BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACION
INGENIERIA EN TECNOLOGIAS DE LA INFORMACION

PLAN DE PRUEBAS
VDALTON

Asignatura: CONTROL DE CALIDAD DE SOFTWARE
Docente: JUAN MANUEL GONZALEZ CALLEROS

Presentan:
LUIS ALBERTO AMADOR FLORES
LUIS ADRIAN PEREZ SAN MARTIN

HOJA DE RESUMEN DE MODIFICACIONES

VERSION	FECHA	CAMBIOS RESPECTO A LA VERSION ANTERIOR	PREPARADO POR	APROBADO POR
0.1	26 / Mayo /2020	Generación del documento	• Luis Alberto Amador Flores • Luis Adrian Pérez San Martin	• Luis Alberto Amador Flores • Luis Adrian Pérez San Martin
0.2	29 / Mayo / 2020	Aumento de casos de prueba	• Luis Alberto Amador Flores • Luis Adrian Pérez San Martin	• Luis Alberto Amador Flores • Luis Adrian Pérez San Martin

Plan de Prueba:	VDALTON
Preparado por:	Luis Alberto Amador Flores Luis Adrian Pérez San Martin 25 / Mayo / 2020

Recursos previos:

- Especificación de Requerimientos
- Descripción de Metodología
- Estimación
- Planificación (Cronograma)
- Plan de Pruebas Selección (Complementario)
- Maquetado de interfaces

Índice

1. Introducción

1.1 Propósito

El propósito fundamental del plan es establecer la cronologías y condiciones para la aplicación de las pruebas, de manera de obtener un producto consistente que tenga una aceptable recepción de los usuarios, y entrar en funcionamiento con todas las operaciones ejecutándose correctamente.

1.2 Alcance

Este documento constituye una guía para el desarrollo de pruebas de forma organizada durante el proceso de construcción del proyecto VDALTON.

Este plan asegura la evaluación de puntos como la funcionalidad, usabilidad, etc.

1.3 Ficha técnica del producto

Características del Producto	
Nombre del Producto	Visual Dalton (VDALTON)
Descripción del Producto	
Descripción General	Se pretende realizar un software de escritorio con la capacidad de poder cambiar la configuración de colores del dispositivo del usuario.
Objetivo	Adaptar los colores de dispositivos para su uso por personas con daltonismo.
Requerimientos del Producto	
Requisitos del sistema	
Hardware	Computadora de Escritorio, Laptop.
Software	Sistema operativo Windows 7 / 8 / 8.1 / 10
Requerimientos	
Funcionales	• Registro de información de usuario (nombre de dispositivo o incluir otro alias) • Mostrar test de evaluación • Mostrar configuración personalizada del usuario • Solicitud de una nueva prueba, si así lo requiere el usuario

No Funcionales	• La aplicación regula los colores de acuerdo con la configuración especifica • La aplicación afecta el S.O • El usuario puede salir en cualquier momento de la aplicación • Módulo de ayuda • Tiempo de respuesta mínimo de 5 segundos • Instalación en arquitectura x86 • Aplicación Portable • Una vez aplicada la prueba, la aplicación se mantendrá oculta en notificaciones para evitar distracciones • Se respetarán las características del sistema operativo
Stakeholders	
Clientes del Producto	• Personas con Déficit de Percepción de Colores (Daltonismo) • Personas en General

1.4 Compromisos organizacionales

1.4.1 Visión del equipo

Crear y desarrollar un software pensando en las necesidades de personas daltónicas

1.4.2 Misión del equipo

Integrar a estas personas al uso de dispositivos sin agravar su interacción mediante un software útil y fácil de manejar

1.4.3 Valores del equipo

Compromiso, honestidad, responsabilidad, respeto, empatía, seguridad

1.5 Metodología

La metodología ágil Scrum es la ocupada de gestionar el proyecto en cuestión, tiene como objetivo la planificación y control con posibilidad de cambio a lo largo de las iteraciones.

Fue elegida por los integrantes por la familiarización que tienen con esta metodología, a la par de centrarse en responder a las exigencias de los usuarios finales.

2. Objetivos y factores de prueba

2.1 Estrategias de prueba

Se planificarán las pruebas especificas a ser aplicadas, por lo que se definen estrategias, recursos y estimación de tiempo.

De manera concreta:

- Se definen las pruebas a aplicarse
- Se describen las técnicas a utilizar
- Se plantea el tiempo de ejecución de cada prueba
- Especificación de uso de herramientas o frameworks
- Criterios de aceptación
- Recursos externos involucrados

2.2 Elementos de prueba

Cuadro resumen de las pruebas

Módulos del sistema a ser probados	Módulos: • Test • Cambio de configuración de color
Objetivos de las pruebas	Objetivos: • Visualizar de forma correcta la información de prueba, según el elegido • Validación de los datos del test • Verificar el funcionamiento de diagnóstico de daltonismo • Verificar que la función de cambio de color se aplique
Detalle de la orden de ejecución de los módulos	Los módulos se ejecutan de manera secuenciada, siendo las alternativas las siguientes.

	Alternativa 1: • Test 1 • Respuesta y Cambio de configuración de color Alternativa 2: • Test 2 • Respuesta y Cambio de configuración de color
Responsabilidad de las pruebas	Las pruebas se llevarán a cabo en su totalidad por el equipo de desarrollo, por lo que los integrantes son los responsables de estas Nota: Algunas pruebas serán realizadas por personas ajenas al equipo

Alcance funcional de las pruebas

Proceso	Funcionalidad	Descripción
Test 1	Seleccionar opción	El usuario podrá seleccionar la opción a la pregunta realizada por el sistema
	Validar opciones	El sistema validara que exista por una opción elegida por pregunta
	Recoger valores	El sistema guardará los valores de las respuestas del usuario para su procesamiento y calcular condición y configuración
Test 2	Seleccionar celda	El usuario podrá seleccionar una celda y ordenarla
	Validar orden	El sistema validara la secuencia de orden que el usuario proporciono

Respuesta y Cambio de configuración de color	Muestra de condición diagnosticada	El sistema mostrara en la interfaz la condición de daltonismo que el usuario pueda tener
	Ejecución de script de configuración	El sistema generara un script con lo valores calculados para cambiar la gama de la pantalla

2.3 Complejidad de casos de prueba

3. Casos de prueba

3.1 Módulos de Prueba

3.1.1 Modulo Test 1

En el módulo Test 1, se considera lo siguiente:

- Se visualizan las instrucciones
- Cada una de las preguntas solo podrá tener una respuesta
- Los campos de respuesta no admiten entrada de teclado
- Cada una de las respuestas se registran en el sistema
- El botón LISTO podrá accederse después de completar todas las preguntas

3.1.2 Modulo Test 2

En el modulo Test 2, se considera lo siguiente:

- Se visualizan las instrucciones
- Todos los campos seleccionables deben ser ordenados
- Ningún campo blanco debe quedar vacío
- El botón LISTO podrá accederse después de completar el ordenamiento

3.1.3 Modulo Respuesta y Cambio de Configuración

En el modulo Respuesta y Cambio de Configuración se considera los siguiente:

- Visualización de diagnóstico, debe ser claro y legible
- Generación y ejecución de script de configuración debe ser rápida

3.2 Entorno y configuración de las pruebas

Para el proceso de prueba del proyecto, se requieren de disponibilidad de los siguientes puntos:

- Equipo de cómputo, escritorio o portátil
- Sistema operativo Windows, no hay preferencia de la versión
- Arquitectura x86 o x64
- Entorno de desarrollo Java (NetBeans, Eclipse, etc.)
- Por comodidad, framework de testing automatizado

4. Estrategia de pruebas

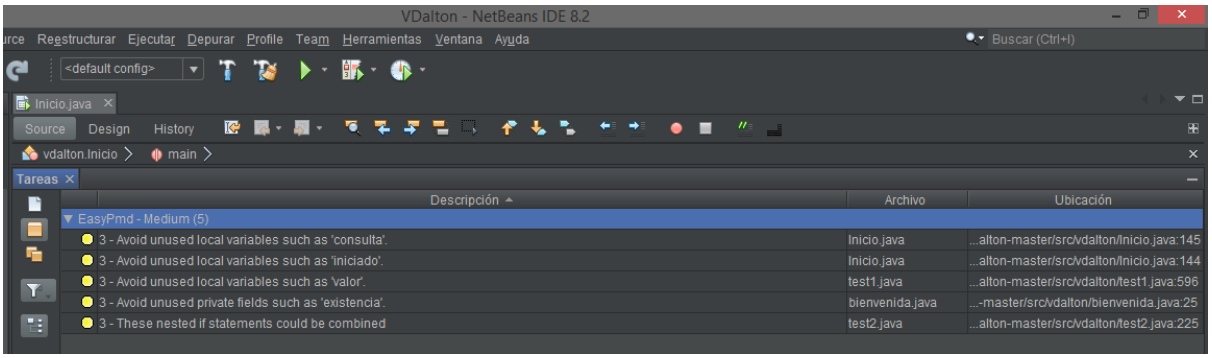
4.1 Pruebas estáticas

Aquellas pruebas que pueden ser ejecutadas a un componente a nivel de especificación o implementación, sin ejecutar el código, sino sólo aplicando una revisión

De acuerdo con las posibilidades encontradas, se realiza pruebas estáticas sobre el código desarrollado con herramientas de análisis sintáctico, en este caso PMD, plugin de Neatbeans, y Checkstyle, un ejecutable en línea de comandos.

A continuación los reportes de las dos herramientas.

PMD:



Siendo en todo el proyecto, las siguientes advertencias:

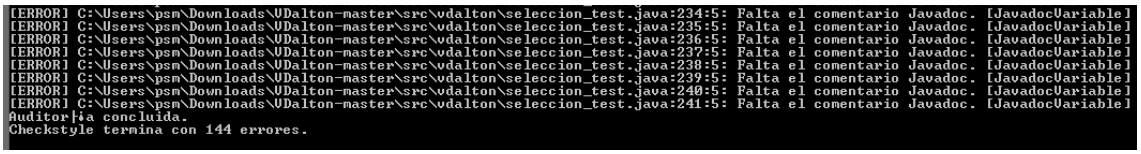
- net.sourceforge.pmd.rules.UnusedLocalVariableRule, el cual diagnostica la declaración de variable locales, pero no se utilizan

En este caso, pmd nos efectúa el análisis sin ningún otro tipo de error.

Checkstyle:

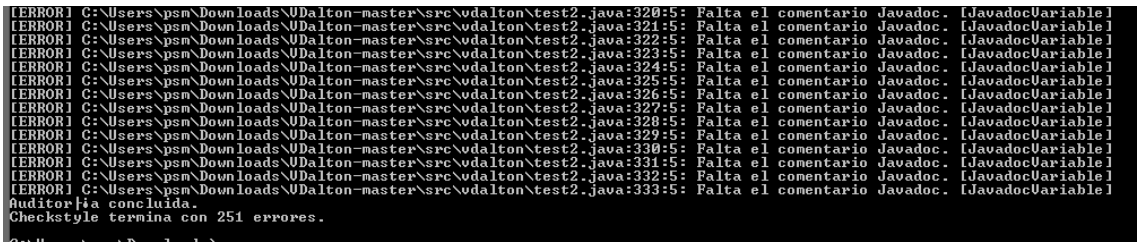
Esta herramienta de análisis de código nos proporciona una auditoria de estándar de codificación, por lo que los errores dentro de los reportes concluyen a tipos de escritura mas no de funcionalidad.

Archivo Inicio.java



Archivo selección_test.java

Archivo test1.java



Archivo test2.java

A primera vista, Checkstyle nos da un reporte detallando algunas fallas, línea por línea, pero revisando de manera objetiva, la mayoría coincide en comentarios o en caracteres dentro de un rango, por lo que, discriminando estos errores, coincidimos con el diagnostico de pmd.

Las revisiones exhaustivas del código a medida de avance del proyecto han sido de relevancia en seguir la codificación de manera que sea entendible.

Puntos de Función

Obtener información del sitio separándola en las siguientes componentes:

- Entradas de usuario
- Salidas de usuario
- Consultas de usuario
- Numero de archivos
- Numero de interfaces externas

Dado esto, se realiza la estimación de puntos de función y puntos de función ajustado, para posteriormente realizar el cálculo del esfuerzo.

Tabla referente:

Tipo / Complejidad	Baja	Media	Alta
Entrada	3 PF	4 PF	6 PF
Salida	4 PF	5 PF	7 PF
Consulta	3 PF	4 PF	6 PF
Archivo	7 PF	10 PF	15 PF
Interfaz externa	5 PF	7 PF	10 PF

Información del sistema:

- Entradas de usuario: 3
- Salidas de usuario: 5
- Consultas de usuario: 2
- Numero de archivos: 0
- Numero de interfaces externas: 1

Por lo que el sistema se identifica con funciones de complejidad baja

Tipo / Complejidad	Baja	Media	Alta
Entrada	3 x 3 PF	4 PF	6 PF
Salida	5 x 4 PF	5 PF	7 PF
Consulta	2 x 3 PF	4 PF	6 PF
Archivo	0 x 7 PF	10 PF	15 PF
Interfaz externa	1 x 5 PF	7 PF	10 PF
Total (PFSA)			40

Factor de ajuste:

Cada característica debe ser especificada en términos de su influencia, utilizándose una escala de 0 a 5

Grado	Descripción
0	No está presente o no influye
1	Influencia Mínima
2	Influencia Moderada
3	Influencia promedio
4	Influencia significativa
5	Influencia fuerte

Factor de ajuste	Valor
Comunicación de datos	0
Procesamiento distribuido	0
Objetivos de rendimiento	1

Configuración del equipamiento	0
Tasa de transacciones	0
Entrada de datos en línea	0
Interfase con el usuario	3
Actualizaciones en línea	0
Procesamiento complejo	2
Reusabilidad del código	4
Facilidad de implementación	0
Facilidad de operación	0
Instalaciones múltiples	3
Facilidad de cambios	1
Factor de ajuste	14

De acuerdo con la fórmula:

PF = PFSA*[(Factor de ajuste * 0.01) + 0.65]

Sustituyendo:

PF= 40* [(14 * 0.01) + 0.65]

PF=31.6 -> 32

Por lo que, con este resultado, se puede estimar que cada punto de función supone 32 líneas de código.

Cálculo de esfuerzo

De acuerdo con la siguiente tabla, se calcula el esfuerzo necesario para desarrollar la aplicación, aunque puede ser poco valorado ya que se sigue una metodología de desarrollo que difiere en productividad

Lenguaje	Horas PF promedio	Lineas de código por PF

Ensamblador	25	300
COBOL	15	100
Lenguajes 4ta Generación	**8**	**20**

Ya que Java se encuentra dentro de los lenguajes de 4ta generación, se estima:

H/H = PFA * Horas PF promedio H/H : Horas Hombre

H/H = 32 X 8

H/H = 256 horas hombre.

Se puede estimar que se trata de un proyecto de índole pequeño.

4.2 Pruebas Dinámicas

4.2.1 Pruebas de caja negra

4.2.2 Pruebas de caja blanca

Pruebas Unitarias, Integración, Sistema, Aceptación, Regresión,

4.3 Prueba de carga

4.4 Prueba de planificación y gestión

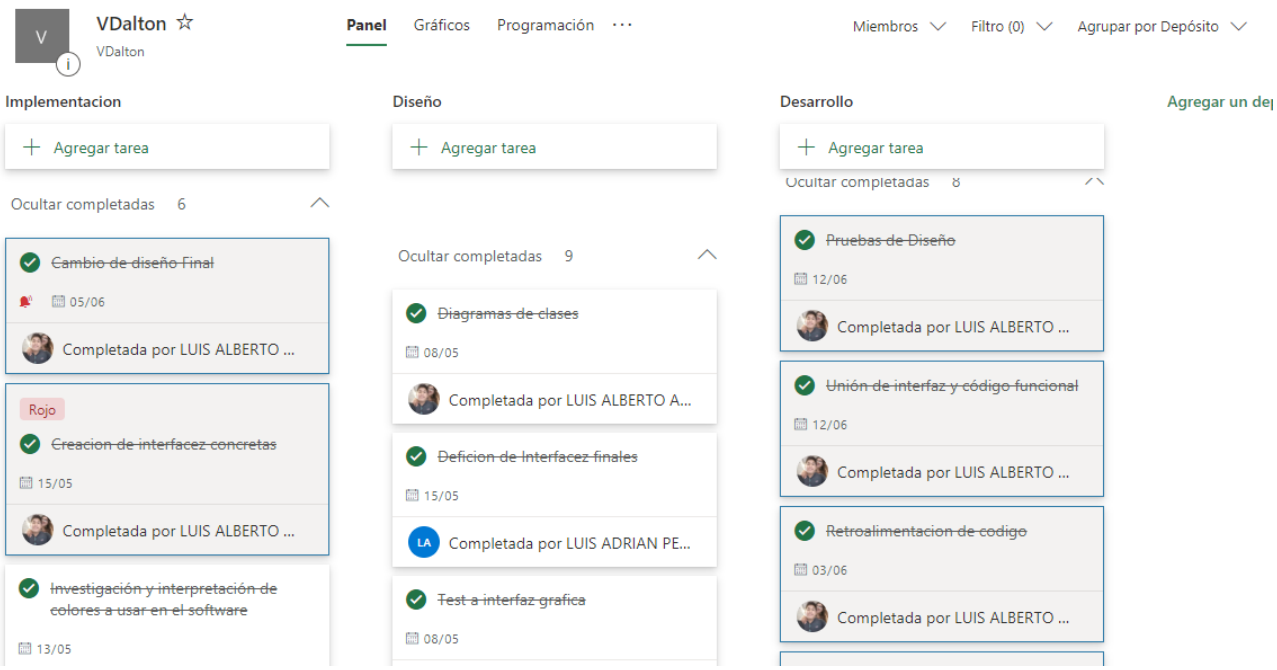
Pruebas con ayuda de herramientas que ayudan en la organización y la administración del ciclo de vida de un proyecto.

Herramienta de planificación ocupada por el equipo: Microsoft Planner.

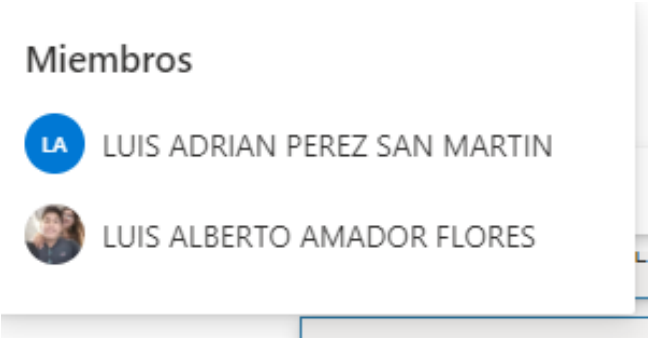
Planner trata de una aplicación de pago por parte Microsoft, el cual sirve para poder gestionar el avance, tareas y tiempo de un proyecto, siendo de áreas distintas, que en este caso fue el desarrollo de una aplicación de software.

Contiene panel de tareas, grafico de asistencia por etapa del ciclo de vida, y programación de las tareas y avance del proyecto.

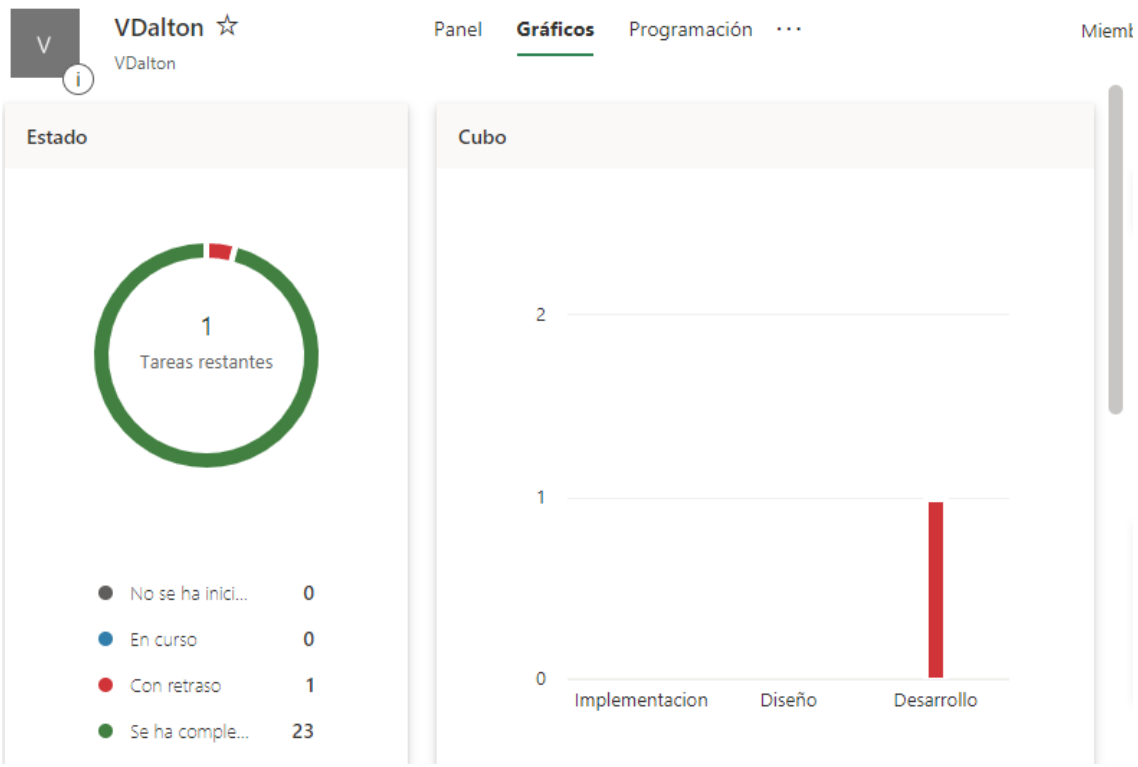
Panel de Tareas:



Miembros del equipo:



Gráficos de avance:



Programación mes / tarea