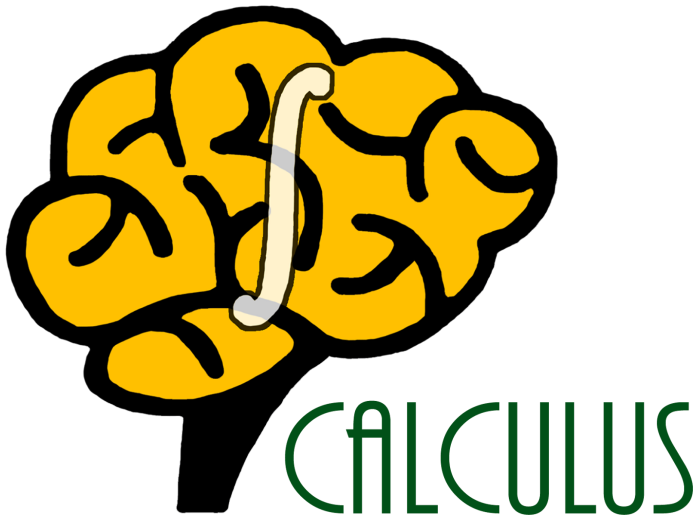
Proyecto Calculus
2016 - Grupo 06
Plan de Verificación y Validación
Versión 4.1



Historia de revisiones

Fecha	Versión	Descripción	Autor
21/09/2016	4.0	Versión inicial	Martín Calcagno
24/09/2016	4.1	Revisión SQA	Manuel Alzugaray

Contenido

1. Introducción

1.1. Propósito

Este Plan de Verificación para el Proyecto Calculus (Grupo 06 - 2016) tiene los siguientes objetivos:

- Identificar la información existente del proyecto y los componentes de software que deben ser verificados.
- Describir las estrategias de verificación que serán utilizadas.
- Identificar los recursos necesarios y proporcionar una estimación del esfuerzo realizado en la verificación.
- Enumerar los entregables del proyecto de verificación.

1.2. Punto de partida

Se verificará el sistema que será implementado por los desarrolladores del equipo. Este consiste en una aplicación web orientada a estudiantes de facultad, que podrá ser accedida a través de dispositivos móviles, y permitirá que los usuarios aprendan y practiquen conceptos de Cálculo 1. Este sistema contará con una plataforma que permitirá a los usuarios registrarse, autenticarse, aprender de distintos temas de la materia, así como también comunicarse con un docente a cargo de la aplicación para que responda sus dudas, o resuelva sus inconvenientes.

1.3. Alcance

- Alcance General: como se decidió utilizar una metodología híbrida (Scrum/MUM), no se definió completamente un alcance global para la verificación del sistema.

- Alcance del Sprint:
    - Se realizarán pruebas unitarias por parte de los desarrolladores de todos los módulos implementados.
    - Se realizarán pruebas de sistema, probando todas las funcionalidades presentes en este prototipo.
    - Las únicas pruebas de desempeño que se realizarán será la de usabilidad o facilidad de uso, y la de interfaz de usuario (estética de la interfaz). Además se asegurará que ninguna funcionalidad tenga un tiempo de respuesta muy elevado (varios segundos)

Se cuenta con el backlog definido casi totalmente, pero dado que se usa una metodología híbrida, el backlog puede ir cambiando, por lo que este plan de verificación y validación podría presentar modificaciones a medida que avanza el proyecto.

1.4. Identificación del proyecto

Los documentos usados para elaborar el Plan de verificación y validación son los siguientes:

- Documento de Requerimientos (Backlog)

1.5. Estrategia de evolución del Plan

- El equipo responsable de la verificación será el encargado de monitorear el plan descrito en este documento.
- Se espera que este documento sea modificado con una frecuencia de una vez por cada iteración, o de una vez por semana.
- Todo posible cambio al plan de verificación y validación será evaluado y analizado por el equipo responsable de la verificación, y a partir de este se podrá determinar qué cambios serán aplicados.
- Cada vez que se realice un cambio en dicho plan, se generará una nueva versión del documento, con fecha y motivo del cambio. Este cambio será comunicado al cliente y al director del proyecto por el medio de comunicación debido.

2. Requerimientos para verificar

La siguiente lista de requisitos describe de manera completa los aspectos a verificar. Dado que se utiliza una metodología híbrida, esta lista seguirá estando sujeta a modificaciones en el correr de las próximas semanas.

Historias a verificar en este Sprint:

- **Historia 200: Autenticar jugador**: Al principio el jugador ingresará al sistema, por correo y contraseña (tal vez luego se pueda incluir una autenticación de jugador mediante facebook o google+).

- **Historia 275: Listado de Temas**: El jugador luego de autenticarse ve los temas posibles y elige uno. El sistema le muestra al jugador un listado de preguntas del tema seleccionado, de acuerdo a su avance.

- **Historia 300: Listado de preguntas**: Luego de seleccionar un tema, habrá un conjunto de preguntas/desafíos de ese tema. El jugador podrá seleccionar una pregunta para responder (H400). El jugador podrá volver al listado de temas.

  Las preguntas se deben ordenar de la siguiente forma: primero las desbloqueadas aún no contestadas, luego las bloqueadas, y finalmente las ya contestadas.

- **Historia 400: Planteo de pregunta:** El sistema le presentará una pregunta al jugador con la posibilidad de responder.

  Las modalidades de respuesta son múltiple-opción, o completar campo en blanco.
  Al responder correctamente, se le hará saber al jugador y se navegará nuevamente al listado de preguntas.
  El nuevo listado de preguntas contendrá eventualmente nuevas preguntas, con un incremento de dificultad respecto a la pregunta recién respondida.
  Si la respuesta es incorrecta, se mostrará un mensaje acorde y se volverá a la pregunta.
  En caso de no saber la respuesta, el jugador podrá navegar a una explicación.

También debe poder regresar al listado de preguntas.

- **Historia 500: Explicación de pregunta:** El sistema muestra la explicación de una pregunta al jugador. Esta explicación puede contener texto, imágenes, links y tal vez video (youtube). Para cada pregunta hay una sola explicación. Se podrá navegar de regreso a la pregunta.

- **Historia 600: Consultar Profesor**: Tanto al mostrar la explicación como la pregunta, el jugador podrá enviarle consultas al profesor. La consulta constará de un texto, pudiendo también contener links.

- **Historia 1100: Profesor, gestión de preguntas y explicaciones**: El profesor podrá acceder a un listado de preguntas y explicaciones, donde podrá agregar, modificar y borrar las mismas.

- **Historia 1200: Profesor, agregar pregunta y explicación**: El profesor podrá agregar preguntas y su explicación asociada al sistema para mostrar al jugador. Las preguntas y explicaciones, podrán incluir los elementos que se nombraron en las historias 400 y 500 (texto, links, etc.). Además podrá seleccionar preguntas que deberán ser respondidas para que esta se le muestre al jugador, así como preguntas que podrían ser desbloqueadas al responder la nueva pregunta, o sea preguntas previas y posteriores a la pregunta actual. Cada pregunta tiene asociado un tema.

- **Historia 1300: Profesor, edición de pregunta y explicación**: El profesor podrá editar preguntas y su explicación asociada para mostrar al jugador. Además podrá seleccionar preguntas que deberán ser respondidas para que esta se le muestre al jugador, así como preguntas que podrían ser desbloqueadas al responder la nueva pregunta. Cada pregunta tiene asociado un tema.

- **Historia 1500: Historial de consultas del profesor**: El profesor podrá acceder a todas las preguntas planteadas y ya respondidas por un jugador. También podrá ver todas las preguntas en general, ordenadas por fecha.

- **Historia 1900: Aumentar puntaje:** Al responder correctamente una pregunta, el puntaje del jugador aumentará, de acuerdo al puntaje de la pregunta respondida, aumentando su ranking en el juego en total. También podrá ganar medallas de acuerdo a la categoría de la pregunta (tema).

  Las medallas se ganarán después de pasado cierto límite en la acumulación de puntaje de cierto tema: bronce, plata y oro. Con la escala de 10000, 20000 y 30000 puntos respectivamente.
  Al ganar una medalla, se mostrará mensaje acorde y se mostrará junto con los puntos obtenidos la misma. Una sustituirá la otra, si gana plata sustituirá a la de bronce, esta última no se mostrará.

- **Historia 2000: Facilidad de uso**: El juego debe ser fácil de usar y atractivo. Flujo principal con la menor cantidad posible de bugs.

- **Historia 2100: Estadísticas de respuestas a preguntas**: El profesor podrá acceder a estadísticas de preguntas, indicando cuántas veces se respondieron bien, cuantas mal, promedio de intentos fallidos.

- **Historia 2200: Gestión de Temas**: El profesor podrá gestionar los temas, podrá agregar, modificar, borrar, y acceder al listado de temas.

Historias que no se van a verificar en este Sprint:

- Historia 100: Registrar jugador
- Historia 250: Desautenticar jugador
- Historia 700: Mostrar nivel de jugador
- Historia 800: Ranking de jugadores
- Historia 900: Historial de consultas del jugador
- Historia 1000: Autenticación del profesor
- Historia 1400: Profesor, responder consulta
- Historia 1600: Reportar error
- Historia 1700: Profesor, estudiar avance del jugador
- Historia 1800: Profesor, ver errores reportados
- Historia 100.1: Prototipo autenticación jugador
- Historia 200.1: Prototipo listado de preguntas
- Historia 300.1: Prototipo planteo de pregunta
- Historia 500.1: Prototipo de explicación de pregunta
- Historia 800.1: Prototipo ranking de jugadores
- Historia 1200.1: Prototipo profesor, agregar pregunta y explicación
- Historia 1900.1: Prototipo Aumentar puntaje, sin medallas

3. Estrategia de Verificación

3.1. Tipos de prueba

3.1.1. Prueba Unitarias

Se van a realizar pruebas unitarias de caja negra y de caja blanca de todos los módulo que se implemente en el sistema. Para ello se utilizará el IDE NetBeans junto con los plugins JUnit y Jacoco. La principal razón para esta decisión es la facilidad de instalación e integración que tienen estas herramientas.

3.1.1.1. Objetivo de la prueba

Verificar los componentes del software en el ambiente de desarrollo, para así poder detectar fallas en los mismos antes de integrarlos con el resto del software.

3.1.1.2. Técnica

Para las pruebas de caja negra se va a utilizar la técnica de partición en clases de equivalencia. Además, se probará

también con valores límite. Para ello, se diseñarán pruebas de manera tal de considerar estas clases de datos de entrada con el objetivo de intentar considerar todos los casos posibles.

Para las pruebas de caja blanca, se aplicará la técnica de cubrimiento de sentencias, para la cual se tomará como oportuno un cubrimiento del 80 % del total de cada módulo. No se llevarán a cabo técnicas de cubrimiento de decisión, ni de decisión/condición, dado que las mismas implicarían evaluar cada una de las combinaciones posibles de valores de cada condición en cada estructura de control del código, lo que puede conllevar un costo en el esfuerzo de testing significante con consecuencias importantes como podría llegar a ser retrasos del proyecto.

3.1.1.3. Criterio de aceptación

Los criterios de detención de las pruebas de caja negra serán el haber realizado todas las pruebas diseñadas, y el corregir todos los errores de nivel de gravedad marginal o superiores que las pruebas de caja negra hayan detectado.

3.1.2. Prueba de Integración

3.1.2.1. Objetivo de la prueba

Asegurar la correcta integración y comunicación entre los módulos del sistema.

3.1.2.2. Técnica

Dado que los únicos módulos a integrar son la base de datos y el servidor web, estas pruebas comenzarán cuando ambas partes estén disponibles. Para probar esta integración se realizarán pruebas con datos válidos (para verificar que se obtienen los datos esperados) y no válidos (para verificar que se despliegan todos los mensajes de error o advertencias apropiadas)

3.1.2.3. Criterio de aceptación

Luego de realizar todas las pruebas planificadas, se deben identificar y registrar todos los defectos encontrados. Además, se deberán corregir todos los defectos encontrados que tengan un nivel de gravedad marginal o superior.

3.1.3. Prueba de Funcionalidad

3.1.3.1. Objetivo de la prueba

Asegurar el correcto funcionamiento de todas las historias incluidas en el sistema.

3.1.3.2. Técnica

Para probar las funcionalidades del sistema se realizarán pruebas con datos válidos (para verificar que se obtienen los resultados esperados) y no válidos (para verificar que se despliegan todos los mensajes de error o advertencias apropiadas). También se harán pruebas de varias funcionalidades a la vez.

3.1.3.3. Criterio de Aceptación

Que el flujo principal sea ejecutado sin errores. Luego de realizar todas las pruebas planificadas, se deben identificar y registrar todos los defectos encontrados. Además, se deberán corregir todos los defectos encontrados que tengan un nivel de gravedad marginal o superior.

3.1.4. Prueba de Usabilidad

3.1.4.1. Objetivo de la prueba

Verificar que la navegación a través de la aplicación que se está probando sea clara y sin ambigüedades, incluyendo un manejo de botones, campos y links; los objetos de los formularios y características, como menúes, tamaño, posición, estado, funcionen de acuerdo a los estándares o convenciones tomadas, proporcionando un uso fluido y una baja curva de aprendizaje para el uso del software producido.

3.1.4.2. Técnica

Para los flujos principales del sistema (las historias más importantes), el cliente pidió que la cantidad de "clicks" necesarios para comenzar a utilizar la aplicación sea bajo. "No deben ser necesarios más de 3 o 4 clicks para empezar a responder preguntas". Por lo tanto, para los flujos principales del sistema se verificará que la cantidad de clicks necesarios para completarlos no sea mayor a 3 o 4.

Además en la vista del alumno, cuando se responden preguntas, deben haber botones para ir hacia atrás. Si se está respondiendo una pregunta, debe haber un botón para regresar al listado de preguntas de ese mismo tema, y otro para regresar al listado de temas.

3.1.4.3. Criterio de aceptación

Cada interfaz de la aplicación debe ser exitosamente verificada, y todos los defectos encontrados que influyan en la facilidad de uso deben ser registrados, y corregidos.

3.1.5. Prueba de interfaz de Usuario (Estética)

3.1.5.1. Objetivo de la prueba

Verificar que la interfaz sea estéticamente aceptable de acuerdo a lo acordado con el cliente.

3.1.5.2. Técnica

Se debe verificar que:

- El color de la interfaz debe ser en escala de azules.
- Los botones de las respuestas de cada pregunta (en caso de ser múltiple-opción) deben tener la misma distancia de separación entre ellos (hacia abajo y hacia los costados).
- Las imágenes utilizadas en las preguntas, y en las explicaciones de las preguntas deben ser de fondo blanco y letras negras.

3.1.5.3. Criterio de aceptación

Cada interfaz de la aplicación debe ser exitosamente verificada, y todos los defectos encontrados que influyan en su estética deben ser registrados, y corregidos.

3.1.6. Prueba de Configuración

Estas pruebas verifican el funcionamiento del software con diferentes configuraciones de software y hardware.

3.1.6.1. Objetivo de la prueba

Verificar que el software funcione apropiadamente en los siguientes navegadores web: Google Chrome (versión 44 en adelante), y Mozilla Firefox (versión 38 en adelante). También se deberá verificar que el software funcione adecuadamente en los navegadores mencionados, en los sistemas operativos Android (version 4.0 en adelante), e iOS (versión 7.1 en adelante). Esto es, que los objetos mostrados en pantalla estén distribuidos de forma adecuada, y que no hayan botones, campos, o menús superpuestos, u ocultos.

3.1.6.2. Técnica

Usar las pruebas de Funcionalidad.

- Abrir una sesión de usuario en los navegadores mencionados, con los sistemas operativos mencionados, ejecutar varias operaciones, y cerrar sesión. Es recomendable utilizar varios dispositivos Android e iOS, con distintas versiones del sistema operativo y distintos tamaños de pantalla, asegurándose que la distribución y el tamaño de los elementos mostrados sea adecuado.
- Repetir el paso anterior tantas veces como se crea necesario.

3.1.6.3. Criterio de Aceptación

Todas las operaciones fueron completadas sin fallas para cada funcionalidad que ha sido probada en estos sistemas operativos y navegadores web.

3.1.7. Prueba de Instalación

Dado que se trata de una aplicación web, no será necesaria una instalación por parte del consumidor, este solo deberá ejecutar un browser o navegador web como se menciona en el punto anterior.

3.1.8. Prueba de Documentación

3.1.8.1. Objetivo de la prueba

Verificar que el documento sea:

- Correcto, esto quiere decir que cumpla con el formato y organización para el documento establecido en el proyecto.
- Consistente, esto quiere decir que el contenido del documento sea fiel a lo que hace referencia. Que no se muestre información de más (que no corresponde con el documento), o de menos (privando a quien lo lea de datos importantes del producto o proceso).

- Entendible, esto quiere decir que al leer el documento se entienda correctamente lo que expresa, sin ambigüedades, además de ser fácil de leer.

3.1.8.2. Técnica

Se hará una revisión al documento del plan de pruebas previa a la generación de los casos de prueba de integración y funcionales basados en historias, con el fin de verificar que el documento contiene toda la información necesaria para la generación de datos de prueba y que ésta sea correcta; esto es, verificar que las salidas son correctas según la especificación de las historias y verificar que las entradas y salidas están correctamente especificadas (no son ambiguas, se corresponden con el diseño del sistema y son comprensibles para el equipo encargado de generar los casos de prueba).

Se hará también una revisión del documento de requerimientos para asegurar que los requerimientos especificados sean verificables o en su defecto sean lo menos ambiguos posible.

Las revisiones serán realizadas por ciertos integrantes del equipo, en presencia de los autores del documento. Se confeccionará una lista de todos los aspectos a verificar de cada documento y se considerará que un documento pasó la verificación si contiene todos los aspectos listados.

Del mismo modo, la documentación generada será inspeccionada siguiendo cierta lista de ítems o aspectos que se deban cumplir para poder pasar la prueba de inspección, y que serán objeto de revisión por parte del equipo de SQA.

3.1.8.3. Criterio de Aceptación

El documento expresa exactamente lo que debe expresar, no hay diferencias entre lo que está escrito y el objeto de la descripción (operación de software, código de programa, decisiones técnicas), y se entiende fácilmente.

3.2. Herramientas

- NetBeans - Entorno de desarrollo - Propiedad de Oracle.
- Git y Github - Control de versiones - Open Source.
- Azure - Ambiente para el deployment - Propiedad de Microsoft.
- Trello - Informe y documentación de errores encontrados - Propiedad de Fog Creek Software.

4. Recursos

En esta sección se presentan los recursos recomendados para el proyecto Calculus, sus principales responsabilidades y su conocimiento o habilidades.

4.1. Roles

En la tabla a continuación se muestra la composición del personal para el proyecto Calculus en el área verificación del Software.

Rol	Cantidad mínima de recursos recomendada	Responsabilidades
Responsable de verificación	1	• Identifica, prioriza e implementa los casos de prueba. • Genera el Plan de Verificación. • Genera el Modelo Casos de Prueba. • Evalúa el esfuerzo necesario para verificar. • Proporciona la dirección técnica. • Adquiere los recursos apropiados. • Proporciona informes sobre la verificación.
Asistente de verificación	6	• Ejecuta las pruebas • Registra los resultados de las pruebas. • Recuperar el software de errores. • Documenta los pedidos de cambio. • Proporciona informes de pruebas unitarias.

4.2. Sistema

En la siguiente tabla se establecen los recursos de sistema necesarios para realizar la verificación.

Recurso	Nombre/Tipo
Servidores	Windows Azure
Servidor cliente alumno	-
Servidor cliente administrador	-
Servidor SQL	-
Servidor Lógica	-

5. Hitos del proyecto de Verificación

La verificación del Proyecto Calculus debe incorporar actividades de prueba para cada verificación identificada en las secciones anteriores. Se deben identificar los hitos del proyecto de verificación separados para comunicar los logros de estado de proyecto.

Actividad que determina el hito	Esfuerzo estimado	Fecha de comienzo	Fecha de finalización
Planificar la verificación	5 horas	20/09/2016	24/09/2016
Elaborar casos de prueba	6 horas	26/09/2016	28/09/2016
Ajuste y Control de Verificación	1 horas	28/09/2016	28/09/2016
Ejecutar la verificación	5 horas	28/09/2016	01/10/2016
Evaluar la verificación	1 hora	01/09/2016	01/09/2016

6. Entregables

6.1. Modelo de Casos de Prueba

Documento	**Modelo de Casos de Prueba**
Creado por	Responsable de verificación
Para quien	Es la guía para realizar las pruebas del sistema y lo usarán los Asistentes de verificación y el responsable de verificación cuando se ejecuten las pruebas del sistema.
Fecha de liberación	02/10/2016

6.2. Informes de Verificación

Documento	Se genera el documento: **Informe de Verificación del Sistema** que resume los errores encontrados con las pruebas de esta aplicación.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	25/09/2016

Documento	Se genera el documento **Informe final de Verificación** que contiene el resumen de los resultados de todas las pruebas realizadas en este iteración
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	02/10/2016

7. Apéndice

7.1. Niveles de gravedad de error

Se asigna un nivel de gravedad a los errores encontrados en la verificación para poder capturar de alguna manera su impacto en el sistema. Además para poder evaluar la verificación y el sistema.

Estos niveles de gravedad son:

- **Catastrófico**: un error cuya presencia impide el uso del sistema.
- **Crítico**: un error cuya presencia causa la pérdida de una funcionalidad crítica del sistema. Si no se corrige el sistema no satisfará las necesidades del cliente.
- **Marginal**: un error que causa un daño menor, produciendo pérdida de efectividad, pérdida de disponibilidad o degradación de una funcionalidad que no se realiza fácilmente de otra manera.
- **Menor**: un error que no causa perjuicio al sistema, pero que requiere mantenimiento o reparación. No causa pérdida de funcionalidades que no se puedan realizar de otra manera.

7.2. Niveles de aceptación para lo elementos verificados

En esta sección se definen niveles de aceptación para los elementos verificados (de modo de poder establecer el estado en el que se encuentra el proyecto), y los criterios de pertenencia a cada nivel.

- **No aprobado**: el elemento verificado tiene errores catastróficos (uno o varios) que impiden su uso o tiene errores críticos (uno o varios) que hacen que el elemento verificado no sea confiable. El usuario no puede depender de él para realizar el trabajo.
- **Aprobado con Observaciones**: el elemento verificado no tiene errores catastróficos, ni errores críticos, pero tiene errores marginales (uno o varios) que hacen que el elemento de software se degrade en algunas situaciones.
- **Aprobado**: el elemento verificado no tiene errores o tiene errores menores que no afectan el normal funcionamiento del elemento.