Proyecto Calculus
2016 - Grupo 06
Plan de Verificación y Validación
Versión 1.2

Historia de revisiones

Fecha	Versión	Descripción	Autor
25/08/2016	1.0	Versión inicial	Martín Calcagno
26/08/2016	1.1	Verificación	Mariano Goicoechea
26/08/2016	1.2	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 le responda dudas, o resuelva sus inconvenientes.

1.3. Alcance

- Alcance General: como se decidió utilizar una metodología ágil (Scrum), no se definió 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 integración verificando la comunicación entre los módulos según su disponibilidad (se comenzará a probar las interacciones entre los módulos que se terminen de desarrollar y testear primero).
  - Se realizarán pruebas de sistema, probando todas las funcionalidades presentes en este prototipo.
  - La única prueba de desempeño que se realizará serán la de usabilidad o facilidad de uso, y se asegurará que ninguna funcionalidad tenga un tiempo de respuesta muy elevado (varios segundos)

  Por el momento no se cuenta con el Backlog definido totalmente, 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 ágil, esta lista seguirá estando sujeta a modificaciones en el correr de las próximas semanas.

Historias a verificar en este Sprint:

- Historia 100.1: Prototipo autenticación jugador: Al principio el jugador ingresará al sistema, por correo electrónico y contraseña. En caso de autenticación correcta, navegará a la historia 200.1, listado de preguntas. En caso de falla de autenticación, mostrará un mensaje acorde. Debe ser lo más atractivo posible para el jugador.

- Historia 200.1: Prototipo listado de preguntas: Luego de autenticarse, habrá un conjunto de preguntas/desafíos. El jugador podrá seleccionar una para poder contestar. En este caso las preguntas no se recuperarán del modelo de datos final, sino estarán hardcodeadas. Habrán unas pocas preguntas, que abarquen los tipos posibles de temas (integrales, derivadas, etc.).

- Historia 300.1: Prototipo planteo de pregunta: El sistema le mostrará al jugador una pregunta, que podrá responder. Al responder correctamente una pregunta, se mostrará un mensaje acorde y se navegará nuevamente al listado de preguntas. El jugador podrá navegar a una explicación de la pregunta. En esta versión simplificada no habrá precedencia de preguntas, todas estarán disponibles a la vez.

Historias que no se van a verificar en este Sprint:

- Historia 100: Registrar jugador
- Historia 200: Autenticar jugador
- Historia 250: Desautenticar jugador
- Historia 275: Listado de Temas
- Historia 300: Listado de preguntas
- Historia 400: Planteo de pregunta
- Historia 500: Explicación de pregunta
- Historia 600: Consultar Profesor
- Historia 700: Mostrar nivel de jugador
- Historia 800: Ranking de jugadores
- Historia 900: Historial de consultas del jugador
- Historia 1000: Autenticación del profesor
- Historia 1100: Profesor, gestión de preguntas y explicaciones
- Historia 1200: Profesor, agregar pregunta y explicación
- Historia 1300: Profesor, edición de pregunta y explicación
- Historia 1400: Profesor, responder consulta
- Historia 1500: Historial de consultas del profesor
- Historia 1600: Reportar error
- Historia 1700: Profesor, estudiar avance del jugador
- Historia 1800: Profesor, ver errores reportados
- Historia 1900: Aumentar puntaje
- Historia 2000: Facilidad de uso
- Historia 2100: Estadísticas de respuestas a preguntas
- Historia 2200: Gestión de Temas
- Historia 400.1: Prototipo de explicación de pregunta
- Historia 700.1: Prototipo ranking de jugadores
- Historia 1100.1: Prototipo profesor, agregar pregunta y explicación

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 aceptable 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 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

Se utilizará la técnica de integración según la disponibilidad de cada módulo (se comenzará a probar las interacciones entre los módulos que se terminen de desarrollar y testear primero). Para probar estas integraciones 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.

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)

3.1.3.3. Criterio de Aceptación

Luego de realizar todas las pruebas planificadas, se deben identificar y registrar todos los defectos encontrados.

3.1.4. Prueba de Interfaz de Usuario

3.1.4.1. Objetivo de la prueba

Verificar que la navegación a través de la aplicación que se está probando refleje los requerimientos, incluyendo 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, y que sean de la forma más

limpia y clara para proporcionar un uso fluido y una baja curva de aprendizaje para el uso del software producido.

3.1.4.2. Técnica

Establecer criterios para determinar cuándo una determinada interfaz de la aplicación se considera "correcta", en el sentido de que satisface los requerimientos establecidos para la facilidad de uso.

3.1.4.3. Criterio de Aceptación

Cada interfaz de la aplicación debe ser exitosamente verificada, y todos los defectos encontrados que afecten a la facilidad de uso deben ser registrados para su reparación.

3.1.5. Prueba de Configuración

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

3.1.5.1. Objetivo de la prueba

Verificar que el software funcione apropiadamente en el siguiente navegador web: Google Chrome (versión 44 en adelante). Solamente se considerará este navegador web por tratarse de un prototipo, en versiones posteriores del software se irán agregando más navegadores web en los que debe funcionar la aplicación.

3.1.5.2. Técnica

Usar las pruebas de Funcionalidad.

- Abrir una sesión de usuario en el navegador mencionado, ejecutar varias operaciones, y cerrar sesión.
- Repetir el paso anterior tantas veces como se crea necesario.

3.1.5.3. Criterio de Aceptación

Todas las operaciones fueron completadas sin fallas para cada funcionalidad que ha sido probada en este navegador web.

3.1.6. Prueba de Instalación

Dado que se trata de un producto 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.

3.1.7. Prueba de Documentación

3.1.7.1. 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 y sin ambigüedades, además de ser fácil de leer.

3.1.7.2. 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 Scrum, 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.

3.1.7.3. 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 (con los plugins JUnit y Jacoco) - Entorno de desarrollo - Propiedad de Oracle

Git y Github - Control de versiones - Open Source

Azure - Ambiente para el deployment - Propiedad de Microsoft

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 de 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 de Prueba. • Evalúa el esfuerzo necesario para verificar. • Proporciona la dirección técnica. • Adquiere los recursos apropiados. • Proporciona informes sobre la verificación.
Asistente de verificación	7	• 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	8 horas	19/08/2016	27/08/2016
Elaborar casos de prueba	15 horas	29/08/2016	31/08/2016
Ajuste y Control de Verificación	3 horas	31/08/2016	31/08/2016
Ejecutar la verificación	8 horas	31/08/2016	02/09/2016
Evaluar la verificación	1 hora	02/09/2016	02/09/2016

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	02/09/2016

6.2. Informes de Verificación

Documento	Se genera un documento **Informe de Verificación Unitaria** por cada prueba unitaria que se realice al sistema.
Creado por	Los desarrolladores 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	Será liberado luego de cada verificación unitaria.

Documento	Se genera un documento **Informe de Verificación de Integración** por cada prueba de integración que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas, junto con el responsable de verificación.
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	Será liberado luego de cada verificación de integración.

Documento	Se genera un documento **Informe de Verificación de Sistema** por cada prueba de sistema que se realice.
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	Será liberado luego de cada verificación de sistema.

6.3. Informe final de verificación

Documento	El documento **Informe final de verificación** es el resumen de la verificación final del prototipo antes de que sea mostrado al cliente para su validación.
Creado por	El Responsable de verificación.
Para quien	Desarrolladores.
Fecha de liberación	Será liberado luego de la verificación final del sistema al final de cada sprint.

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.