-cucumber-starter/
        serenity.properties
        src/

no changes added to commit (use "git add" and/or "git commit -a")
```

Paso 10) Para poder subir al repositorio nuevo se deberá ejecutar los comandos en el siguiente orden:

Comando 1: **git add .**

Comando 2: **git commit -m upload Serenity BDD framework**

Comando 3: **git push**

Una vez ejecutado estos 3 comandos, ir al repositorio del proyecto y ya se podrán visualizar las nuevas carpetas mostradas en el paso 9.

The file has been successfully deleted.

main automation_new_repository / History Find file Web IDE

Delete .DS_Store

Fernando Ayasta authored just now

Name	Last commit
gradle/wrapper	upload new framework
src	upload new framework
LICENSE	upload new framework
README.md	upload new framework
build.gradle	upload new framework
gradlew	upload new framework
gradlew.bat	upload new framework
pom.xml	upload new framework
serenity.properties	upload new framework

Con estos pasos ya tendríamos la versión de Serenity-BDD en el repositorio del proyecto deseado.

4. Instalación del IDE IntelliJ versión Community

Paso 1) Descargar el IDE IntelliJ desde <https://www.jetbrains.com/es-es/idea/download/#section=mac> . Seleccionar el sistema operativo y descargar.

Paso 2) Abrir el instalador y proseguir con la instalación por defecto y recomendada por el IDE.

5. Ejecución del test ejemplo de Serenity

Paso 1) Abrir el IntelliJ

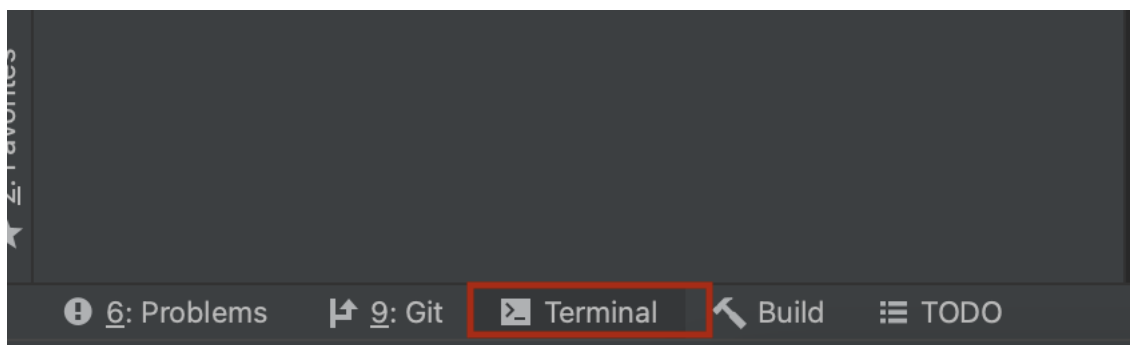
Paso 2) Seleccionar la opción Abrir o Importar

Paso 3) Seleccionar la ubicación de la carpeta creada en el paso 3.2. Paso 4.

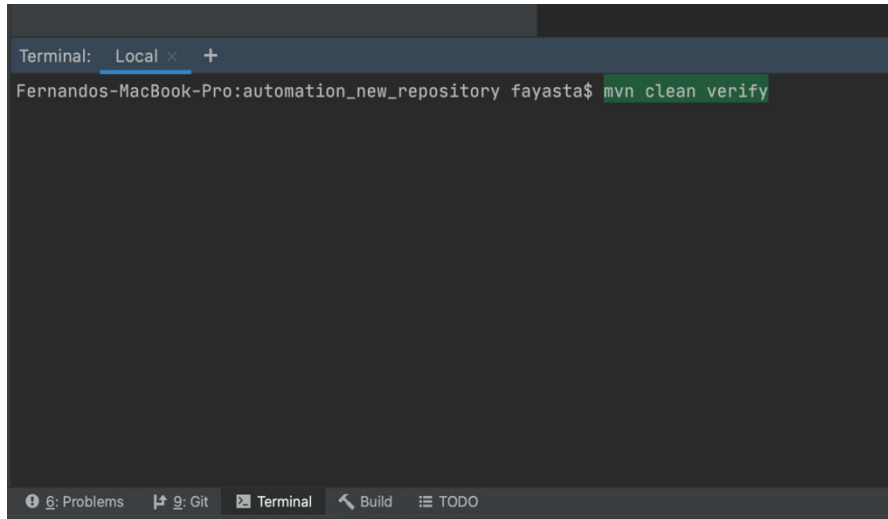
Paso 4) Este mensaje aparecerá, seleccionar Maven y clic en OK.



Paso 5) En la sección inferior de IntelliJ, clic en el tab terminal



Paso 6) En la terminal, ejecutar el comando *mvn clean verify*, este comando permitirá ejecutar la etapa Verify del ciclo de vida de Maven.



```
Terminal: Local x +
Fernandos-MacBook-Pro:automation_new_repository fayasta$ mvn clean verify
```

Paso 7) Después de ejecutar, se obtendrá un resultado similar a la siguiente imagen



```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- serenity-maven-plugin:2.3.2:aggregate (serenity-reports) @ cucumber-starter ---
[INFO] Test results for 1 tests generated in 793 ms in directory: file://Users/fayasta/Clients/CreceMas/untitled%20folder/automation_new_repository/target/site/serenity/
[INFO]
[INFO] SERENITY TESTS : SUCCESS
[INFO]
[INFO] -----
[INFO] | Tests executed      | 1
[INFO] | Tests passed       | 1
[INFO] | Tests failed       | 0
[INFO] | Tests with errors   | 0
[INFO] | Tests compromised  | 0
[INFO] | Tests pending      | 0
[INFO] | Tests ignored/skipped | 0
[INFO] | -----
[INFO] | Total Duration      | 7s 438ms
[INFO] | Fastest test took   | 7s 438ms
[INFO] | Slowest test took   | 7s 438ms
[INFO] | -----
```