UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

ET.01. Problema Primer Trabajo Teórico

Donate García, Adrián

Cogolludo Corroto, Carlos

Gómez Fernández, Carlos

Flores Villajos, Manuel

Baltasar Fernández, Jesús

Camacho García, Jaime

Asignatura: Ingeniería del Software II

Titulación: Grado en Ingeniería Informática

Fecha: 16 de noviembre de 2021

1. Introducción

La siguiente aplicación desarrollada tiene la función de gestionar los diferentes tipos de póliza de seguro para las distintas ramas de una aseguradora. Esta aplicación da soporte a sus procesos de negocio, como son gestionar las pólizas, gestionar los pagos o gestionar las diferentes ramas de seguros que tiene dicha aseguradora.

Como en muchas otras aplicaciones, esta también tendrá diferentes roles para los diferentes trabajadores, ya que habrá acciones como la creación de pólizas que pertenece a unos trabajadores determinados o creación de ramas y productos que pertenece a otros.

El desarrollo de esta aplicación sigue el Proceso Unificado de Desarrollo y en el cual empezamos en la interacción 0, para así en la primera semana tener 500€ de presupuesto para todo el proyecto, una agenda para el cliente y a su vez un equipo de trabajo:

Trabajo	Cantidad	Coste/Hora
Analistas	2	90€/hora
Diseñadores	3	60€/hora
Implementadores	10	30€/hora
Testers	5	20€/hora

2. Análisis de requisitos.

Los requisitos funcionales a los que tiene que dar soporte la aplicación software son los siguientes:

1. Crear una nueva rama de seguros
2. Dar de baja una nueva rama de seguros
3. Consultar el listado de ramas
4. Añadir un nuevo producto a una de las ramas
5. Modificar un producto existente
6. Dar de baja un producto, haciendo que deje de esta vigente.
7. Consultar el listado de productos por rama
8. Crear una nueva póliza para un tipo de producto
9. Modificar una póliza para un tipo de seguro
10. Dar de baja una póliza
11. Consultar todas las pólizas por producto en un rango de fechas determinado
12. Consultar el número de producto que tiene contratado un cliente.
13. Gestionar políticas de descuentos en función de los productos contratados.
14. Gestionar los pagos de las pólizas
15. Gestionar los impagos de las pólizas.

3. Supuestos.

Para resolver el problema hemos establecido los siguientes supuestos:

- Suponemos un caso de uso por cada requisito funcional y una iteración por cada caso de uso.
- Suponemos que no hay dependencia software.

- Suponemos ordenación por valor.
- Suponemos que no hay solapamiento.
- Suponemos que los trabajadores dan un 100% de rendimiento
- A la hora de realizar el calendario se tienen en cuenta los festivos nacionales debido a que el grupo de trabajo reside en España.
- Las jornadas laborales son de 8 horas de lunes a viernes.
- Los analistas y los testers trabajan en los requisitos
- La iteración inicial tiene un coste de 500€
- La iteración final tiene un coste de 2500€
- Se define la arquitectura del sistema tras primera iteración.
- Trabajaremos con MVC.
- Suponemos que no hay dependencia entre CDUs

4. Priorización de requisitos y casos de uso.

Para la priorización de los requisitos y casos de uso hemos establecido la siguiente lógica:

- Rama (1) > Producto (2) > Póliza (3) > Pagos (4) > Descuentos (5)
- Crear (1) > Modificar = Eliminar (2) > Consultar (3)

Los casos de uso que se refieran a ramas tendrán más prioridad que los que se refieran a productos y así sucesivamente. También tendremos en cuenta qué se quiere hacer en esos casos de uso, un caso de uso que se refiera a crear tendrá más prioridad que modificar y eliminar, que tendrán el mismo nivel de prioridad y finalmente, estarán los que consistan en consultar.

CDU	RF	DESCRIPCION	Prioridad
1	1	Crear una nueva rama de seguros	1
4	4	Añadir un nuevo producto a una de las ramas	2
2	2	Dar de baja una nueva rama de seguros	3
8	8	Crear una nueva póliza para un tipo de producto	4
5	5	Modificar un producto existente	5
6	6	Dar de baja un producto, haciendo que deje de estar vigente	6
3	3	Consultar el listado de ramas	7
14	14	Gestionar los pagos de las pólizas	8
15	15	Gestionar los impagos de las pólizas	9
9	9	Modificar una póliza para un tipo de seguro	10
10	10	Dar de baja una póliza	11
13	13	Gestionar políticas de descuentos en función de los productos contratados	12
12	12	Consultar el número de producto que tiene contratado un cliente	13
7	7	Consultar el listado de productos por rama	14
11	11	Consultar todas las pólizas por producto en un rango de fechas determinado	15

5. Estimaciones temporales.

Para cada caso de uso hemos estimado en requisitos, análisis, diseño, implementación y *testing* los siguientes tiempos en horas:

REQ	CDU	PRIORIDAD	R	A	D	I	T
RF.1	CDU 1	1	13	20	28	37	10
RF.2	CDU 2	3	3	5	10	11	5
RF.3	CDU 3	7	2	4	12	14	5
RF.4	CDU 4	2	10	16	24	32	10
RF.5	CDU 5	5	8	12	15	20	8
RF.6	CDU 6	6	2	4	8	10	5
RF.7	CDU 7	14	5	8	14	16	7
RF.8	CDU 8	4	16	21	30	40	14
RF.9	CDU 9	10	8	14	16	22	8
RF.10	CDU 10	11	2	4	6	8	5
RF.11	CDU 11	15	2	2	4	8	5
RF.12	CDU 12	13	2	2	4	8	5
RF.13	CDU 13	12	8	10	20	24	12
RF.14	CDU 14	8	16	24	28	34	24
RF.15	CDU 15	9	10	18	20	24	20

Los casos de uso con prioridad 1, 2, 3 y 4 definen la arquitectura del sistema.

6. PUD

Para el desarrollo del PUD nos hemos apoyado en unas tablas realizadas en Excel.

En este documento Excel contiene 3 páginas:

- En la primera (tabla general) establecemos una visión general del proyecto indicando las fases de desarrollo del proyecto, en función de en qué iteración comienzan y terminan, además vemos costes totales de cada iteración y su tiempo de agenda para ser desarrolladas.

En esta podemos observar que nuestra fase de elaboración acaba en la it 4 ya que como hemos mencionado anterior mente, la arquitectura de sistema se nos define al terminar los casos de uso de prioridad 3.

- En la segunda tabla (tabla iteraciones) podemos ver el uso que hacemos de nuestros recursos en las distintas fases de desarrollo (RADIT) así como el coste total de cada iteración y cada recurso, además de las horas totales, tanto de cada iteración como de cada recurso

Esta tabla solo contendrá datos sobre las 15 iteraciones correspondientes a los casos de uso, ignora así la iteración inicial y final ya que se nos facilitan en el enunciado sus tiempos de agenda y sus costes.

- En la tercera tabla nos encontramos la agenda le proyecto más detallada, es decir, día a día detallada en un calendario, incluyendo el número de semana en el que nos encontramos

En esta tabla ya si nos encontramos todas las iteraciones detalladas, incluidas la inicial y final, y gracias a ello podemos calcular una fecha estimada de entrega de este proyecto.

(Consultar tablas en el archivo Excel debido a que su extensión impide visualizarlas correctamente en Word).

7. Gestión de la Configuración

Para la gestión de la configuración trataremos varios puntos, en el primero decidiremos la manera en la que vamos a versionar nuestra aplicación, definiremos el lenguaje de programación que utilizaremos y nuestro agente de bases de datos para realizar la aplicación y finalmente definiremos cuando haremos cambios lo suficientemente grandes como para cambiar de versión.

Nuestra manera de versionar la aplicación será utilizando *Semantic Versioning*, utilizando el formato X.Y.Z siendo:

X: *major*, representa la versión.
Y: *minor*, representa los cambios realizados en iteraciones intermedias, se debe reestablecer a 0 cuando cambiemos de versión *major(X).*
Z: *path*, representa nuestra corrección de errores sobre una versión, se vuelve a 0 tanto cuando cambiamos de *minor* , como cuando cambiamos versión de *path.*

En cuanto al lenguaje de programación utilizaremos *Java* ya que es un lenguaje orientado a objetos que todo nuestro equipo de desarrollo conoce y que nos permitirá modelar el sistema de una manera más sencilla. En cuanto a la base de datos utilizaremos *MySQL*.

En cuanto a cuando haremos los cambios de versionado, basándonos en el control de versiones de S*emantic Versioning* nos encontraremos con las siguientes versiones:

Versión	Caso de uso	Descripción
0.1.0	Crear ramas de seguros	En este caso de uso definiremos las clases de Rama y sus correspondientes gestores además se prepararán las interfaces que tengan que ver con estas
0.2.0	Añadir producto a las ramas	Confeccionamos la base de datos con productos y sus correspondientes gestores además se prepararán las interfaces que tengan que ver con estas, además definimos la relación entre rama y producto
0.3.0	Dar de baja una nueva rama de seguros	Confeccionamos métodos adicionales para la gestión de ramas

1.0.0	**Crear una nueva póliza para un tipo de producto**	Iteración que nos define la arquitectura de sistema, aumentamos el *Major* y la consideramos primera versión funcional. Creación de la clase póliza y sus gestores, implementación del cliente como entidad de la que depende la póliza. Implementar clases gestor y métodos necesarios
1.1.0	Modificar un producto existente	Añadimos los métodos de gestión necesarios e interfaces de usuario necesarios para realizar la modificación de un producto
1.2.0	Dar de baja un producto, haciendo que deje de estar vigente	Añadimos los métodos de gestión necesarios e interfaces de usuario necesarios para realizar la eliminación de un producto
1.3.0	Consultar el listado de ramas	Caso de uso de consulta que implementara métodos de consulta a nuestro agente de bases de datos para la obtención de información sobre las ramas que esta contenga
2.0.0	**Gestionar los pagos de las pólizas**	Creación y almacenamiento en una base de datos de los pagos, registrando cliente póliza importe y fecha
2.1.0	Gestionar los impagos de las pólizas	Modificación del uso anterior, tanto de las bases de datos como de los métodos y clases de gestión para poder gestionar impagos
2.2.0	Modificar una póliza para un tipo de seguro	Añadir métodos de gestión necesarios para modificar las pólizas
2.3.0	Dar de baja una póliza	Añadir métodos de gestión necesarios para eliminar las pólizas
2.4.0	Gestionar políticas de descuentos en función de los productos contratados	Añadir una interfaz con datos sobre descuentos y modificar los métodos para cuando estos valores sean aplicables, poder aplicarlos en la creación de pólizas
2.5.0	Consultar el número de producto que tiene contratado un cliente	Caso de uso de consulta que implementara métodos de consulta a nuestro agente de bases de datos para la obtención de información sobre los productos que esta contenga en función de cada cliente
2.6.0	Consultar el listado de productos por rama	Caso de uso de consulta que implementara métodos de consulta a nuestro agente de bases de datos para la obtención de información sobre los productos que esta contenga en función de cada rama
3.0.0	**Consultar todas las pólizas por producto en un rango de fechas determinado**	Caso de uso de consulta que implementara métodos de consulta a nuestro agente de bases de datos para la obtención de información sobre los productos que esta contenga en función uno de sus atributos, en este caso las fechas

Además del versionado anterior nos reservamos la nomenclatura *path* para el control de errores, fallos o modificaciones que podamos hacer/cometer durante el desarrollo del proyecto.

Además, a todo lo anterior sumamos la implementación de todo el diseño e interfaces de usuario del sistema, incluidos en cada caso de uso.

8. Gestión de la Calidad

Las características de calidad que nos parecen más importantes de cara al desarrollo de este sistema es la seguridad y fiabilidad.

Fiabilidad: en este apartado nos encontramos con unas de las características más importantes, debido a que una de las principales prioridades es la disponibilidad del producto de software. Para lograr alcanzar este objetivo, necesitaremos una base de datos de respaldo que funcione de manera idéntica que la base de datos principal de las propiedades, solo utilizaremos esta base de datos en caso de que la principal llegase a fallar, se apagase o no se pudiera acceder a ella por cualquier motivo. Esto permitiría la disponibilidad de la aplicación informática, así como la capacidad de recuperación. El punto negativo sería el aumento del coste de la implementación y diseño.

Seguridad: debido a que la aplicación es utilizada por diferentes tipos de usuarios, se debe alcanzar un nivel de seguridad para evitar el uso incorrecto del software. Esto puede lograrse implementando un sistema de inicio de sesión, que diferenciaría entre administradores y agentes de seguros. Cada rol tendría restricciones de acceso a los componentes y datos del software con los que no deben trabajar, garantizando la confidencialidad y la integridad. Debe guardarse un registro de las acciones realizadas en el software en la base de datos principal, para garantizar la responsabilidad de cada usuario, y cada usuario (con permisos para ello) podría consultar los cambios realizados sobre datos de clientes o pólizas. Esto aumentaría el coste de su diseño e implementación, pero garantizaría un buen uso del sistema por parte de todos los usuarios.