Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software 1
Obligatorio 2

Marco Fiorito
Agustín Hernandorena

Entregado como requisito de la materia Ingeniería de Software 1

16 de noviembre de 2019

Declaraciones de autoría

Nosotros, AUTORES , declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Resumen

Letra del obligatorio 2 de la materia Ingeniería de Software 1. Se plantea un caso para desarrollar un prototipo aplicando técnicas de ingeniería de software. Se incluye la rúbrica utilizada para evaluar el obligatorio. El código fuente de este documento está disponible para ser usado como plantilla de la documentación entregable.

Índice general

1. Desarrollo de una aplicación

El objetivo del trabajo es el desarrollo de una aplicación, utilizando prácticas tecnológicas y de gestión de la ingeniería de software. Se espera como resultado un software de calidad y la aplicación de un conjunto de prácticas profesionales. En el obligatorio no se evalúa únicamente la implementación, sino el proceso de desarrollo y su impacto en la calidad del software. Los temas centrales de aplicación para este obligatorio son los siguientes:

- Control de versiones
- Código profesional
- Pruebas unitarias automatizadas
- Diseño de interfaz de usuario y usabilidad
- Prueba funcional y reporte de defectos

1.1. Proyecto: 3R EcoShop

Se desea desarrollar una aplicación que soporte las actividad de un **EchoShop** que busca reducir desperdicio en el ciclo de venta. El sistema debe manejar los siguientes conceptos:

1. Gestionar la información básica de artículos. Debe explicar el origen de la materia prima, su respectivo precio, material, código identificador.
2. Empaquetado / envasado: el EcoShop promueve la reutilización de envases, el sistema informa los envases reutilizables que puede aplicarse para cada tipo de producto.
3. Puntos de venta. El sistema deberá mostrar la ubicación de los puntos de venta del EchoShop.
4. Registro de venta. Se deberá registrar la venta, junto con el envase reutilizado y generar ticket electrónico sin imprimir papel. Se podrá utilizar como referencia la normativa de DGI [1].
5. Reportes de estadísticas de relevancia para el negocio: productos más vendidos, envases reutilizados, total de ventas en un mes dado y estimación del beneficio del impacto ambiental generado.

6. Preventa de producto. Se debe disponer de un calendario que permita la preventa de productos. Esto ahorra tiempos de transporte y mejora la conservación de productos perecederos.

7. Se deberá presentar una funcionalidad por parte del equipo que promueva el uso de las 3R y considere aspectos de emprendedurismo del EchoShop.

1.2. Restricciones

- La aplicación debe ejecutar correctamente en la versión de Java disponible en los laboratorios de la Facultad.
- No se requiere autenticación para utilizar el sistema.

1.3. Entrega

El obligatorio se debe entregar según los criterios especificados en entregas en línea de la Facultad de Ingeniería de la Universidad ORT Uruguay.

1. El informe debe estar subido previo a fecha de entrega y no se podrán hacer modificaciones en el repositorio online luego de entregado.

El repositorio debe ser nombrado con el apellido de los integrantes del equipo. Se deben evitar los nombres genéricos para el repositorio, paquete principal o documento. Por ejemplo, los nombres: IS1, Obligatorio, Sistema, Proyecto, Informe, etc. dificultan la identificación del equipo.

Proyecto NetBeans y ejecutable JAR

Se debe incluir todo el proyecto NetBeans con el código fuente y las pruebas unitarias (JUnit). Las referencias a librerías deben ser relativas (el proyecto se puede abrir en cualquier ubicación).

Documentación

Se debe dar evidencia de las actividades realizadas para el desarrollo de la aplicación. En particular, se deben explicar en un documento las actividades de aseguramiento de la calidad.

El documento entregado debe cumplir con los estándares de la facultad para proyectos de fin de carrera [2]. La preparación del documento se debe realizar en LaTeX[1]. El código fuente del presente documento se encuentra disponible[2] para ser usado como plantilla del obligatorio.

[1] LaTeX es un sistema de software para la elaboración de documentos que tiene funciones específicas para creación de documentos técnicos y científicos. Este sistema permite separar la especificación de formato, permitiendo enfocarse en el contenido del documento.

[2] https://www.overleaf.com/read/gghfybswvhjw

2. Rúbrica

En esta sección se presenta la rúbrica del obligatorio, en la que se enumeran los criterios de evaluación usados por los docentes. Además, ayuda a los estudiantes a entender lo que se les pide que realicen y a autoevaluar su trabajo.

Los aspectos evaluados se dividen en distintas categorías. En cada una de ellas se indica el puntaje máximo que se puede obtener.

- Redacción y forma: 3 puntos
- Versionado e instalación: 6 puntos
- Calidad de código: 3 puntos
- Pruebas unitarias: 3 puntos
- Implementación: 6 puntos
- Diseño de interfaz de usuario y usabilidad: 3 puntos
- Pruebas funcionales: 3 puntos
- Reporte de defectos: 3 puntos
- Defensa y reflexión (total de los puntos)

-

3. Plantilla

Esta plantilla sirve como guía del documento que acompaña los otros entregables del proyecto (software ejecutable, proyecto con pruebas unitarias y código fuente).

3.1. Versionado

3.2. Repositorio utilizado

La gestión de versiones implica gestionar grandes cantidades de información para asegurar que los cambios en el sistema se registren y se controlen. Las herramientas de gestión de versiones controlan un repositorio de elementos, para trabajar sobre un elemento de la configuración, debe extraerse del repositorio y colocarlo en un directorio de trabajo. Después de hacer los cambios en el software, se introducirá de nuevo en el repositorio creándose automáticamente una nueva versión.
En nuestro caso, usamos como software de control de versiones Git, mediante este link: https://github.com/Marcoo09/IngSoft1.git se tiene acceso al repositorio online utilizado.

3.2.1. Elementos de la Configuración del Software (ECS)

En un sistema de software puede haber muchos módulos de código fuente, documentación, casos de pruebas, librerías, entre otros. Para seguir el registro de toda esta información, se necesita un esquema consistente de identificación para todos los elementos del sistema de gestión de configuraciones.
Para ello, antes de iniciar el proyecto, nos centramos en el proceso de planificación de la gestión de configuraciones, es decir, decidir exactamente que elementos del sistema se van a controlar.
En nuestro proyecto, decidimos incluir como elementos de configuración al código fuente (proyecto de NetBeans) y a la documentación.

Se deberá incluir un link al repositorio online utilizado. A su vez, se agregará al proyecto el usuario indicado por el docente para poder acceder al mismo.

Identificar los Elementos de la Configuración de Software (ECS) incluidos en el repositorio.

3.2.2. Criterios de versionado

Las ramas son una de las principales utilidades que dispone Git, que nos ayuda a llevar un mejor control del código. Se trata de una bifurcación del estado del código que crea un nuevo camino de cara a la evolución del código, en paralelo a otras ramas que se puedan generar. En nuestro caso, dicidimos hacer uso de Git mediante 3 tipos de ramas: Master (principal), Develop (desarrollo) y Features (funcionalidades). Cada nueva funcionalidad del sistema la realizamos en una rama nueva, específica para esa funcionalidad. Esto lo hacemos principalmente porque no queremos que la versión estable del sistema se vea afectada por esta funcionalidad hasta que no estemos seguros que esta funcionando correctamente (posteriormente hacemos un merge para integrar con el resto de las funcionalidades). Las ramas features, las sacamos de la rama Develop. Una vez que la funcionalidad esté desarrollada, realizamos un commit indicando un mensaje corto pero claro de la nueva funcionalidad del sistema, y por ultimo el comando push para enviar los datos hacia el repositorio remoto. Una vez que la nueva funcionalidad esta en una rama feature, realizamos un pull request, esto es la acción de validar un código que se va mergear de una rama a otra (en nuestro caso de feature a develop), en este proceso de validación, el miembro del equipo que no trabajo en esa funcionalidad se encarga de hacer una revisión del código, y si considera necesario puede dejar comentarios, que le pueden servir de ayuda al desarrollador para realizar cambios en pos de mejoras en cuanto a estándares de codificación, detección de errores, entre otros. En caso de que le parezca que todo esta funcionando correctamente, aprueba el pull request. Una vez aprobado, se hace un merge de la rama sobre Develop, donde se integrará con las demás funcionalidades.

Criterios para versionar los ECS. Explicar los mecanismos de trabajo distribuído, uso de comentarios en los commits, uso de las ramas (develop, master y otras).

3.2.3. Resumen del log de versiones

El resumen debe dar evidencia de la evolución del proyecto en el tiempo y del trabajo realizado por distintos integrantes del equipo.

3.3. Codificación

Nota: en esta sección, no se debe incluir código fuente de todo el proyecto. Se deberá discutir los criterios de calidad de código, pruebas unitarias y herramientas de análisis utilizadas. Solamente se deben incluiran ejemplos de código ilustrativos de los criterios aplicados.

3.3.1. Estándar de codificación

Se deberá respetar el estándar de codificación establecida en el curso y dar ejemplos de su aplicación.

3.3.2. Pruebas unitarias

Exponer los criterios para la codificación de pruebas unitarias. Incluir aquí el análisis de cobertura de pruebas.

3.3.3. Análisis de código

Discutir la aplicación de herramientas de análisis de código, mediciones de calidad y decisiones tomadas para mejorar el código.

3.4. Interfaz de usuario y usabilidad

3.4.1. Criterios de interfaz de usuario

Discutir los criterios aplicados para la construcción de la interfaz de usuario. Por ejemplo: legibilidad, navegación, layout, controles, gestión de errores.

3.4.2. Evaluación de usabilidad

Explicar las técncias usadas para evaluar la usabilidad de la aplicación.

3.5. Pruebas funcionales

3.5.1. Técnicas de prueba aplicadas

Explicar las técnias de prueba aplicadas en el proyecto.

3.5.2. Casos de prueba

Listar los casos de prueba agrupados según la funcionalidad probada. Para cada caso de prueba, idicar entradas y salidas esperadas con datos concretos. Si varios casos comparten un mismo juego de datos de prueba, indicarlo como precondición.

3.5.3. Sesiones de ejecución de pruebas

Indicar las sesiones de ejecución de prueba realizadas, indicar la versión probada, configuación del entorno, tester, fecha y hora. Incluir las sesiones de prueba exploratoria.

3.6. Reporte de defectos

3.6.1. Definición de categorías de defectos

Definir las categorías de defectos que se van a usar en el reporte. Se debe definir en forma separada el tipo y la severidad del defecto.

3.6.2. Defectos encontrados por iteración

Listar todos los defectos encontrados. La descripción del defecto debe ser reproducible. Los defectos deben estar categorizados.

3.6.3. Estado de calidad global

Discutir el estado de calidad global del software al momento de la entrega.

3.7. Reflexión

Discutir los principales resultados y aprendizajes a partir de la aplicación de técnicas de ingeniería de software en el proyecto.

Bibliografía

[1] DGI. (2019) Dgi. [Online]. Available: https://bit.ly/35mBDn3

[2] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf

[3] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, eight ed. McGraw-Hill, 2014.

[4] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.