ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN EDITOR DE TEXTO PARA UN LENGUAJE DE DOMINIO ESPECÍFICO (DSL), PARA LA DETERMINACIÓN DE PROBLEMAS DE OPTIMIZACIÓN LINEAL

MICHELL ANGELLO GUEVARA VELASQUEZ
ESTEFANIA SANCHEZ ROMERO

UNIVERSIDAD TECNOLOGICA DE PEREIRA
FACULTAD DE INGENIERIAS: ELECTRICA, ELECTRONICA, FISICA Y CIENCIAS DE LA COMPUTACION
PROGRAMA DE INGENIERIA DE SISTEMAS Y COMPUTACION
PEREIRA
2015

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN EDITOR DE TEXTO PARA UN LENGUAJE DE DOMINIO ESPECÍFICO (DSL), PARA LA DETERMINACIÓN DE PROBLEMAS DE OPTIMIZACIÓN LINEAL

MICHELL ANGELLO GUEVARA VELASQUEZ
ESTEFANIA SANCHEZ ROMERO

DIRECTOR
CESAR AUGUSTO JARAMILLO ACEVEDO

UNIVERSIDAD TECNOLOGICA DE PEREIRA
FACULTAD DE INGENIERIAS: ELECTRICA, ELECTRONICA, FISICA Y CIENCIAS DE LA COMPUTACION
PROGRAMA DE INGENIERIA DE SISTEMAS Y COMPUTACION
PEREIRA
2015

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del Jurado

Pereira, Mayo de 2015

Yo Michell Angello Guevara Velásquez quiero dedicar mi primer título profesional a mi padre, a mi madre, a mi hermana, a mi novia y a mi Abuelo que en paz descanse.

Yo Estefanía Sánchez Romero quiero dedicar este triunfo a mi querida madre.

AGRADECIMIENTOS

Primero que todo damos gracias a Dios por darnos las aptitudes para cumplir nuestras metas, sin él esto no hubiese sido posible, segundo dar gracias a nuestras familias y amigos que nos dieron su apoyo incondicional para culminar este proyecto, también sin menos merito a los profesores de la Universidad Tecnológica de Pereira por trasmitirnos sus conocimientos durante nuestra etapa de formación, conocimientos que se vieron plasmados en este trabajo, en especial queremos hacer mención al ingeniero Cesar Augusto Jaramillo Acevedo por ser el líder de nuestro equipo de trabajo y guiarnos de la mejor manera y también hacer una pequeña mención al ingeniero Jorge Alberto Gálvez Correa que nos aportó sus conocimientos durante el desarrollo del proyecto, ambos excelentes personas tanto en lo personal como en lo profesional.

CONTENIDO

1. TÍTULO DEL ANTEPROYECTO

"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN EDITOR DE TEXTO PARA UN LENGUAJE DE DOMINIO ESPECÍFICO (DSL), PARA LA DETERMINACIÓN DE PROBLEMAS DE OPTIMIZACIÓN LINEAL"

2. INTRODUCCIÓN

Este proyecto expone la construcción de un módulo, que hace parte de un macro proyecto que consiste en la implementación de un entorno de desarrollo integrado (EDI) de un lenguaje de dominio específico (DSL) para la determinación de problemas de optimización lineal, el cual forma parte de un doctorado. El modulo correspondiente a este trabajo es la construcción de un editor de texto plano para dicho lenguaje.

Un editor de texto plano ofrece la posibilidad de ser leído fácil y rápidamente por la máquina, además de que puede ser interpretado en diferentes plataformas dado que la mayoría de éstas tienen herramientas nativas que pueden leer texto sin formato o plano. Algunas de las ventajas que ofrece un editor de texto son:

- El manejo de la integridad de los datos es una de las grandes ventajas de un texto sin formato, un ejemplo de ello se puede comprobar cuando se codifica una página HTML o se edita un código fuente de programación, ya que es imperativo que los datos permanezcan en formato de texto plano. Cualquier formato adicional, como el formato enriquecido que manejan los procesadores de texto puede alterar la integridad de los datos.

- El manejo de archivos grandes es uno de los inconvenientes del texto enriquecido ya que el código fuente de este puede ser muy grande. Los archivos que van desde varios a varias docenas de megabytes no son poco comunes. Muchos procesadores de palabras no están optimizados para trabajar con archivos grandes, especialmente al realizar búsquedas. En contraste, los archivos de texto usualmente se destacan al manipular archivos grandes.

- Uno de los beneficios fundamentales de muchos editores de textos es su soporte para la búsqueda de expresiones regulares. Las expresiones regulares permiten que un usuario busque patrones en vez de texto específico. Por ejemplo, en vez de buscar una sola dirección de correo electrónico en un documento, las expresiones permiten que un usuario busque y recupere todas las direcciones que se encuentran en el archivo.

- Tanto MacOS X, Linux/Unix como Windows tienen su propia forma de registrar el final de una línea de texto. Los procesadores de palabras no siempre reconocen las diferentes terminaciones de una línea, lo que

ocasiona que el texto se vea perfecto en una plataforma y unido en otra. Los editores de texto tienen un soporte mucho mejor para las diferentes terminaciones de las líneas, y algunos incluso las manipulan de forma transparente.

3. DESCRIPCIÓN DEL PROBLEMA

En la actualidad existen diversos lenguajes orientados a resolver problemas generales o específicos, los cuales se pueden utilizar para trasladar un sistema complejo del mundo real a un modelo computacional para obtener la simulación del comportamiento del modelo deseado.

Los lenguajes de propósito general (LPG) tales como C, Fortran, Basic, Cobol, Lisp, Algol, Pascal, entre otros, tienen la capacidad de expresar cualquier algoritmo, están formados por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación, este proceso puede ser aplicado a cualquier dominio, además suelen estar respaldados por un conjunto de librerías de carácter matemático, expresiones regulares, tiempo y entre otras que permiten el fácil modelamiento de los sistemas, permitiendo disminuir el tiempo de ejecución del problema. Sin embargo para expresar un problema en específico en un lenguaje de propósito general (LPG) puede significar mayor esfuerzo del planeado, dado que posee un alto nivel de abstracción, alta complejidad y requieren mayor asignación dinámica de memoria.

DSL orientados a la optimización se encuentran pocos como OPL, GAMS, AMPL, AIMMS, XPRESS-MP, entre otros, aparte de que en el dominio de la optimización se encuentran escasos DSL estos no cuentan con un entorno de desarrollo integrado (IDE), si se desea desarrollar en alguno de estos DSL se debe utilizar un editor de texto predefinido por el sistema operativo sobre el cual se está trabajando o uno de preferencia personal, además los editores de texto más comunes en el mercado no se complementan con un DSL en el dominio de la optimización, esto no facilita el proceso a la hora de codificar, porque se debe conocer a fondo el lenguaje de optimización sobre el cual esté trabajando y tener en cuenta la extensión, palabras reservadas y sintaxis del lenguaje.

4. JUSTIFICACIÓN

La evolución de la informática en los últimos años ha hecho que los ingenieros desarrollen cada vez mejores algoritmos que brinden soluciones a problemas de toda índole y que los sistemas sean cada vez más expertos; sin embargo estos algoritmos son implementados en lenguajes de propósito general lo cual no permite el nivel de abstracción adecuado para el dominio del problema, puesto que al estar destinados para varios propósitos, estos deben aislar varios elementos del modelo real y no pueden entrar en detalles para generar una solución óptima como sí lo haría un lenguaje de propósito específico. Este último es construido con un conjunto de herramientas y tecnologías mediante unas reglas preestablecidas, las cuales permiten que el DSL ofrezca una gramática reducida para el dominio en el cual se ha creado, además proporciona al usuario mayor facilidad de leerlo, entenderlo y validarlo, siendo así un sistema más experto y ello repercutirá directamente con la mejora en la calidad, confianza, productividad, portabilidad y reusabilidad de las aplicaciones. Dentro de las herramientas de construcción de un lenguaje de propósito específico se encuentra el editor texto, que para mayor facilidad y mejor funcionamiento del DSL debe tener características asociadas al dominio, por ello esta investigación ve la necesidad de que exista un editor de texto para un DSL en el dominio de la optimización de problemas lineales ya que no existe un editor propio asociado a los DSL con las características adecuadas para la optimización de problemas de programación lineal.

Estudiantes del común se verán influenciados con el manejo del DSL, al ser usuarios de este, por ende, podrán interactuar con él por medio de un editor de texto que permita la escritura de problemas de manera sencilla a los cuales quieran dar solución y sin costo alguno dado que este será con fines académicos.

5. OBJETIVOS

5.1. OBJETIVO GENERAL

- Analizar, diseñar e implementar un editor de texto para un lenguaje de dominio específico (DSL) para la determinación de problemas de optimización lineal e incluirlo dentro del entorno de desarrollo integrado de dicho lenguaje.

5.2. OBJETIVOS ESPECÍFICOS

- Obtención de requerimientos, análisis de requisitos, limitaciones, especificaciones y diseño del sistema.
- Desarrollo de la arquitectura del sistema a partir del análisis de requerimientos del editor.
- Realizar la implementación del editor de texto en un lenguaje de programación a partir del análisis y diseño.

6. MARCO DE REFERENCIA

6.1 ESTADO DEL ARTE

Un lenguaje de dominio específico (DSL) se diseña con el fin de resolver situaciones en un dominio específico, además de que se pueda conseguir un nivel de abstracción apropiado del lenguaje[1]. Con base en las ventajas de construir un DSL han desarrollado herramientas que permiten la construcción de un DSL, algunas de ellas son:

ANTLR (ANother Tool for Language Recognition) es una herramienta de lenguaje que provee un framework para la construcción de reconocimiento, interpretación, compilación y translación de descripciones gramaticales que contienen acciones basadas en un lenguaje objetivo como C, C++, C#, Java, entre otros[2]. ANTLR provee un buen soporte para árboles de sintaxis abstracta, descubrimiento de errores y reporte de errores. Además permite la generación de parsers y lexers. ANTLR genera analizadores pred-LL(k), y él mismo utiliza un analizador pred-LL(k) para leer los archivos en los que están escritas las reglas EBNF[3]. ANTLR admite acciones en sus reglas, además de otras prestaciones como paso de parámetros, devolución de valores o herencia de gramáticas[4].

Xtext es un ambiente de desarrollo basado en Eclipse que permite la creación de DSLs con capacidad de edición mediante un entorno de desarrollo integrado (IDE) de Java. Xtext provee APIs para describir los aspectos del DSL mediante la notación EBNF. Basado en la gramática escrita para un lenguaje en particular, Xtext genera una implementación completa en Java[5]. La interpretación en Xtext se facilita mediante el uso de diferentes patrones que capturan el código fuente de un programa basado en el DSL creado, capturando línea por línea ejecutando las instrucciones escritas. Xtext provee un editor de texto robusto y descubrimiento de errores tanto en la gramática como en el código fuente de los programas que se basan de un DSL.

Java CC es un generador de parser y generador de analizador léxico para usar con aplicaciones Java. El generador de parser es una herramienta que lee una gramática y la convierte en un programa de Java que puede reconocer diferentes puntos de la gramática[6].

Estas herramientas de desarrollo tienen su propio IDE para lo cual satisface la necesidad de no tener que desarrollar un editor de texto independiente, además son editores de gran robustez. Aunque se puede ver a un editor de texto como parte de

un IDE, cabe resaltar que muchas veces los editores de texto que se encuentran en los IDEs son menos eficaces que los editores de texto puros, esto debido a que los desarrolladores de texto se centran solo tratar texto, y a veces ofrecen funciones verdaderamente útiles, mientras que los IDEs ofrecen un conjunto de herramientas, y su objetivo es dar la mejor experiencia de usuario al usarlas juntas, no cada una por separado[7].

La Programación Lineal corresponde a un algoritmo a través del cual se resuelven situaciones reales en las que se pretende identificar y resolver dificultades para aumentar la productividad respecto a los recursos (principalmente los limitados y costosos), aumentando así los beneficios. El objetivo primordial de la programación lineal es optimizar, es decir, maximizar o minimizar funciones lineales en varias variables reales con restricciones lineales (sistemas de inecuaciones lineales), optimizando una función objetivo también lineal[8]. En base a estas necesidades surgen varias herramientas para satisfacer estos problemas y algunas de ellas son:

- WINQSB es un paquete de herramientas muy versátil que permite el análisis y resolución de modelos matemáticos, problemas administrativos, de producción, proyectos, inventarios, transporte, entre muchos otros. Ofrece una interfaz básica pero amigable, y es la aplicación por excelencia utilizada por profesionales de Ingeniería Industrial y áreas administrativas para la resolución de sus modelos de programación lineal, continua o entera[9].

- Solver es una herramienta que forma parte de una serie de comandos a veces denominados de "análisis Y si". Con Solver, puede buscarse el valor óptimo para una fórmula de celda, denominada celda objetivo, en una hoja de cálculo. Solver funciona en un grupo de celdas que estén relacionadas, directa o indirectamente, con la fórmula de la celda objetivo. Solver ajusta los valores en las celdas cambiantes que se especifiquen, denominadas celdas ajustables, para generar el resultado especificado en la fórmula de la celda objetivo. Pueden aplicarse restricciones para restringir los valores que puede utilizar Solver en el modelo y las restricciones pueden hacer referencia a otras celdas a las que afecte la fórmula de la celda objetivo, lo cual lo constituyen en una herramienta adecuada para solucionar problemas de programación lineal, y programación lineal entera[10].

- El software TORA de optimización es un programa basado en Windows® que tiene por objeto usarse con muchas de las técnicas presentadas en el libro Investigación de Operaciones de TAHA. TORA es una aplicación muy simple, con una interfaz gráfica de baja calidad. Una de las ventajas de TORA es que puede utilizarse en procesadores de 32 y 64 bits, hoy por hoy su principal desventaja es que deberá ajustarse la configuración de pantalla para adecuarse a sus ajustes de presentación de 800 x 600 y 1024 x 768 pixeles.

Se recomienda el segundo ajuste, porque produce una distribución más proporcionada de la pantalla[11].

- LINGO es una herramienta diseñada para construir y resolver modelos de optimización matemática. LINGO proporciona un paquete integrado que incluye un potente lenguaje para expresar modelos de optimización, un ambiente con todas las funciones para los problemas de construcción y edición, y un conjunto de solucionadores rápidos incorporados, capaces de resolver de manera eficiente la mayoría de las clases de modelos de optimización. Aprender acerca del acceso a las herramientas más poderosas de LINGO a través de su lenguaje es una tarea compleja, sin embargo los modelos que no precisan de un complejo uso de recursos pueden resolverse con una sintaxis sumamente sencilla[12].

6.2 MARCO CONCEPTUAL

Los conceptos claves desarrollados en el presente proyecto son los siguientes:

Lenguaje de dominio específico (DSL): Es un lenguaje de programación orientando a resolver un problema en particular, presenta las herramientas necesarias para dar solución a este, posee un alto nivel de abstracción para el dominio del problema, y ofrece notaciones y construcciones específicas de dominio las cuales permiten expresar la solución más claramente que los lenguajes de propósito general (LPG). El DSL presentando en este proyecto trabaja en el dominio de la optimización para determinar problemas de programación lineal.

Editor de texto: Es un programa que permite la creación y edición de un archivo de texto, para luego leerlo e interpretarlo. Algunos editores de texto cuentan con un conjunto de características para ser usados de manera general, otros únicamente para escribir y programar en un lenguaje.

Optimización: Consiste en determinar un valor máximo o un valor mínimo de una función de una variable, teniendo en cuenta que incluye un conjunto de variables relacionadas con el problema, es decir, la optimización permite obtener los mejores valores entre un conjunto de elementos.

Programación lineal: Es una rama de la programación matemática orientada a la resolución de problemas de optimización, donde la función objetivo define la maximización o minimización de un modelo de programación lineal. Además cuenta

con un conjunto de restricciones que limitan o reducen el grado en el que se puede buscar el objetivo. Su función principal es hacer uso eficiente de los recursos de la forma más óptima buscando alcanzar el objetivo deseado.

Diseño: El concepto a hacer énfasis es el diseño en ingeniería del software, en este ámbito el diseño consiste en una serie de documentos y diagramas que describen la solución de software. La teoría general de sistemas expone que para conocer un sistema se debe describir como un conjunto de variables y su nivel de resolución, como una actividad en el tiempo, como un comportamiento, como una estructura y como estados y transiciones[13].

Requerimientos: Es un área de conocimiento que incluye los requerimientos del software, su importancia y sus principales tipos, es decir, producto vs proceso, funcional vs no funcional y propiedades emergentes[13].

Análisis: Esta área se centra en los procesos de análisis de requerimientos del software, para elaborar los requerimientos del sistema en requerimientos del software, para detectar y resolver conflictos entre los requerimientos y descubrir las fronteras del software. El análisis de requerimientos incluye su clasificación, el modelado conceptual, diseño arquitectónico, asignación de requerimientos y negociación de requerimientos[13].

Implementación: Es la fase que contempla la ejecución del software, donde se involucra al usuario en el desarrollo que se realizando[13].

Lenguajes de propósito general (LPG): Son lenguajes de programación dedicados a resolver problemas de diversos propósitos.

Entorno de desarrollo integrado (IDE): Se define como un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Entre las herramientas se encuentra un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Python: Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma[14].

Notepad++: Es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. De soporte nativo a Microsoft Windows[15].

Gedit: Es un editor de textos compatible con UTF-8 para GNU/Linux, Mac OS X y Microsoft Windows[16].

PyQt4: PyQt4 es un conjunto de herramientas para crear aplicaciones GUI. Es una mezcla de lenguaje de programación Python y la librería Qt. La biblioteca Qt es una de las más poderosas bibliotecas GUI. PyQt4 es desarrollado por Riverbank Computing. PyQt4 se implementa como un conjunto de módulos de Python. Tiene 440 clases y 6.000 funciones y métodos. Se trata de un kit de herramientas multiplataforma que funciona en todos los sistemas operativos, incluyendo Unix, Windows y Mac OS[17].

UML (Lenguaje unificado de modelado): Es utilizado para analizar, especificar, y diseñar esquemas de sistemas de software orientado a objetos. UML está controlado por el grupo de administración de objetos (OMG) y es el estándar de descripción de esquemas de software. Es imposible capturar todos los detalles sutiles de un sistema de software complejo en un solo diagrama grande. El UML tiene numerosos tipos de diagramas, cada uno proporciona una cierta visión de su sistema.

- Diagrama de casos de uso que muestra a los actores (otros usuarios del sistema), los casos de uso (las situaciones que se producen cuando utilizan el sistema) y sus relaciones.
- Diagrama de clases que muestra las clases y la relaciones entre ellas.
- Diagrama de secuencia muestra los objetos y sus múltiples relaciones entre ellos.
- Diagrama de colaboración que muestra objetos y sus relaciones, destacando los objetos que participan en el intercambio de mensajes.
- Diagrama de estado muestra estados, cambios de estado y eventos en un objeto o en parte del sistema.
- Diagrama de actividad que muestra actividades, así como los cambios de una a otra actividad junto con los eventos que ocurren en ciertas partes del sistema.
- Diagrama de componentes que muestra los componentes de mayor nivel de la programación.

- Diagrama de implementación que muestra las instancias de los componentes y sus relaciones[18].

8. DESARROLLO DEL EDITOR

La gramática para la cual ha sido elaborado el editor es la siguiente:

La gramática que se presenta a continuación se encuentra en la notación de Backus-Naur-Form (BNF), para mayor claridad los términos que están en mayúscula y en negrita al lado derecho de las reglas de producción representan palabras reservadas del lenguaje que estarán definidas en el analizador léxico y serán validadas durante el proceso de reconocimiento de componentes léxicos.
Para la notación y estructuración de las reglas gramaticales se usarán los siguientes meta-símbolos:

- **::=** meta-símbolo de definición, indica que el elemento de su izquierda se puede definir según el esquema que se encuentra a la derecha.
- **|** meta-símbolo de opción, indica que puede elegirse uno y solo uno de los elementos separados por este metasímbolo.
- **{}** meta-símbolo de repetición, indica que los elementos incluidos dentro de ellos, se pueden repetir cero o más veces.

Con los anteriores elementos se puede especificar una gramática que genere un lenguaje para un problema de PL, así:

<Programa Lineal > ::= < FunciónObjetivo > ***SA***
< Conjunto Restricciones > (1)

Esta regla indica que un programa Lineal se puede definir como una función objetivo seguido de la palabra reservada SA y un conjunto de restricciones.

< Función Objetivo> ::= <Fo> ***IDEQUAL*** *<Expresión>* (2)

La regla indica que una función objetivo está especificada con el literal Fo seguido de la palabra reservada **IDEQUAL (=)** y una expresión:

<Conjunto Restricciones > ::= <Restricción>
{< Restricción>} (3)

Esta regla permite definir un Conjunto de Restricciones como una restricción seguida de cero o más restricciones

< *Restricción* > ::= < *Expresión*><*Operador Relación* > ***NUMBER | { IDCOMMA }*** <*Operador Relación* > ***NUMBER*** (4)

Regla que permite especificar una restricción como una expresión seguida de un operador de relación y un número.

<*Expresión*> ::= ***NUMBERID*** {<*Operador Aritmético*>***NUMBERID***} (5)

Una expresión se define como un identificador seguido cero o más veces de un operador aritmético y un identificador.

< *Operador Relación* > ::= *GREATEREQUAL* | *LESSEQUAL* | *EQUAL* (6)

Un operador de relación se define mediante palabras reservadas como símbolos de igual, mayor o igual, menor o igual.

<*OperadorAritmético*> ::= ***PLUS*** | ***MINUS*** (7)

La regla especifica que un operador aritmético puede ser definido mediante palabras reservadas como un símbolo (+) o (-).

< *Fo* > ::= ***MAX \ MIN*** (8)

Por último esta regla gramatical muestra que el literal Fo puede ser definido con las palabras reservadas para maximización o minimización.

Para el desarrollo del editor de texto se aplica una metodología de desarrollo ágil (SCRUM) y se toma apoyo en un plan de entregas que detalla cuántas y cuáles historias de usuario se desarrollaran en cada entrega y en que fechas se entregarán.

Se inicia con el desarrollo de la primera iteración (SPRINT en la terminología de SCRUM). En cada una de las iteraciones se realiza una reunión para planificar la iteración (SPRINT Planning Meeting), se desarrollan las tareas de usuario asociadas a ese plan y se finaliza con una reunión de revisión del SPRINT (Review Meeting) y una de retrospectiva (Retrospective Meeting).

Al inicio del SPRINT se realiza una reunión para planificar lo que se va a hacer, se realiza la división en tareas de cada historia de usuario (HU) que se va a desarrollar en la iteración, se estima la duración de esta, se re-planifica la iteración en base a esta nueva estimación, y por último, se asigna cada una de las tareas a los

miembros del equipo de desarrollo. La información obtenida de esta reunión sirve para crear el plan de iteración y para definir el “SPRINT Backlog” de la iteración.

Cada SPRINT permite tener una versión del producto que incluya toda la funcionalidad que describen las historias de usuarios seleccionadas para éste.

Se finaliza el SPRINT con una reunión de revisión en la cual se presenta el producto desarrollado al “Product Owner” (PO) y a los Stakeholders que estén interesados. El PO debe decidir que HU se consideran “terminadas” y cuáles no. Otra actividad importante de esta reunión es volver a analizar la prioridad del Product Backlog para ver si algo ha cambiado o si han aparecido nuevas HU debido a la información obtenida durante el desarrollo de la iteración.

El SPRINT concluye con la reunión de retrospectiva, la reunión es guiada por el “Scrum Master”, y consiste en analizar la forma en la que se aplica el proceso de desarrollo y los impedimentos que se detectan.

El desarrollo del editor se realiza en cinco SPRINT de una duración de dos semanas cada uno, por lo cual el proceso descrito anteriormente se repite para cada iteración.

En cada sprint se utiliza el lenguaje unificado de modelado (UML) para describir el sistema. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.
Los diagramas utilizados para describir el editor son los siguientes:

Diagrama de casos de uso:

Un caso de uso describe las acciones del sistema desde el punto de vista del usuario. En este caso los diagramas modelan la funcionalidad del editor usando un usuario y casos de uso. Los casos de uso son servicios o funciones provistas por el editor para sus usuarios. Las relaciones entre un usuario y un caso de uso, se dibujan con una línea simple, para las relaciones entre casos de uso, se utilizan flechas etiquetadas "incluir" o "extender." Una relación "incluir" indica que un caso de uso es necesitado por otro para poder cumplir una tarea. Una relación "extender" indica opciones alternativas para un cierto caso de uso. (Ver ilustración 1).

Diagrama de clases:

Los diagramas de clases describen la estructura estática de un sistema. Una clase es una categoría que tiene atributos y acciones. Un ejemplo es la clase “Canvas” que tiene atributos como el “ancho”, “alto” y “barra”. Entre las acciones de esta clase se encuentran: “barramenú”, “barraiconos”, “nuevo”, entre otros.

La clase “Canvas” tiene relaciones estáticas con otras clases, por ejemplo, relación de asociación con la clase “MensajeInicial”, es decir, se colaboran entre sí y ninguna depende de la otra. Relación de composición con la clase “archivoTexto”, esta clase se construye a partir de la clase “Canvas”, por tanto su ciclo de vida está condicionado. Relación de dependencia entre la clase “Canvas” y la clase “Resaltar”, sirve para denotar la dependencia que tiene la clase “Resaltar” de la clase “Canvas”. Por último, relación de generalización entre las clases “Canvas” y “QMainWindow”, la cual indica que la clase “QMainWindow” posee características y atributos de la clase “Canvas”. (Ver ilustración 2).

Diagrama de secuencia:

Los diagramas de secuencia muestran la iteración entre objetos a través del tiempo y se modelan para cada caso de uso. En el diagrama secuencia de la ilustración3 el rol de la clase es describir la manera en que el objeto “C” se va a comportar para crear un nuevo archivo de texto, los mensajes son flechas que representan la comunicación entre esos objetos.

Diagrama de colaboración:

El diagrama de colaboración especifica las interacciones entre los objetos en términos de mensajes secuenciados. Los diagramas de colaboración representan una combinación de información tomada de los diagramas de clases, de secuencias y de casos de uso. A diferencia de los diagramas de secuencia, los diagramas de colaboración no tienen una manera explícita para denotar el tiempo, entonces numeran a los mensajes en orden de ejecución. (Ver ilustración 5).

Diagrama de estados:

El diagrama de estados UML captura en cualquier momento una pequeña realidad. Un objeto se encuentra en un estado particular, por ejemplo, el archivo de texto existe o no existe. La transición entre estados se etiqueta con el evento que lo provoca (Ver ilustración 7).

Diagrama de actividades:

El diagrama de actividades ilustra la dinámica del editor mediante el modelado del flujo que ocurre de actividad en actividad. Como se muestra en la ilustración 9 “Nuevo” es un estado que representa la acción y “Canvas” es un objeto, entre ellos

se crea una flecha de flujo de objeto, la cual significa que la acción “Nuevo” está creando o influyendo sobre el objeto “Canvas”.

Diagrama de componentes:

Un diagrama de componentes describe la forma en que se organizan los componentes físicos del editor. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema, existe entre ellos relaciones como las antes nombradas en el diagrama de clases, las relaciones se pueden expresar de dos formas: el componente “Canvas” usa la interfaz que provee el componente “MensajeInicial” o el “Canvas” provee la interfaz que usa el componente “Selección”. (Ver ilustración 59).

Diagrama de paquetes:

El diagrama de paquetes permite organizar los elementos de un diagrama en un grupo. Tal vez quiera mostrar que ciertas clases o componentes son parte de un subsistema en particular. Para ello, se pueden agrupar en un paquete, que se representa por una carpeta, como se muestra en la ilustración 60, en el paquete “Canvas” se agrupan “BarraMenu”, “FuenteLetra”, “BarraIconos”, “BarraEstado” y “Texto”.

Diagrama de despliegue:

El diagrama de distribución UML muestra la arquitectura física del editor. Representa a los equipos y a los dispositivos de entrada, y también mostrar sus interconexiones y el software que se encuentra en el equipo (Ver ilustración 61).

8.1 PLAN DE ENTREGAS

1. Breve descripción del alcance del sistema

El proyecto a abordar consiste en un editor de texto para un lenguaje de dominio específico, para la determinación de problemas de optimización lineal.

2. Listado inicial de historias de usuario

El siguiente listado incluye las historias de usuario que se han obtenido en la reunión de planificación de las entregas del producto, realizada con el cliente y el equipo de desarrollo. El listado incluye la estimación del esfuerzo realizado por el equipo de desarrollo y la priorización de las historias por parte del cliente.

La estimación del esfuerzo esta expresada en Puntos de Historia y la prioridad esta medida en el rango 1 a 3, siendo el valor de 1 el más prioritario para el cliente.

Ident	Título	Est	Prio
HU.1	Un usuario puede crear un nuevo archivo de texto	3	1
HU.2	Un usuario puede abrir un archivo de texto	4	1
HU.3	Un usuario puede guardar un archivo	4	1
HU.4	Un usuario puede escribir en un archivo de texto	6	1
HU.5	Un usuario puede eliminar un archivo de texto	3	3
HU.6	Un usuario puede realizar funciones básicas de edición	--	2
HU.7	Un usuario puede autocompletar en el archivo de texto	5	2
HU.8	Un usuario puede buscar dentro del archivo de texto	3	3
HU.9	Un usuario puede reemplazar una palabra dentro del archivo de texto	3	3

Tabla 1. Historias de usuario

A continuación se describen algunas modificaciones realizadas en el listado inicial de HU durante el análisis para la estimación del esfuerzo y la priorización.

2.1 Historias que se han dividió en varias:

Ident.	Título	Est	Prio
HU.6	Un usuario puede realizar funciones básicas de edición	--	2
HU.6.1	Un usuario puede copiar texto dentro del archivo	½	2
HU.6.2	Un usuario puede cortar texto dentro del archivo	½	2
HU.6.3	Un usuario puede pegar texto dentro del archivo	½	2

HU.6.4	Un usuario puede borrar contenido del archivo	½	2
HU.6.5	Un usuario puede seleccionar todo el contenido del archivo	½	2
HU.6.6	Un usuario puede deshacer la última acción dentro del archivo	½	2
HU.6.7	Un usuario puede rehacer la última acción dentro del archivo	½	2

Tabla 2. HU.6 Funciones básicas de edición

3. Calculo de la velocidad del equipo

Se cuenta con un **equipo de desarrollo** formado por 2 programadores que va a dedicar un 100% de su trabajo al proyecto.

La duración de cada una de las iteraciones a realizar en el proyecto va a ser de 2 semanas.
La estimación realizada del esfuerzo de cada una de las historias de usuario se ha expresado en días ideales de programación. En el entorno de trabajo se estima que un día ideal de programación va a corresponder de 2 a 3 días reales de trabajo.

La **duración de una iteración** es:

1 Iteración (SPRINT) = 2 semanas = 10 Días reales

La **velocidad** del equipo de desarrollo medido en punto de historia es:

2 Programador * 10 = 20 días reales por iteración => de 8 a 10 PH por iteración.
Se ha decidido usar **8 Puntos de historia** como la velocidad estimada del equipo.

4. Descripción de las entregas

Esfuerzo total del proyecto = 34 ½ PH
Velocidad del equipo = 8 PH (por iteración)

En base al esfuerzo necesario y la velocidad estimada del equipo, para el desarrollo del proyecto se van a realizar dos entregas de dos iteraciones cada una y la última entrega de una sola iteración.
El desarrollo del proyecto va a comenzar el día **23 de Febrero del 2015.**

El **plan de entregas** del producto es el siguiente:

Entrega	Objetivo	Fecha de la entrega
1	Tener un editor de texto que permita crear, escribir, guardar y abrir al usuario.	23 de Marzo de 2015

Iteración	Objetivo
1	Permitir al usuario crear un archivo de texto en blanco y escribir en el archivo de texto utilizando el lenguaje de dominio específico.
2	Implementar la función de abrir y guardar con la extensión .odsl para que el usuario pueda leer y asegurar sus archivos de texto respectivamente.

Tabla 3. Plan de entregas No.1

Entrega	Objetivo	Fecha de la entrega
2	Añadir al editor de texto herramientas básicas para que el usuario pueda realizar funciones de edición, autocomplete y busque, además elimine su archivo de texto.	20 de Abril de 2015

Iteración	Objetivo
1	Permitir al usuario realizar funciones de edición a sus archivos tales como copiar, cortar, pegar, seleccionar, borrar, rehacer y deshacer. Además permite autocompletar dentro del archivo palabras reservadas del lenguaje de dominio específico para problemas de optimización lineal.
2	Proporcionar al usuario la herramienta de buscar una palabra dentro del archivo y eliminar un fichero del disco duro.

Tabla 4. Plan de entregas No.2

Entrega	Objetivo	Fecha de la entrega
3	Permitir que el usuario pueda reemplazar	28 de Abril de 2015

Iteración	Objetivo
1	Proporcionar al usuario la herramienta de reemplazar una palabra en su archivo de texto.

Tabla 5. Plan de entregas No.3

5. Lista Inicial del Producto (Product Backlog)

La lista del producto con las historias que se usaran durante el desarrollo es la siguiente:

Ident.	Historias de usuario	Iter	Ent
HU.1	Un usuario puede crear un nuevo archivo de texto	1	1
HU.4	Un usuario puede escribir en un archivo de texto	1	1
HU.3	Un usuario puede guardar un archivo	2	1
HU.2	Un usuario puede abrir un archivo de texto	2	1
HU.6.1	Un usuario puede copiar texto dentro del archivo	1	2
HU.6.2	Un usuario puede cortar texto dentro del archivo	1	2
HU.6.3	Un usuario puede pegar texto dentro del archivo	1	2
HU.6.4	Un usuario puede borrar contenido del archivo de texto	1	2
HU.6.5	Un usuario puede seleccionar todo el contenido del archivo	1	2
HU.6.6	Un usuario puede deshacer la última acción dentro del archivo	1	2
HU.6.7	Un usuario puede rehacer la última acción dentro del archivo	1	2
HU.7	Un usuario puede autocompletar en el archivo de texto	1	2
HU.5	Un usuario puede eliminar un archivo de texto	2	2
HU.8	Un usuario puede buscar dentro del archivo de texto	2	2
HU.9	Un usuario puede reemplazar una palabra dentro del archivo de texto	1	3

Tabla 6. Product Backlog

6. Tarjetas de las historias de usuario

Se incluye una descripción completa de las historias de usuario que se van a tratar en el desarrollo del proyecto, incluyendo los criterios de aceptación de cada una de ellas y la información obtenida en las reuniones de conversación con el cliente.

Identificador: HU.1	Crear un nuevo archivo de texto	
Descripción: Como usuario quiero crear un nuevo archivo de texto en blanco para codificar en base al lenguaje de dominio específico para problemas de optimización lineal.		
Estimación: 3	**Prioridad: 1**	**Entrega: 1**
Pruebas de aceptación:		

• Creamos un nuevo archivo y verificamos que se muestra un lienzo en blanco.
Observaciones:

Tabla 7. Tarjeta de HU.1: Crear

Identificador: HU.4	Escribir en un archivo de texto	
Descripción: Como usuario quiero escribir dentro de un archivo de texto para programar un caso de estudio de optimización lineal		
Estimación:6	**Prioridad: 1**	**Entrega:1**
Pruebas de aceptación: • Codificamos y comprobamos que existe coherencia entre los datos de entrada y salida.		
Observaciones: Esta HU requiere que este desarrollada la función de crear un archivo para poder realizar una codificación previa a su uso.		

Tabla 8. Tarjeta de HU.4: Escribir

Identificador: HU.3	Guardar un archivo de texto	
Descripción: Como usuario quiero guardar un archivo de texto con extensión .odsl en el disco duro para tener acceso a este en cualquier momento.		
Estimación: 4	**Prioridad: 1**	**Entrega: 1**
Pruebas de aceptación: • Guardamos un archivo y verificamos que su extensión sea .odsl • Guardamos un archivo y comprobamos que el archivo quede almacenado en la ruta definida		
Observaciones:		

Tabla 9. Tarjeta de HU.3: Guardar

Identificador: HU.2	Abrir un archivo de texto	
Descripción: Como usuario quiero abrir un archivo de texto con extensión .odsl para ver su contenido		
Estimación: 2	**Prioridad: 2**	**Entrega: 1**
Pruebas de aceptación: • Abrimos un archivo con extensión .odsl y verificamos que el editor muestre su contenido		
Observaciones:		

Tabla 10. Tarjeta de HU.2: Abrir

Identificador: HU.6.1	Copiar texto dentro del archivo	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función copiar del sistema operativo para duplicar datos y almacenarlos durante una corta duración en el portapapeles		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Seleccionamos el segmento de texto a copiar y verificamos que este no sea removido de su origen		
Observaciones: Para verificar la coherencia de los datos copiados, es necesario la HU 6.3		

Tabla 11. Tarjeta de HU.6.1: Copiar

Identificador: HU.6.2	Cortar texto dentro del archivo	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función cortar del sistema operativo para remover datos de su origen y almacenarlos durante una corta duración en el portapapeles		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Seleccionamos el segmento de texto a cortar y verificamos que este fue removido de su origen		
Observaciones: Para verificar la coherencia de los datos cortados, es necesario la HU 6.3		

Tabla 12. Tarjeta de HU.6.2: Cortar

Identificador: HU.6.3	Pegar texto dentro del archivo	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función pegar del sistema operativo para insertar datos almacenados en el portapapeles		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Pegamos un segmento de texto y verificamos que se haya insertado		
Observaciones: Para verificar esta funcionalidad, deben haber datos almacenados en el portapapeles		

Tabla 13. Tarjeta de HU.6.3: Pegar

Identificador: HU.6.4	Borrar contenido del archivo de texto	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función borrar del sistema operativo para eliminar datos del archivo		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Seleccionamos el segmento de texto a borrar y verificamos que fue removido del archivo		
Observaciones:		

Tabla 14. Tarjeta de HU.6.4: Borrar

Identificador: HU.6.5	Seleccionar todo el contenido del archivo	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función seleccionar todo del sistema operativo para realizar funciones de edición sobre dicho texto		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Activamos la función seleccionar todo y verificamos que todo el contenido del archivo se encuentre sombreado		
Observaciones:		

Tabla 15. Tarjeta de HU.6.5: Seleccionar todo

Identificador: HU.6.6	Deshacer la última acción dentro del archivo	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función deshacer del sistema operativo para anular la última acción realizada		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Activamos la función deshacer y comprobamos que el editor vuelve al estado anterior a dicha acción		
Observaciones: Para la validación de esta HU es necesario que previamente se haya realizado acción		

Tabla 16. Tarjeta de HU.6.6: Deshacer

Identificador: HU.6.7	Rehacer la última acción dentro del archivo	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función rehacer del sistema operativo para revertir la última acción anulada		
Estimación: 1/2	**Prioridad: 2**	**Entrega: 2**

Pruebas de aceptación: • Activamos la función rehacer y comprobamos que el editor revierte el último cambio realizado
Observaciones: Para la validación de esta HU es necesario que previamente se haya anulado una acción

Tabla 17. Tarjeta de HU.6.7: Rehacer

Identificador: HU.7	Autocompletar en el archivo de texto	
Descripción: Como usuario quiero autocompletar una palabra reservada del lenguaje odsl en el archivo de texto para una fácil codificación		
Estimación: 5	**Prioridad: 2**	**Entrega: 2**
Pruebas de aceptación: • Codificamos en el lenguaje de dominio específico y verificamos que el editor indica las palabras reservadas del lenguaje.		
Observaciones:		

Tabla 18. Tarjeta de HU.7: Autocompletar

Identificador: HU.5	Eliminar un archivo de texto	
Descripción: Como usuario quiero tener acceso en el menú del editor de texto la función eliminar del sistema operativo para borrar el fichero actual del disco duro		
Estimación: 3	**Prioridad: 3**	**Entrega: 2**
Pruebas de aceptación: • Eliminamos un archivo y verificamos que no se encuentre en el disco duro		
Observaciones:		

Tabla 19. Tarjeta de HU.5: Eliminar

Identificador: HU.8	Buscar dentro del archivo de texto	
Descripción: Como usuario quiero buscar una palabra dentro del archivo texto para encontrarla con facilidad		
Estimación: 3	**Prioridad: 3**	**Entrega: 2**
Pruebas de aceptación: • Buscamos una palabra que se encuentre dentro del archivo y verificamos que se encuentra sombreada • Buscamos una palabra que no se encuentre dentro del archivo y comprobamos que se informa del error		
Observaciones:		

Tabla 20. Tarjeta de HU.8: Buscar

Identificador: HU.9	Reemplazar una palabra dentro del archivo de texto	
Descripción: Como usuario quiero reemplazar una palabra dentro del archivo de texto para una fácil modificación		
Estimación: 3	**Prioridad: 3**	**Entrega: 3**
Pruebas de aceptación: • Buscamos la palabra que deseamos cambiar, insertamos la nueva palabra, reemplazamos y verificamos que se haya realizado el cambio		
Observaciones:		

Tabla 21. Tarjeta de HU.9: Reemplazar

8.2 PLAN DE ITERACIONES

8.2.1 Plan iteración No. 1 entrega No. 1

1. Objetivos de la iteración

El objetivo que se ha diseñado para esta primera iteración es permitirle al usuario crear un archivo de texto en blanco y escribir en el archivo de texto utilizando el lenguaje de dominio específico.

2. Listado inicial de historias de usuario a desarrollar

La lista de las Historias de usuario que se van a desarrollar en esta iteración son:

Ident	Título	Est
HU.1	Un usuario puede crear un nuevo archivo de texto	3
HU.4	Un usuario puede escribir en un archivo de texto	6

Tabla 22. HU para la iteración No.1 Entrega No.1

3. Descomposición en Tareas de Desarrollo

Se incluye el resultado de la descomposición de las historias de usuario en tareas de desarrollo, así como la asignación a los desarrolladores y la estimación realizada de su duración.

Identificador: HU.1	Crear un nuevo archivo de texto	3

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 1-1	Diseñar los diagramas UML	2	Desarrollador 1
Tarea 1-2	Diseñar e implementar la interfaz gráfica del editor para crear el archivo.	4	Desarrollador 2

Observaciones:

Tabla 23. Tareas de desarrollo para la HU.1

Identificador: HU.4	Escribir un nuevo archivo de texto	6

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 4-1	Diseñar los diagramas UML	1	Desarrollador 1
Tarea 4-2	Diseñar e implementar la interfaz gráfica que le permita al usuario escribir	3	Desarrollador 2

Observaciones:

Tabla 24. Tareas de desarrollo para la HU.4

4. Carga prevista en los desarrolladores

Información final sobre la carga prevista de trabajo de cada uno de los miembros del equipo de desarrollo en base a las tareas asignadas en la iteración.

Desarrollador	Velocidad Inicial (días ideales)	Dedicación (% de tiempo)	Carga de trabajo (días ideales)	Tareas Aceptadas (cantidad)
Desarrollador 1	10	100%	10	4
Desarrollador 2	10	100%	10	4

Tabla 25. Carga para los desarrolladores en la Iteración No.1

5. Planificación temporal de la iteración.

La planificación temporal que se ha utilizado para hacer las estimaciones es la siguiente:

Semana	Desarrollador	Día 1	Día 2	Día 3	Día 4	Día 5
Semana 1	Desarrollador 1	Tarea 1-1	Tarea 1-1	Tarea 1-1	Tarea 1-1	Tarea 1-1
	Desarrollador 2	Tarea 1-2	Tarea 1-2	Tarea 1-2	Tarea 1-2	Tarea 1-2
Semana 2	Desarrollador 1	Tarea 4-1	Tarea 4-1	Tarea 4-1	Tarea 4-1	Tarea 4-1
	Desarrollador 2	Tarea 4-2	Tarea 4-2	Tarea 4-2	Tarea 4-2	Tarea 4-2

Tabla 26. Planificación temporal de la iteración No.1

8.2.1.1 Desarrollo de las tareas

Para el diseño de los diagramas UML hemos tenido en cuenta solo aquellos que nos permitan visualizar, especificar, construir y documentar el editor de texto, en cada iteración.

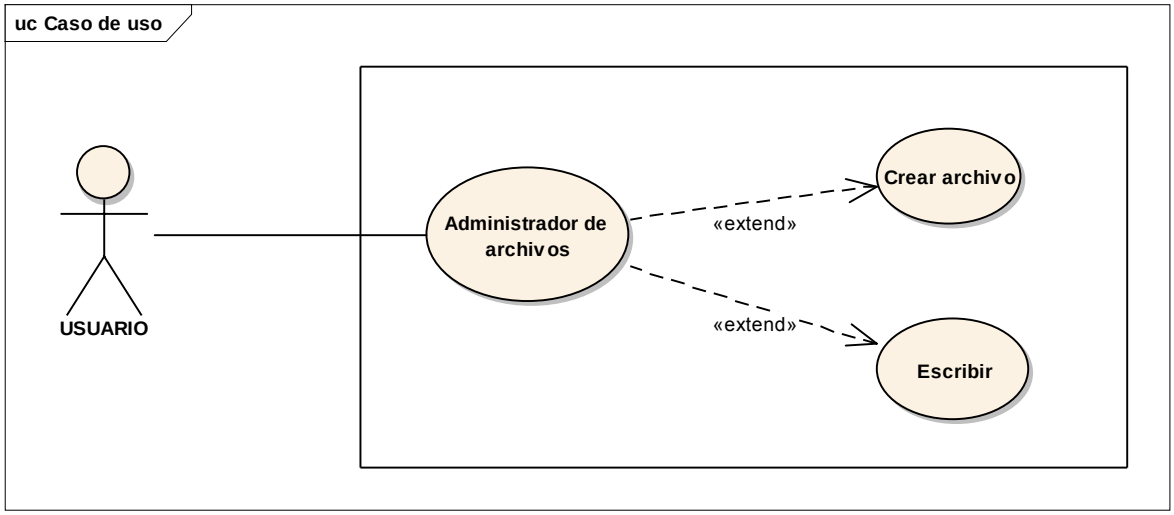

Ilustración 1. Diagrama de casos de uso iteración 1

CU-001	**Crear archivo de texto**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	Ninguna
Precondición	Ninguna
Descripción	El usuario crea un nuevo archivo de texto en blanco para codificar en base al lenguaje de dominio específico para problemas de optimización lineal.
Referencias	HU.1
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario solicita al sistema crear un nuevo archivo de texto.	**2.** El sistema despliega un lienzo en blanco dentro de la ventana del editor.
Postcondición	Se creó un nuevo fichero de texto en blanco.
Comentarios	

Tabla 27. CU: Crear archivo de texto

CU-002	**Escribir en un archivo de texto**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1
Precondición	
Descripción	El usuario escribe dentro de un archivo de texto para programar un caso de estudio de optimización lineal.
Referencias	HU.2
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario escribe en el lienzo del archivo de texto.	**2.** Hace visible en el lienzo del editor lo escrito por el usuario.
Postcondición	
Comentarios	

Tabla 28. CU: Escribir archivo de texto

Diagrama de clases

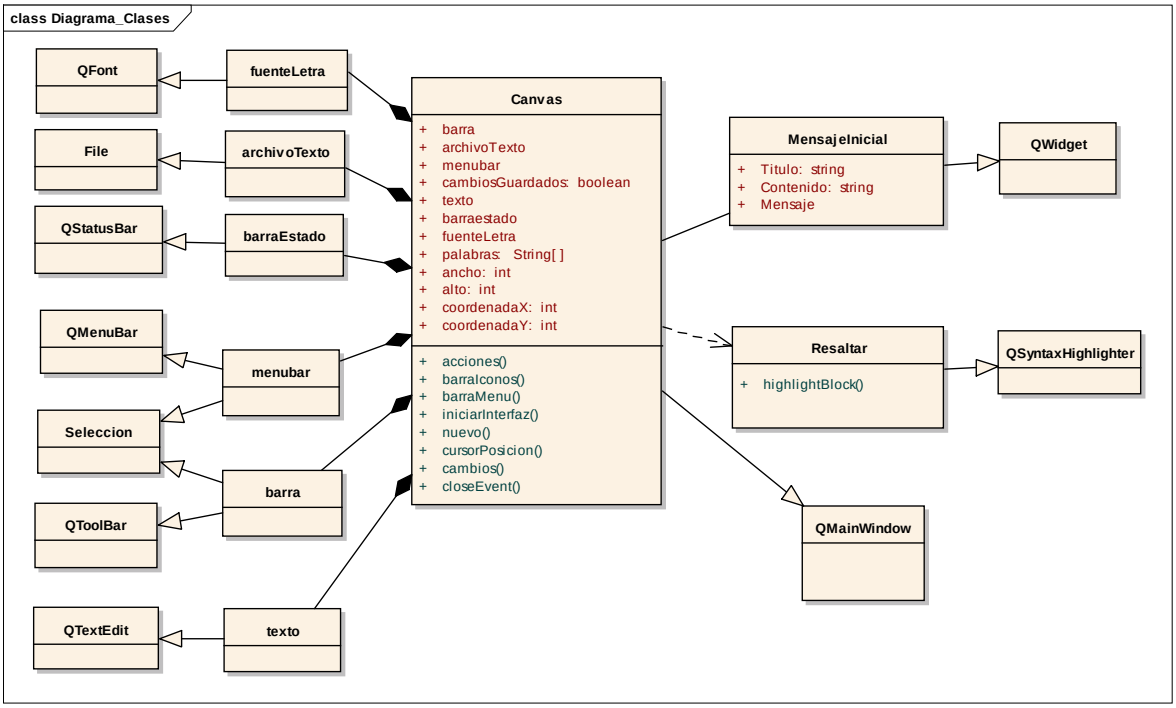

Ilustración 2. Diagrama de clases Iteración 1 Entrega No1

Diagrama de secuencia

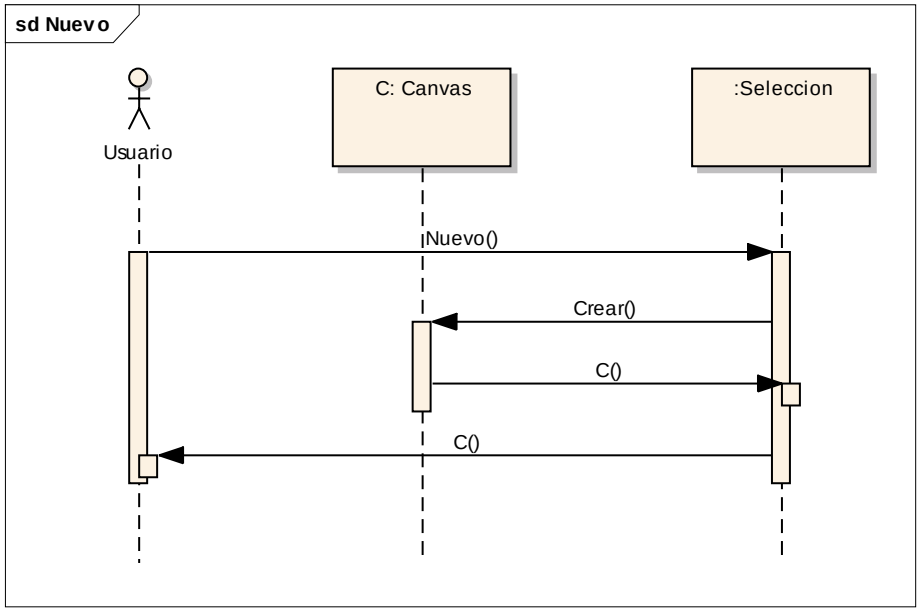

Ilustración 3. Diagrama de Secuencia: Crear archivo

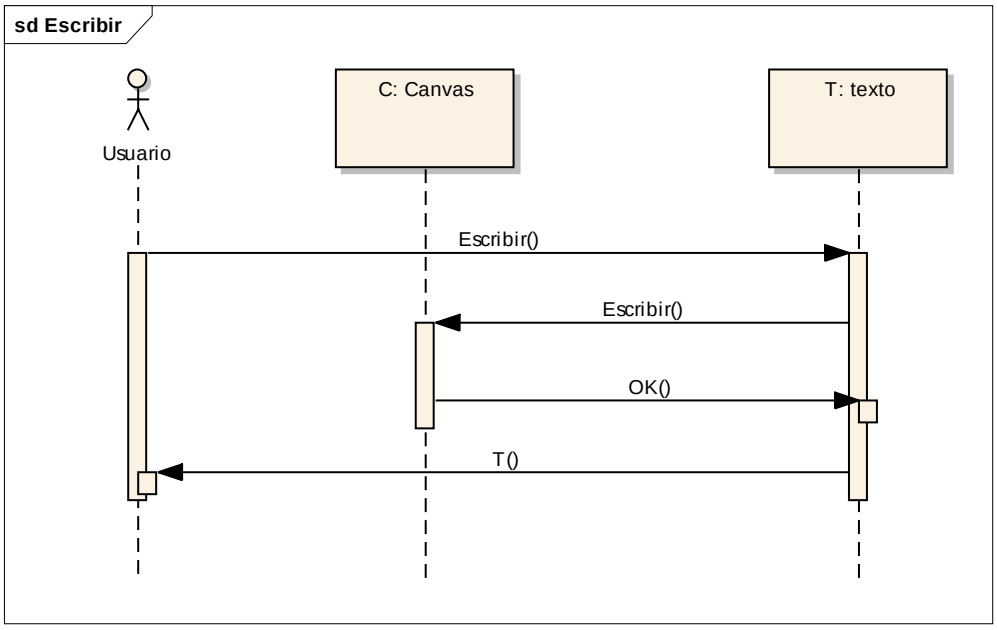

Ilustración 4. Diagrama de Secuencia: Escribir

Diagrama de colaboración

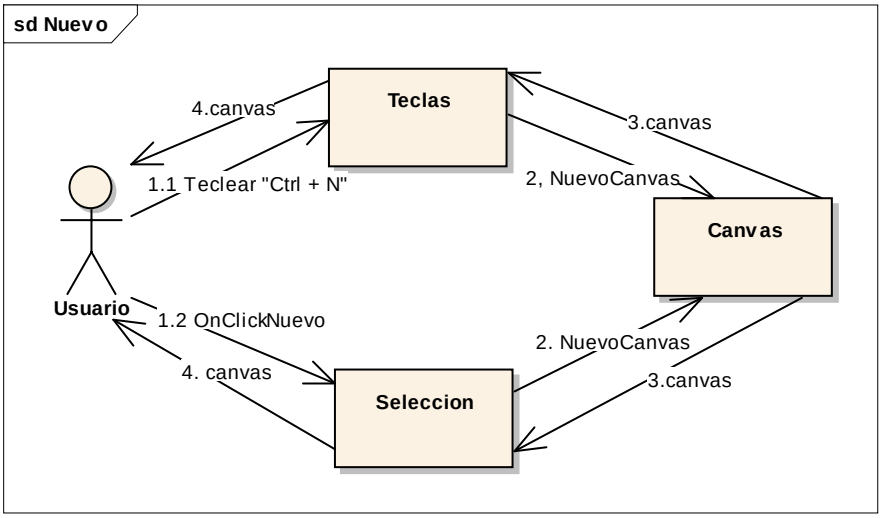

Ilustración 5. Diagrama de colaboración: Crear Archivo

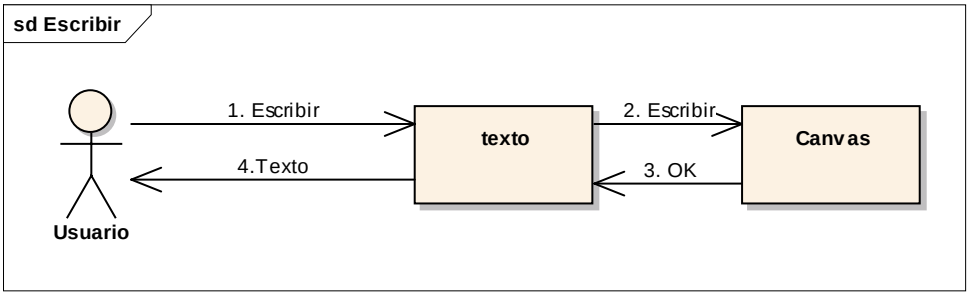

Ilustración 6. Diagrama de colaboración: Escribir

Diagrama de estados

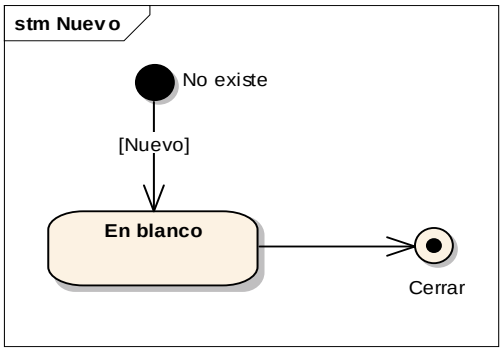

Ilustración 7. Diagrama de estado: Nuevo

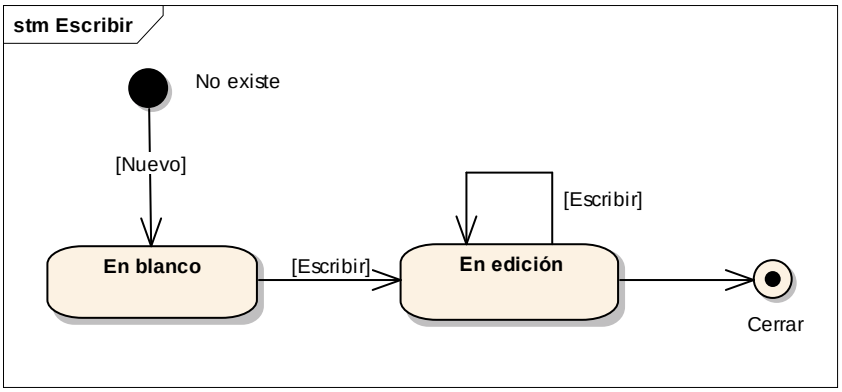

Ilustración 8. Diagrama de estado: Escribir

Diagrama de actividades

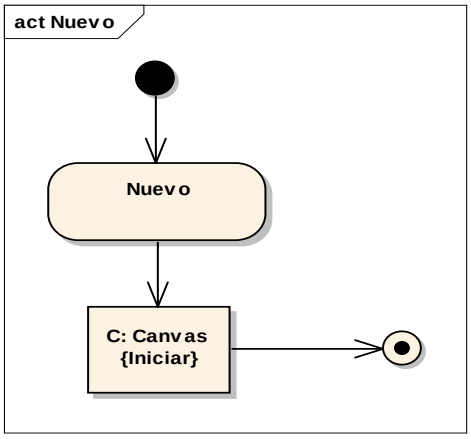

Ilustración 9. Diagrama de Actividades Nuevo

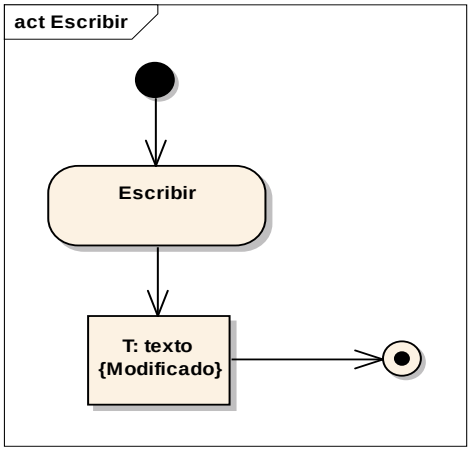

Ilustración 10. Diagrama de actividades: Escribir

8.2.1.1.1 Interfaz Hombre-Maquina

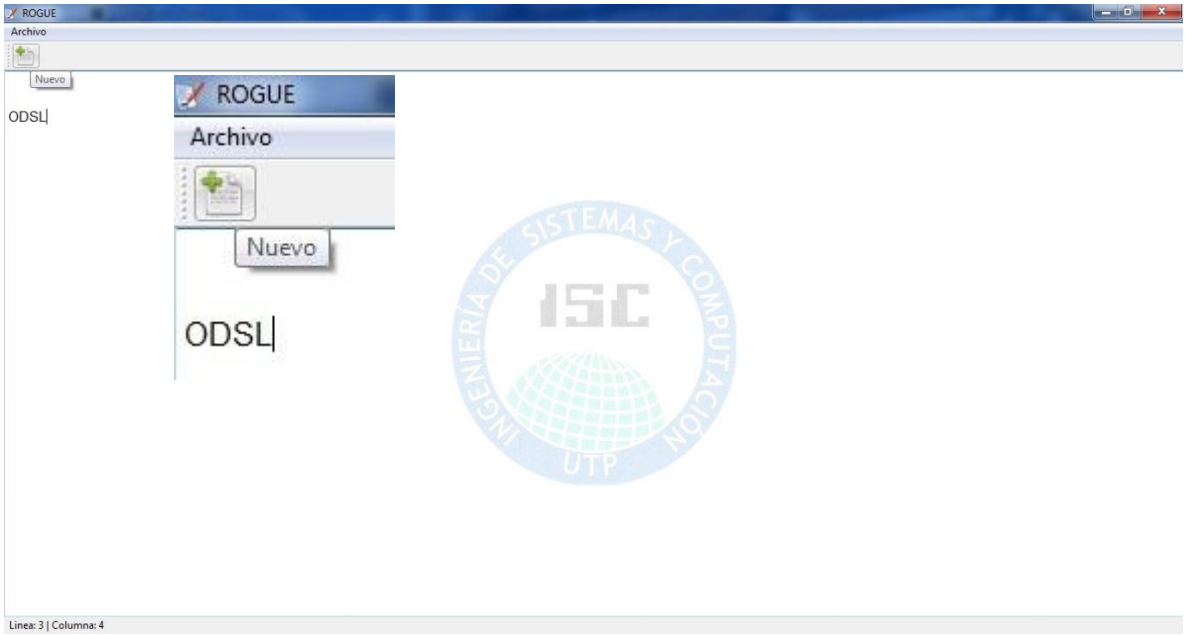

Ilustración 11. Crear nuevo archivo

8.2.1.2 Revisión del SPRINT

Se cumple con el objetivo de permitirle al usuario crear un archivo de texto en blanco y escribir en el archivo de texto utilizando el lenguaje de dominio específico.

Se cumple con las pruebas de aceptación de cada una de las HU:

- Creamos un nuevo archivo y verificamos que se muestra un lienzo en blanco.
- Codificamos y comprobamos que existe coherencia entre los datos de entrada y salida

8.2.1.3 Retrospectiva del SPRINT

¿Qué salió bien en la iteración?

Se cumplió con el objetivo planteado inicialmente, el editor cuenta con la funcionalidad de crear un nuevo archivo de texto y le permite al usuario escribir sobre el lienzo.

Valor agregado

Además de cumplir con el objetivo principal, el editor cuenta con el conteo de líneas y columnas, también resalta las palabras reservadas del lenguaje.

¿Qué no salió bien en la iteración? (errores)

Aunque se cumplieron con los objetivos de la iteración, se requirió de tiempo adicional para aprender a utilizar las herramientas del lenguaje de programación, dado que se tenía poco conocimiento previo.

¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua).

Para las próximas iteraciones es importante documentar la codificación del proyecto.

8.2.2 Plan iteración No. 2 entrega No. 1

1. Objetivos de la iteración

El objetivo que se ha diseñado para esta segunda iteración de la primera entrega es permitirle al usuario guardar y abrir un archivo con extensión .odsl para que pueda tener, asegurar y leer sus archivos de texto respectivamente.

2. Listado inicial de historias de usuario a desarrollar

La lista de las Historias de usuario que se van a desarrollar en esta iteración son:

Ident	Título	Est
HU.3	Un usuario puede guardar un archivo	4
HU.2	Un usuario puede abrir un archivo de texto	4

Tabla 29. HU para la iteración No.2 Entrega No.1

3. Descomposición en Tareas de Desarrollo

Se incluye el resultado de la descomposición de las historias de usuario en tareas de desarrollo, así como la asignación a los desarrolladores y la estimación realizada de su duración.

Identificador: HU.3	Guardar un nuevo archivo de texto	4

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 3-1	Diseñar los diagramas UML	2	Desarrollador 2
Tarea 3-2	Diseñar e implementar la interfaz gráfica del editor para guardar el archivo de texto.	4	Desarrollador 1

Observaciones:

Tabla 30. Tareas de desarrollo para la HU.3

Identificador: HU.2	Abrir un nuevo archivo de texto	4

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 2-1	Diseñar los diagramas UML	1	Desarrollador 2
Tarea 2-2	Diseñar e implementar la interfaz gráfica del editor para abrir un archivo de texto.	3	Desarrollador 1

Observaciones:

Tabla 31. Tareas de desarrollo para la HU.2

4. Carga prevista en los desarrolladores

Información final sobre la carga prevista de trabajo de cada uno de los miembros del equipo de desarrollo en base a las tareas asignadas en la iteración.

Desarrollador	Velocidad Inicial (días ideales)	Dedicación (% de tiempo)	Carga de trabajo (días ideales)	Tareas Aceptadas (cantidad)
Desarrollador 1	20	100%	10	4
Desarrollador 2	20	100%	10	4

Tabla 32. Carga para los desarrolladores en la Iteración No.2

5. Planificación temporal de la iteración.

La planificación temporal que se ha utilizado para hacer las estimaciones es la siguiente:

Semana	Desarrollador	Día 1	Día 2	Día 3	Día 4	Día 5
Semana 1	Desarrollador 1	Tarea 3-1	Tarea 3-1	Tarea 3-1	Tarea 3-1	Tarea 3-1
	Desarrollador 2	Tarea 3-2	Tarea 3-2	Tarea 3-2	Tarea 3-2	Tarea 3-2
Semana 2	Desarrollador 1	Tarea 2-1	Tarea 2-1	Tarea 2-1	Tarea 2-1	Tarea 2-1
	Desarrollador 2	Tarea 2-2	Tarea 2-2	Tarea 2-2	Tarea 2-2	Tarea 2-2

Tabla 33. Planificación temporal de la iteración No.2

8.2.2.1 Desarrollo de las tareas de la iteración 2

Para el diseño de los diagramas UML hemos tenido en cuenta solo aquellos que nos permitan visualizar, especificar, construir y documentar el editor de texto, en cada iteración.

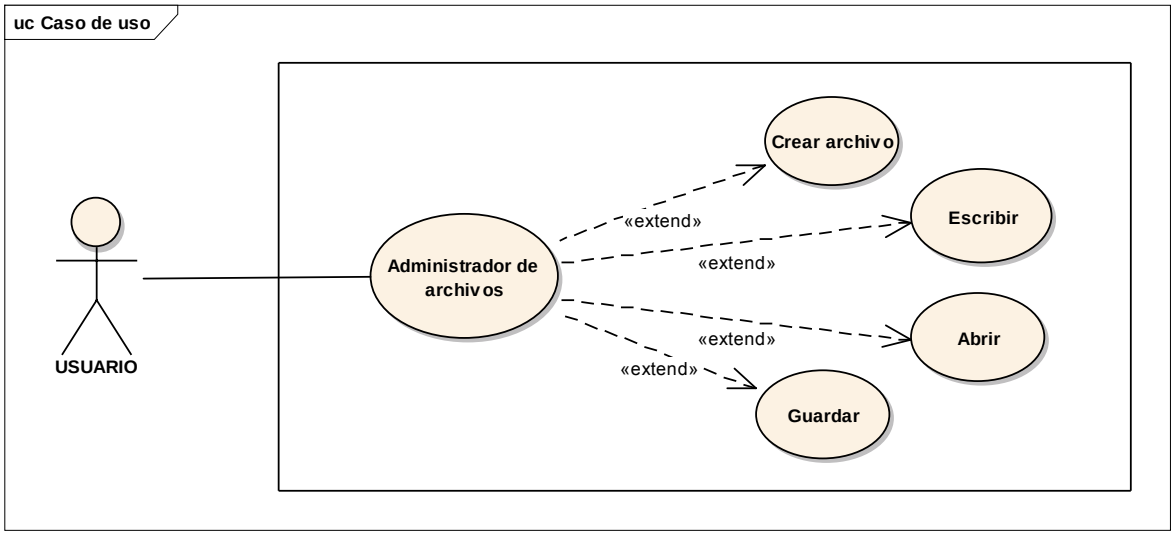

Ilustración 12. Diagrama de casos de uso iteración 2

CU-003	**Guardar fichero de texto**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1
Precondición	El lienzo dentro de la ventana del editor tenga texto.
Descripción	El usuario guarda un archivo de texto con extensión .odsl en el disco duro para tener acceso a este en cualquier momento.
Referencias	HU.3
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario escribe un texto sobre el lienzo del editor. **2.** El usuario Selecciona la opción guardar.	**3.** Se almacena el contenido del lienzo en un archivo del disco duro
Postcondición	Se guardó un nuevo fichero de texto.
Comentarios	

Tabla 34. CU: Guardar fichero de texto

CU-004	**Abrir archivo de texto**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.3
Precondición	El usuario debe haber creado previamente un archivo de texto con extensión específica (odsl) dentro del disco.
Descripción	El usuario abre un archivo de texto con extensión .odsl para ver su contenido
Referencias	HU.4
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario selecciona la opción abrir.	**2.** Abre un nuevo lienzo con archivo indicado por el usuario.
Postcondición	Se abre el archivo de texto que especifica el usuario.
Comentarios	

Tabla 35. CU: Abrir en un archivo de texto

Diagrama de clases

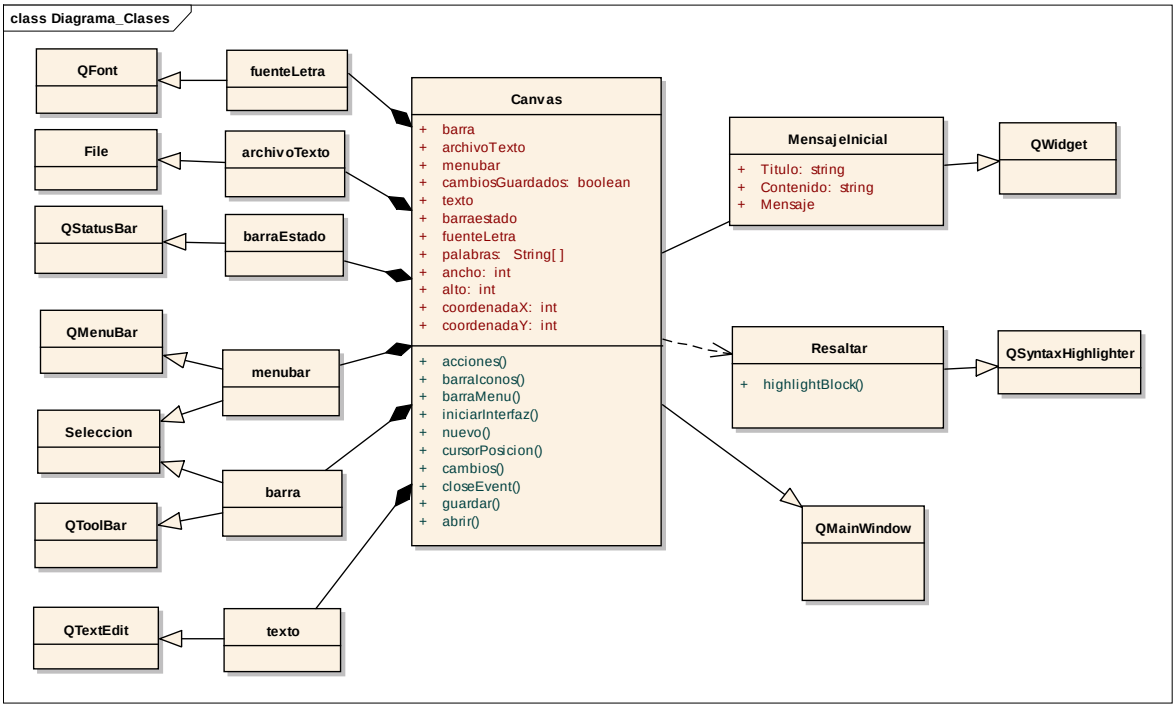

Ilustración 13. Diagrama de clases iteración 2 Entrega No1

Diagrama de secuencia

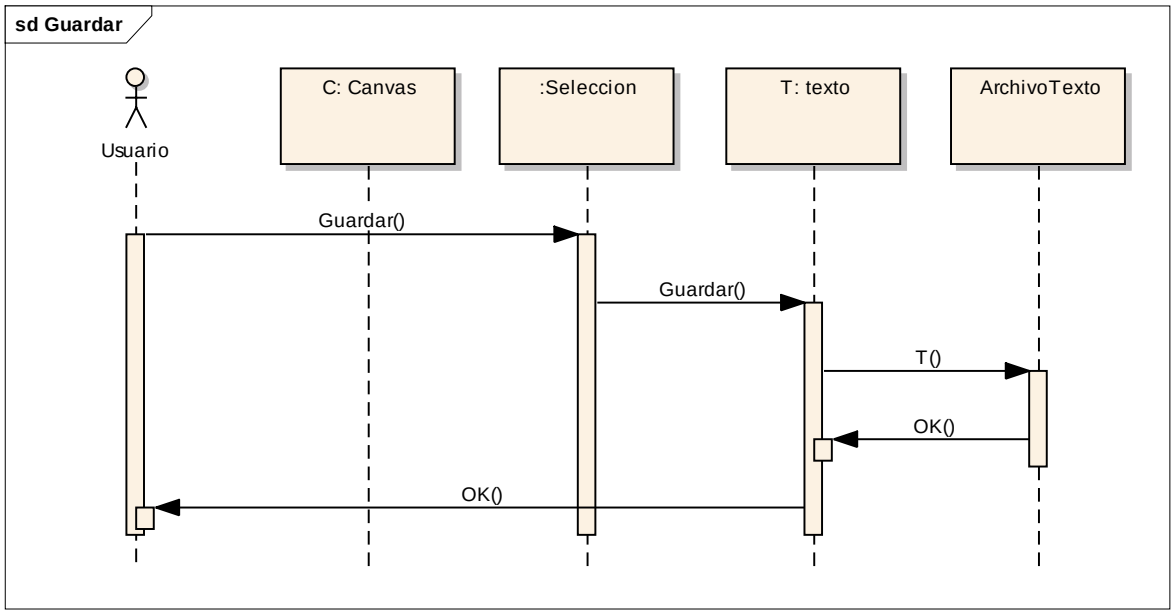

Ilustración 14. Diagrama de secuencia: Guardar archivo

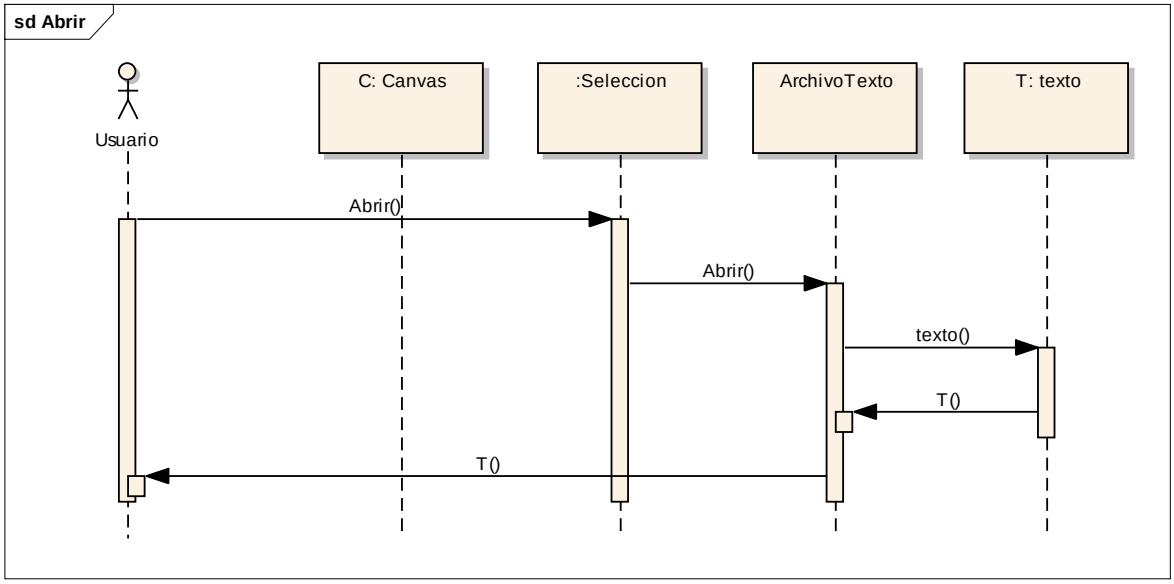

Ilustración 15. Diagrama de secuencia: Abrir archivo

Diagrama de colaboración

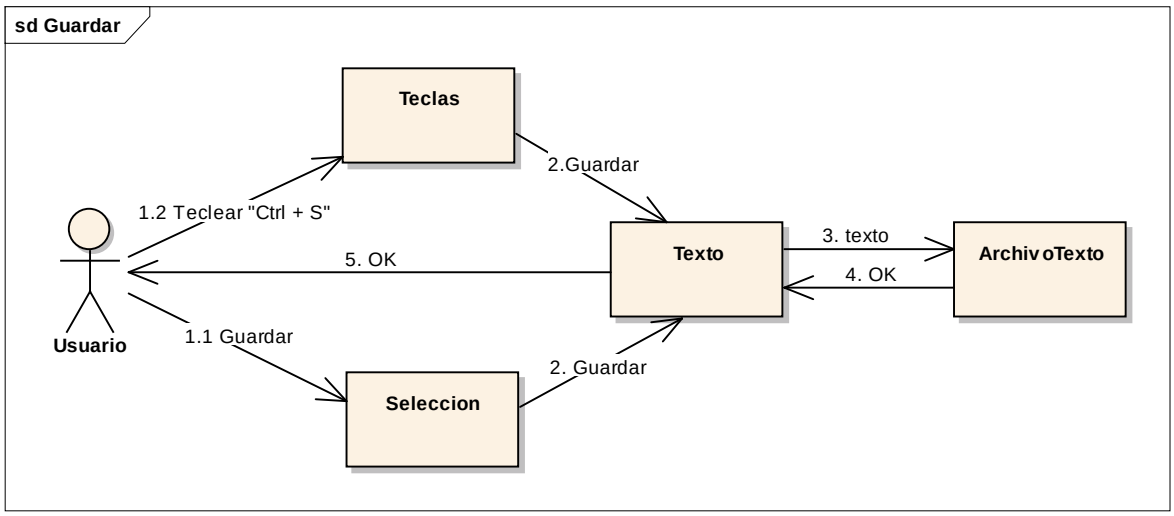

Ilustración 16. Diagrama de colaboración: Guardar

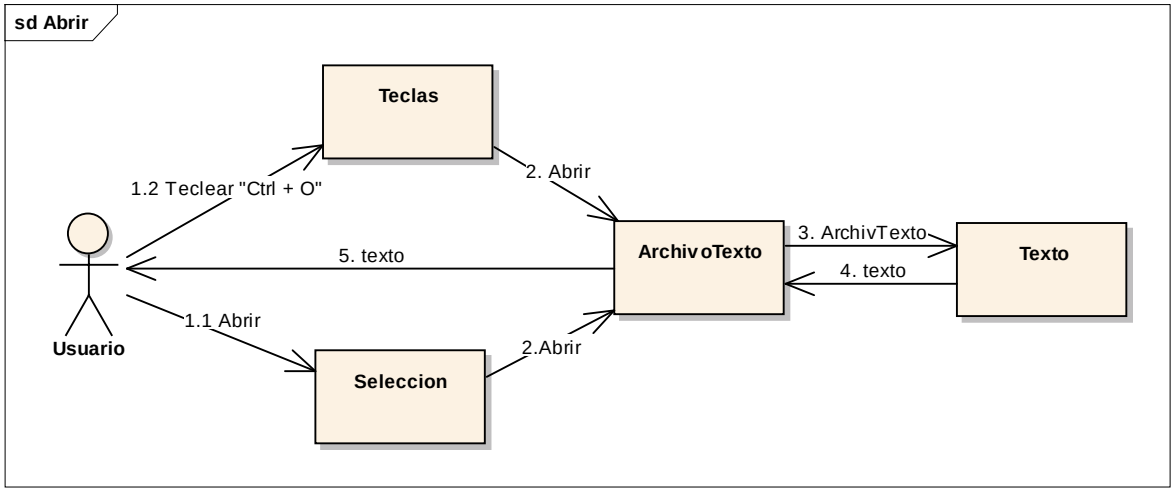

Ilustración 17. Diagrama de colaboración: Abrir

Diagrama de estados

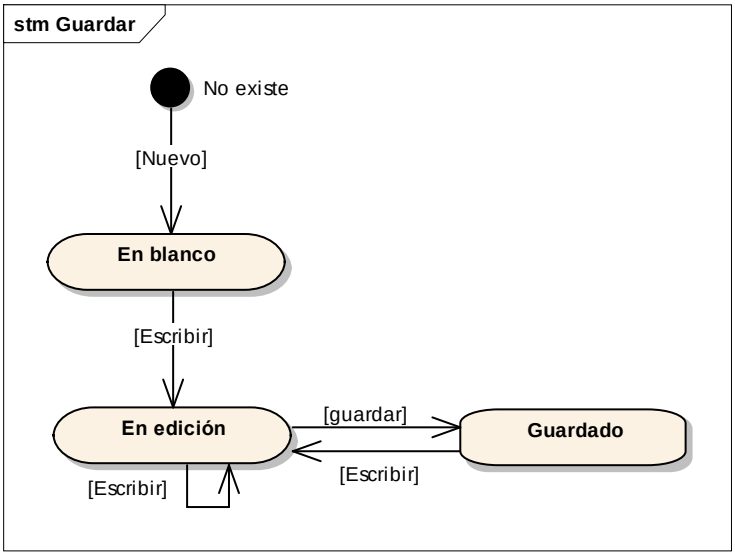

Ilustración 18. Diagrama de estado: Guardar

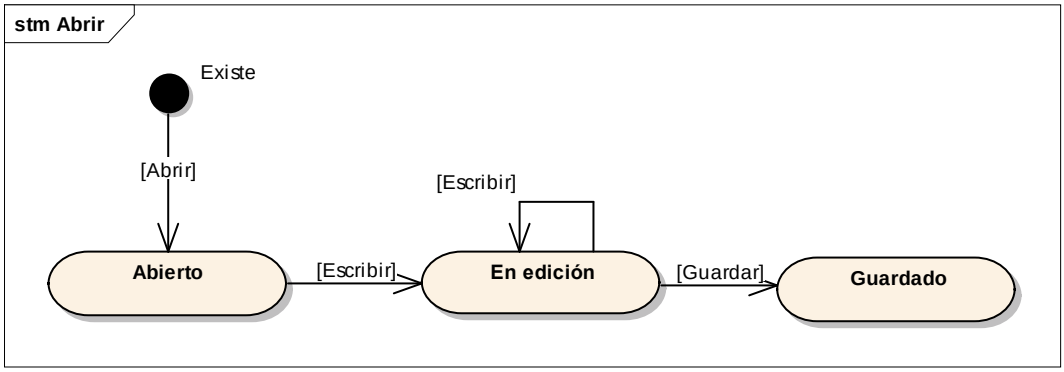

Ilustración 19. Diagrama de estado: Abrir

Diagrama de actividades

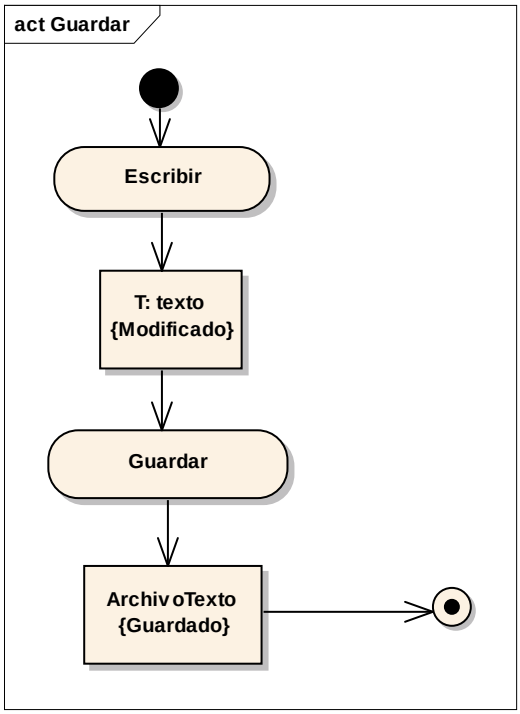

Ilustración 20. Diagrama de actividades: Guardar

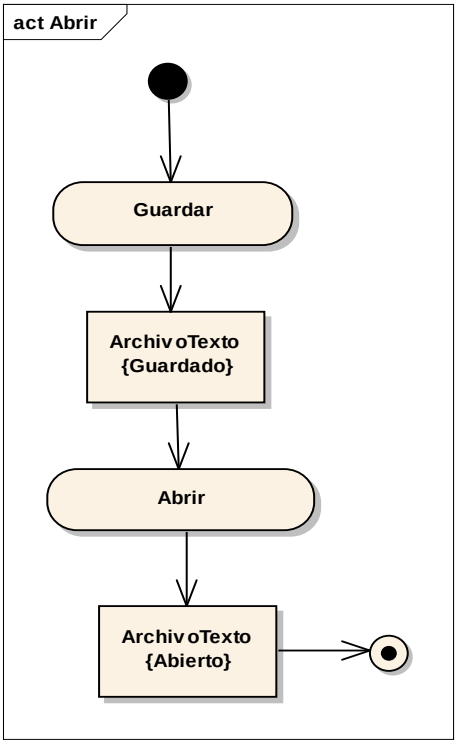

Ilustración 21. Diagrama de actividades: Abrir

8.2.2.1.1 Interfaz Hombre-Maquina

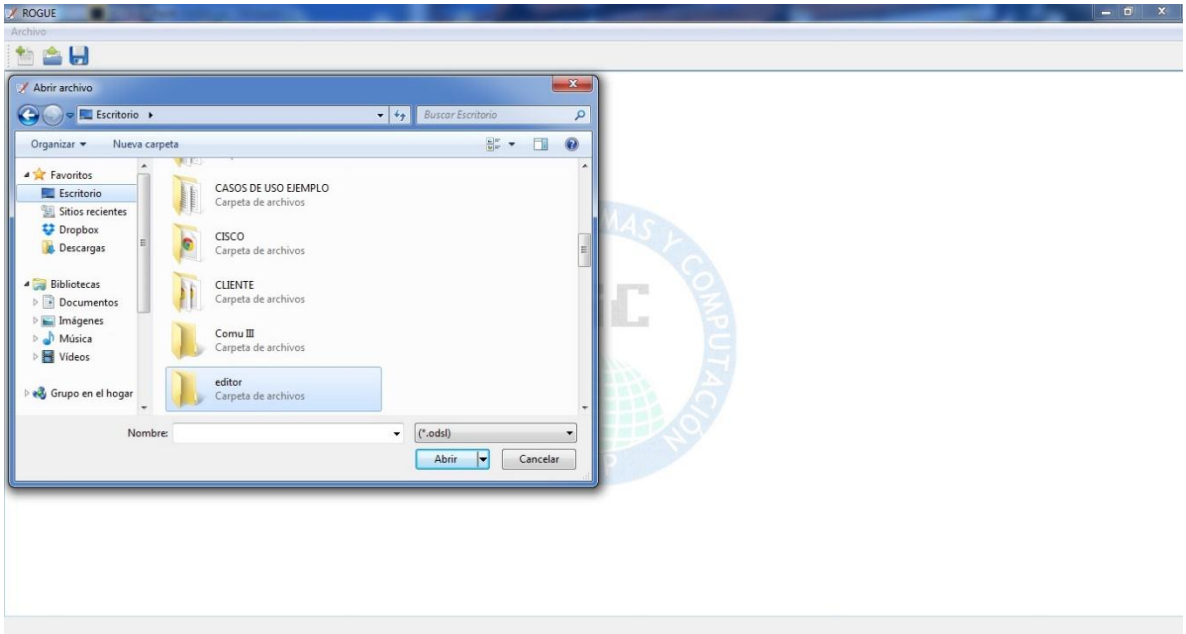

Ilustración 22. Abrir archivo

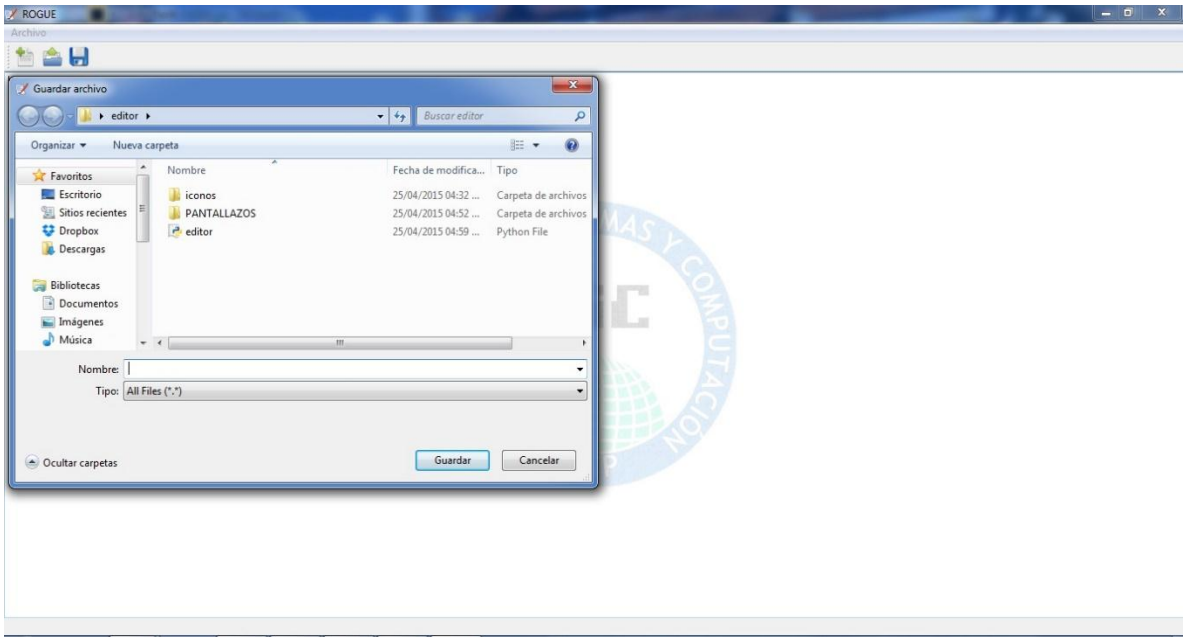

Ilustración 23. Guardar archivo

8.2.2.2 Revisión del SPRINT

Se cumple el objetivo de permitirle al usuario guardar y abrir un archivo con extensión .odsl para que pueda tener, asegurar y leer sus archivos de texto respectivamente.

Se cumple con las pruebas de aceptación de cada una de las HU:

- Guardamos un archivo y verificamos que su extensión sea .odsl
- Guardamos un archivo y comprobamos que el archivo quede almacenado en la ruta definida
- Abrimos un archivo con extensión .odsl y verificamos que el editor muestre su contenido

8.2.2.3 Retrospectiva del SPRINT

¿Qué salió bien en la iteración?

Se cumplió con el objetivo planteado inicialmente, el editor cuenta con la funcionalidad de guardar un archivo de texto y le permite al usuario abrir el archivo guardado.

Valor agregado

El editor cuenta con un diseño intuitivo para que el usuario se sienta cómodo, este puede ver el nombre del archivo en la parte superior del editor.

¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua).

Para las próximas iteraciones es importante diseñar un editor de texto intuitivo, darle un nombre y crear un logo representativo.

8.2.3 Plan iteración No. 1 entrega No. 2

1. Objetivos de la iteración

El objetivo que se ha diseñado para esta primera iteración de la segunda entrega es permitir al usuario realizar funciones de edición a sus archivos tales como copiar, cortar, pegar, seleccionar, borrar, rehacer y deshacer. Además permite autocompletar dentro del archivo palabras reservadas del lenguaje de dominio específico para problemas de optimización lineal.

2. Listado inicial de historias de usuario a desarrollar

La lista de las Historias de usuario que se van a desarrollar en esta iteración son:

Ident.	Título	Est
HU.6	Un usuario puede realizar funciones básicas edición	--
HU.6.1	Un usuario puede copiar texto dentro del archivo	½
HU.6.2	Un usuario puede cortar texto dentro del archivo	½
HU.6.3	Un usuario puede pegar texto dentro del archivo	½
HU.6.4	Un usuario puede borrar contenido del archivo de texto	½
HU.6.5	Un usuario puede seleccionar todo el contenido del archivo	½
HU.6.6	Un usuario puede deshacer la última acción dentro del archivo	½
HU.6.7	Un usuario puede rehacer la última acción dentro del archivo	½
HU.7	Un usuario puede autocompletar en el archivo de texto	5

Tabla 36. HU para la iteración No.1 Entrega No.2

3. Descomposición en Tareas de Desarrollo

Se incluye el resultado de la descomposición de las historias de usuario en tareas de desarrollo, así como la asignación a los desarrolladores y la estimación realizada de su duración.

Identificador: HU.6.1	Copiar texto dentro del archivo	½

Identificador	**Título de la tarea de desarrollo**	**Estimación (días Ideales)**	**Desarrollador Asignado**
Tarea 6.1-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.1-2	Diseñar e implementar la interfaz gráfica del editor para copiar en el archivo.	1/2	Desarrollador 2

Observaciones:

Tabla 37. Tareas de desarrollo para la HU.6.1

Identificador: HU.6.2	Cortar texto dentro del archivo	½

Identificador	**Título de la tarea de desarrollo**	**Estimación (días Ideales)**	**Desarrollador Asignado**
Tarea 6.2-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.2-2	Diseñar e implementar la interfaz gráfica del editor para cortar en el archivo.	1/2	Desarrollador 2

Observaciones:

Tabla 38. Tareas de desarrollo para la HU.6.2

Identificador: HU.6.3	Pegar texto dentro del archivo	½

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 6.3-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.3-2	Diseñar e implementar la interfaz gráfica del editor para pegar en el archivo.	1/2	Desarrollador 2

Observaciones:

Tabla 39. Tareas de desarrollo para la HU.6.3

Identificador: HU.6.4	Borrar contenido del archivo de texto	½

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 6.4-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.4-2	Diseñar e implementar la interfaz gráfica del editor para copiar en el archivo.	1/2	Desarrollador 2

Observaciones:

Tabla 40. Tareas de desarrollo para la HU.6.4

Identificador: HU.6.5	Seleccionar todo el contenido del archivo	½

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 6.5-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.5-2	Diseñar e implementar la interfaz gráfica del editor para	1/2	Desarrollador 2

	seleccionar todo el contenido del archivo.		

Observaciones:

Tabla 41. Tareas de desarrollo para la HU.6.5

Identificador: HU.6.6	Deshacer la última acción dentro del archivo	½

Identificador	**Título de la tarea de desarrollo**	**Estimación (días Ideales)**	**Desarrollador Asignado**
Tarea 6.6-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.6-2	Diseñar e implementar la interfaz gráfica del editor para deshacer la última acción dentro del archivo.	1/2	Desarrollador 2

Observaciones:

Tabla 42. Tareas de desarrollo para la HU.6.6

Identificador: HU.6.7	Rehacer la última acción dentro del archivo	½

Identificador	**Título de la tarea de desarrollo**	**Estimación (días Ideales)**	**Desarrollador Asignado**
Tarea 6.7-1	Diseñar los diagramas UML	1/2	Desarrollador 1
Tarea 6.7-2	Diseñar e implementar la interfaz gráfica del editor para Rehacer la última acción dentro del archivo.	1/2	Desarrollador 2

Observaciones:

Tabla 43. Tareas de desarrollo para la HU.6.7

Identificador: HU.7	Autocompletar en el archivo de texto	5

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 7-1	Diseñar los diagramas UML	3	Desarrollador 1
Tarea 7-2	Diseñar e implementar la interfaz gráfica del editor para autocompletar en el archivo de texto.	6	Desarrollador 2

Observaciones:

Tabla 44. Tareas de desarrollo para la HU.7

4. Carga prevista en los desarrolladores

Información final sobre la carga prevista de trabajo de cada uno de los miembros del equipo de desarrollo en base a las tareas asignadas en la iteración.

Desarrollador	Velocidad Inicial (días ideales)	Dedicación (% de tiempo)	Carga de trabajo (días ideales)	Tareas Aceptadas (cantidad)
Desarrollador 1	20	100%	10	4
Desarrollador 2	20	100%	10	4

Tabla 45. Carga para los desarrolladores en la Iteración No.1

5. Planificación temporal de la iteración.

La planificación temporal que se ha utilizado para hacer las estimaciones es la siguiente:

Semana	Desarrollador	Día 1	Día 2	Día 3	Día 4	Día 5
Semana 1	Desarrollador 1	Tarea 6.1-1	Tarea 6.3-1	Tarea 6.5-1	Tarea 6.7-1	Tarea 7-1

		Tarea 6.2-1	Tarea 6.4-1	Tarea 6.6-1		
	Desarrollador 2	Tarea 6.1-2 Tarea 6.2-2	Tarea 6.3-2 Tarea 6.4-2	Tarea 6.5-2 Tarea 6.6-2	Tarea 6.7-2	Tarea 7-2
Semana 2	Desarrollador 1	Tarea 7-1	Tarea 7-1	Tarea 7-1		
	Desarrollador 2	Tarea 7-2	Tarea 7-2	Tarea 7-2	Tarea 7-2	Tarea 7-2

Tabla 46. Planificación temporal de la iteración No.1

8.2.3.1 Desarrollo de las tareas

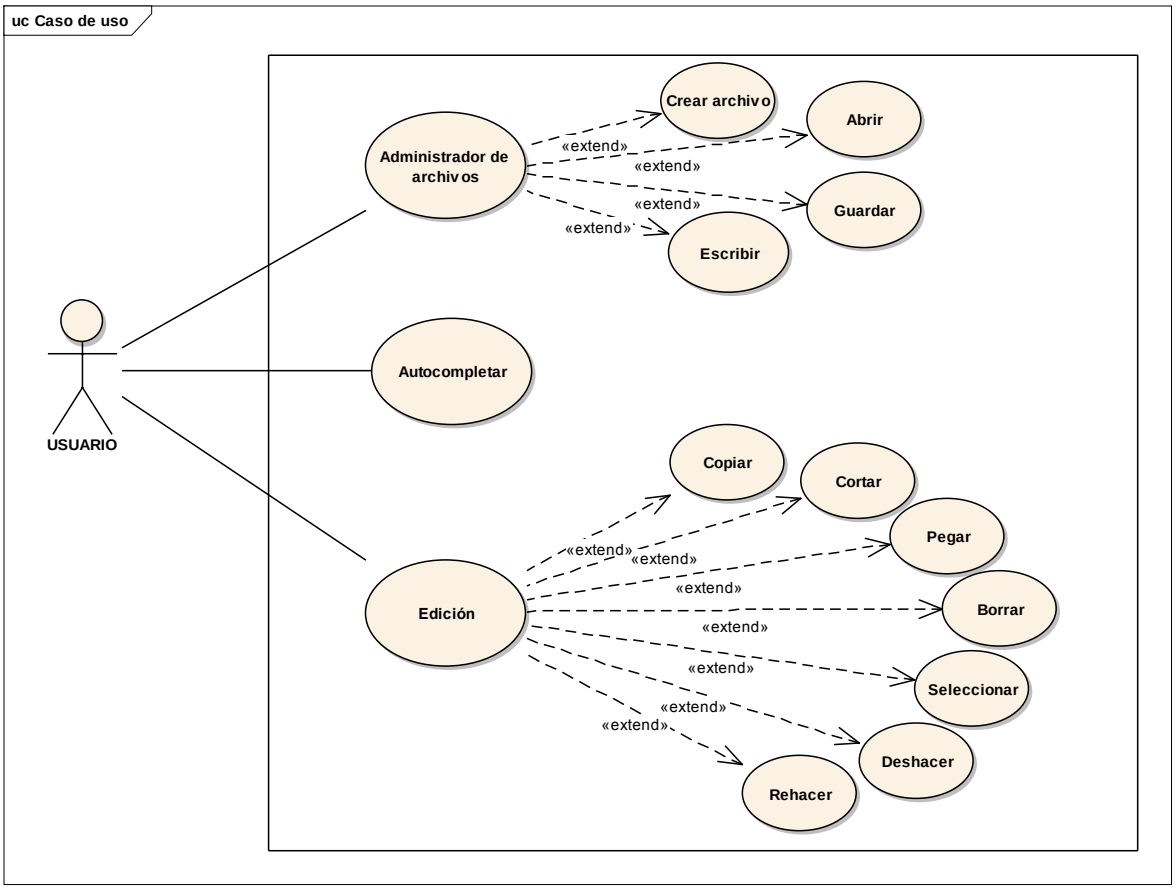

Ilustración 24. Diagrama de casos de uso iteración 3.

CU-006	**Realizar edición**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1 HU.2
Precondición	El usuario debe tener texto dentro del archivo o almacenado en el portapapeles.
Descripción	El usuario puede tener acceso en el menú del editor de texto a las funcione copiar, cortar, pegar, borrar, seleccionar,

	deshacer y rehacer del sistema operativo para realizar la edición que desee.
Referencias	HU.6.1, HU.6.2, HU.6.3, HU.6.4, HU.6.5, HU.6.6, HU.6.7
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario solicita al sistema cualquiera de las funciones básicas de edición tales como copiar, cortar, pegar, borrar, seleccionar, deshacer y rehacer en el archivo de texto.	**2.** El sistema realizar la función según la opción seleccionada.
Postcondición	Se edita el archivo de texto
Comentarios	

Tabla 47. CU: Realizar edición

CU-007	**Autocompletar**
Versión	1.0 (23/08/2014)
Actores	Usuario común
Dependencias	HU.1 HU.2
Precondición	Ingresar Ctrl+E para ver el listado de las palabras reservadas.
Descripción	El usuario puede autocompletar una palabra reservada del lenguaje odsl en el archivo de texto para una fácil codificación.
Referencias	
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. Ingresar Ctrl + E	**2.** Listar las palabras reservadas del lenguaje.

Postcondición	Se autocompleta la palabra acorde a las palabras reservadas del lenguaje.
Comentarios	

Tabla 48. CU: Autocompletar

Diagrama de clases

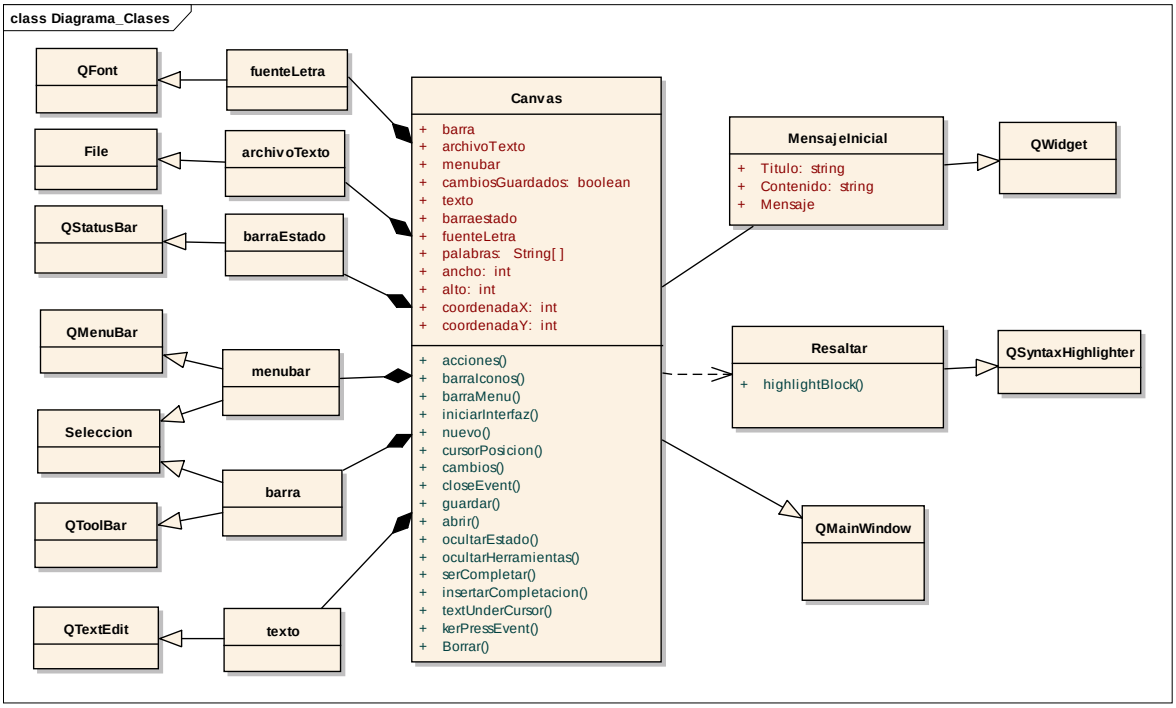

Ilustración 25. Diagrama de clases iteración 1 Entrega No2

Diagrama de secuencia

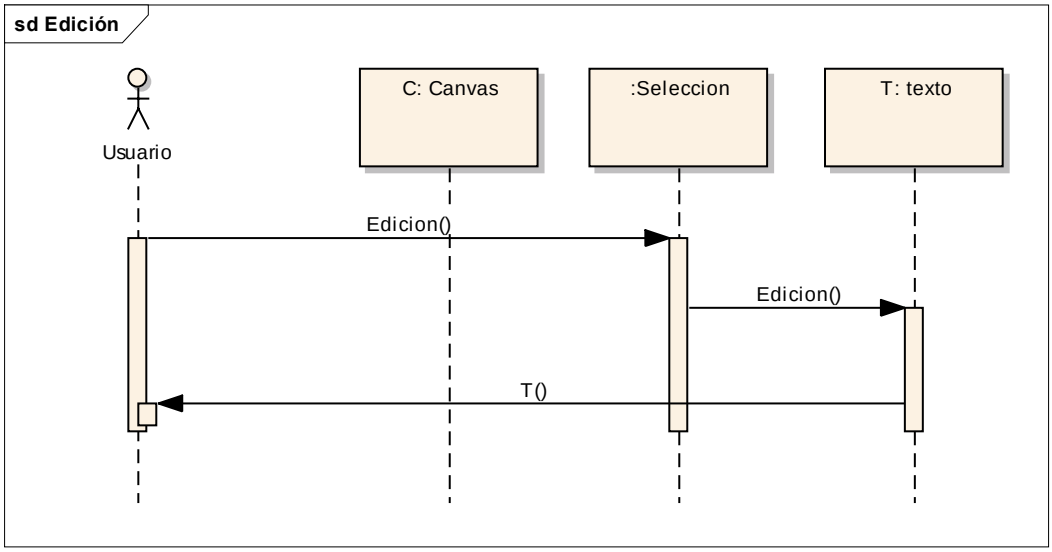

Ilustración 26. Diagrama de secuencia: Edición

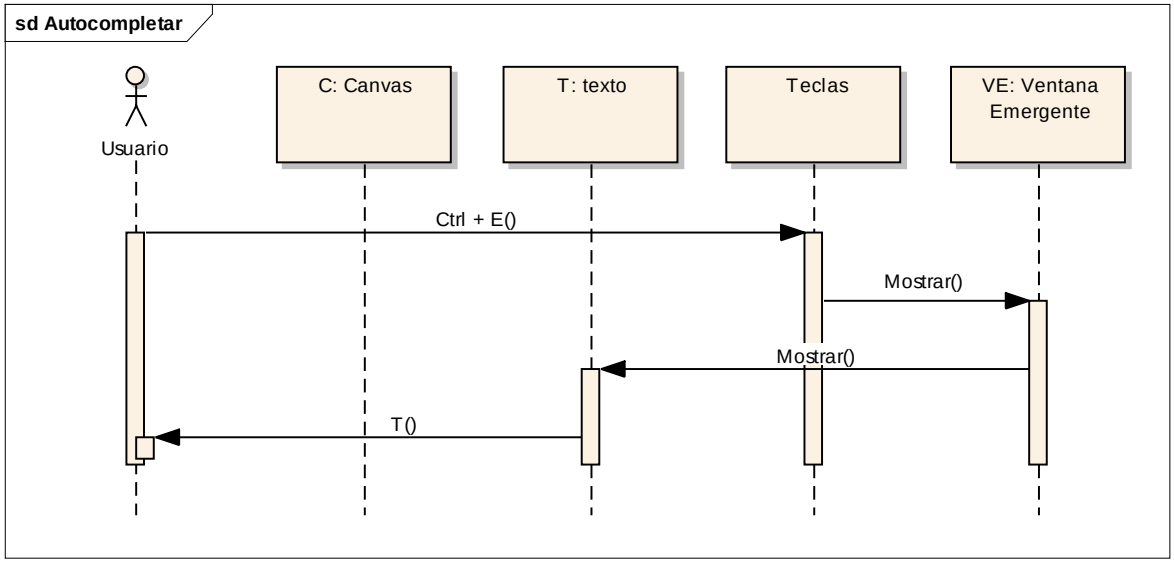

Ilustración 27. Diagrama de secuencia: Autocompletar

Diagrama de colaboración

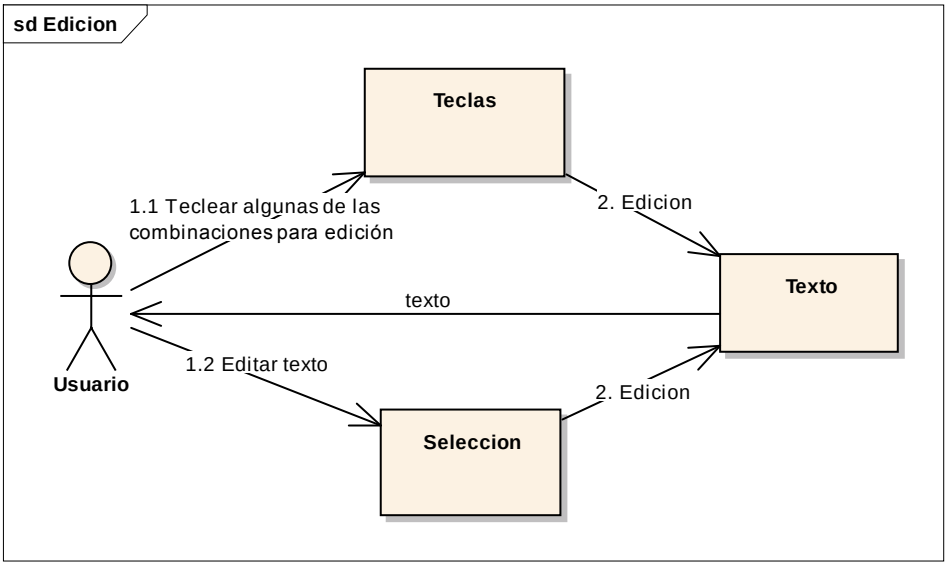

Ilustración 28. Diagrama de colaboración: Edición

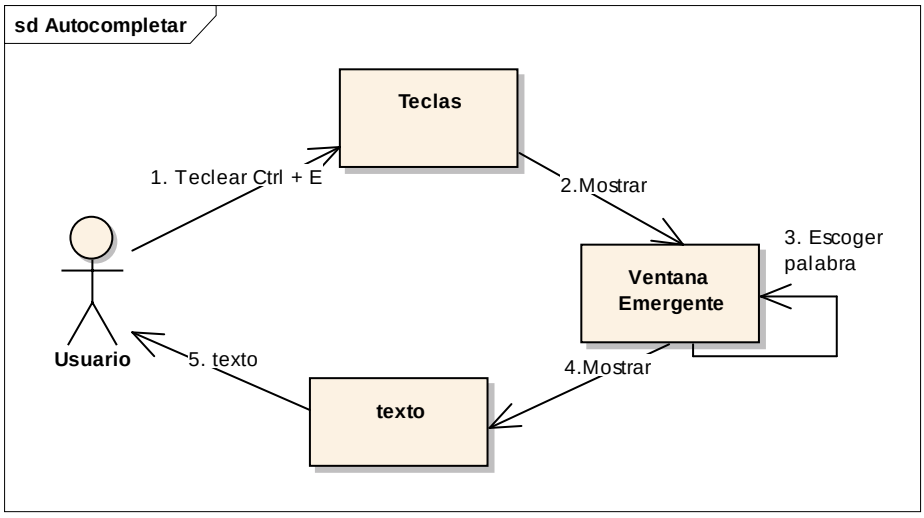

Ilustración 29. Diagrama de colaboración: Autocompletar

Diagrama de estados

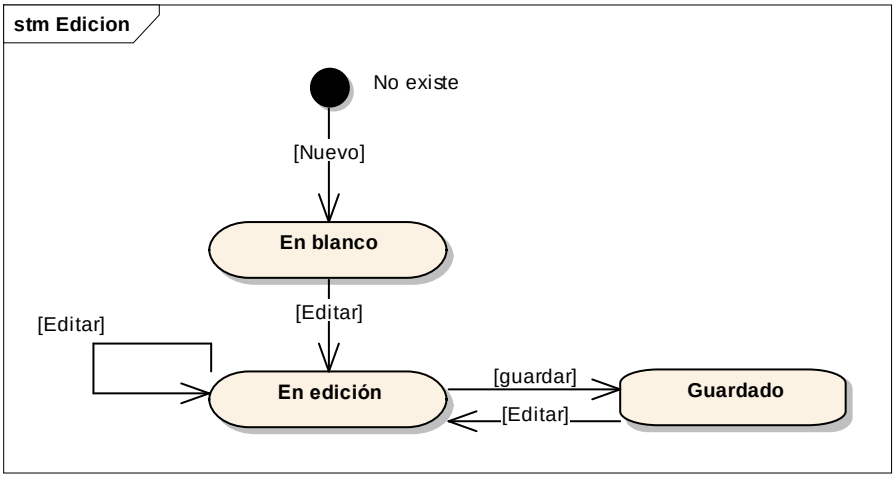

Ilustración 30. Diagrama de estados: Edición

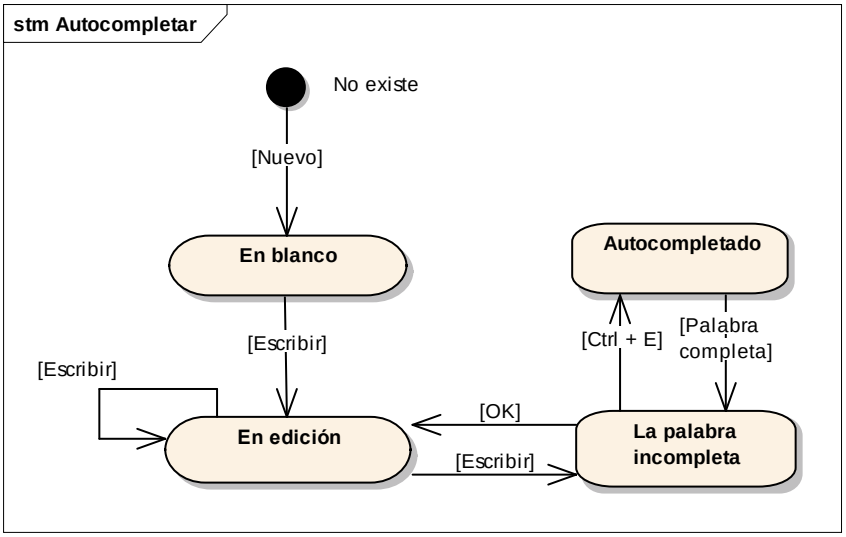

Ilustración 31. Diagrama de estados: Autocompletar

Diagrama de actividades

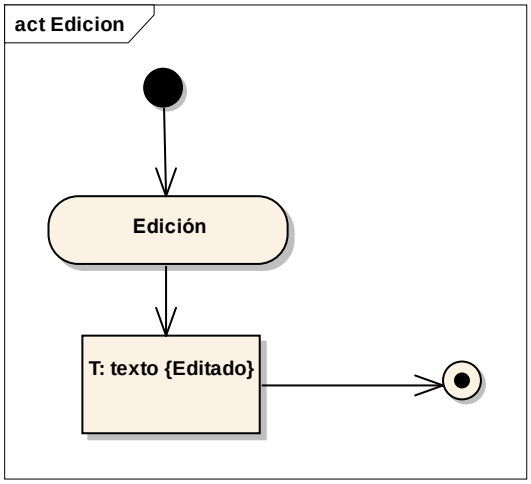

Ilustración 32. Diagrama de actividades: Edición

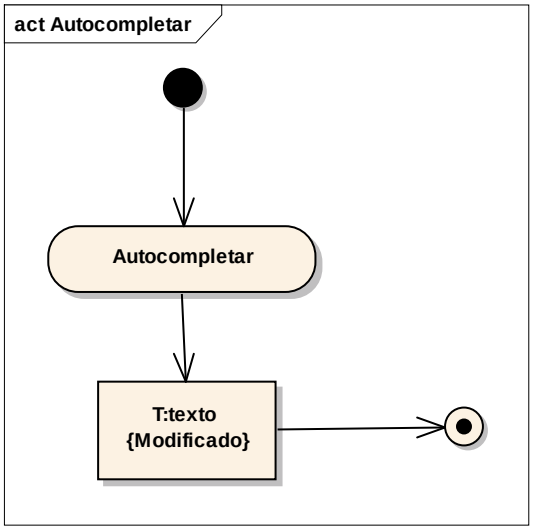

Ilustración 33. Diagrama de actividades: Autocompletar

8.2.3.1.1 interfaz Hombre-Maquina

Ilustración 34. Funciones de edición

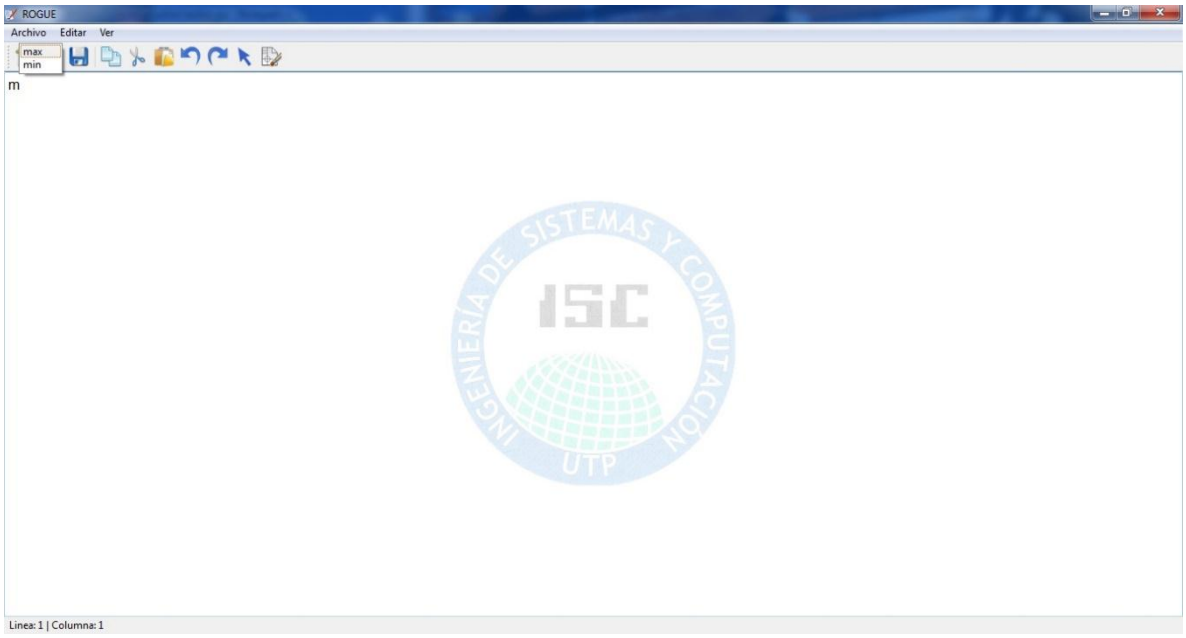

Ilustración 35. Autocompletar

8.2.3.2 Revisión del SPRINT

Se cumple con el objetivo de permitir al usuario realizar funciones de edición a sus archivos, tales como copiar, cortar, pegar, seleccionar, borrar, rehacer y deshacer. Además permite autocompletar dentro del archivo palabras reservadas del lenguaje de dominio específico para problemas de optimización lineal.

Se cumple con las pruebas de aceptación de cada una de las HU:

- Seleccionamos el segmento de texto a copiar y verificamos que este no sea removido de su origen.

- Seleccionamos el segmento de texto a cortar y verificamos que este fue removido de su origen.
- Pegamos un segmento de texto y verificamos que se haya insertado
- Seleccionamos el segmento de texto a borrar y verificamos que fue removido del archivo
- Activamos la función seleccionar todo y verificamos que todo el contenido del archivo se encuentre sombreado
- Activamos la función deshacer y comprobamos que el editor vuelve al estado anterior a dicha acción
- Activamos la función rehacer y comprobamos que el editor revierte el último cambio realizado.
- Codificamos en el lenguaje de dominio específico y verificamos que el editor indica las palabras reservadas del lenguaje.

8.2.3.3 Retrospectiva del SPRINT

¿Qué salió bien en la iteración?

Se cumplió con el objetivo planteado inicialmente, el editor cuenta con la funcionalidad de edición y le permite al usuario autocompletar.

Valor agregado

El editor de texto continúa siendo intuitivo, se adiciona la funcionalidad de ocultar la barra de herramientas y la barra de estado.

¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua).

Ninguna

8.2.4 Plan iteración No. 2 entrega No. 2

1. Objetivos de la iteración

El objetivo que se ha diseñado para esta segunda iteración de la segunda entrega es proporcionar al usuario la herramienta de buscar una palabra dentro del archivo y eliminar un fichero del disco duro.

2. Listado inicial de historias de usuarios a desarrollar

La lista de las Historias de usuario que se van a desarrollar en esta iteración son:

Ident	Título	Est
HU.8	Un usuario puede buscar dentro del archivo de texto	3
HU.5	Un usuario puede eliminar un archivo de texto	3

Tabla 49. HU para la iteración No.2 Entrega No.2

3. Descomposición en Tareas de Desarrollo

Se incluye el resultado de la descomposición de las historias de usuario en tareas de desarrollo, así como la asignación a los desarrolladores y la estimación realizada de su duración.

Identificador: HU.5	Eliminar un archivo de texto	3

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 5-1	Diseñar los diagramas UML	5	Desarrollador 1
Tarea 5-2	Diseñar e implementar la interfaz gráfica del editor para eliminar un archivo del disco duro.	5	Desarrollador 2

Observaciones:

Tabla 50. Tareas de desarrollo para la HU.5

Identificador: HU.8	Buscar dentro del archivo de texto	3

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 8-1	Diseñar los diagramas UML	5	Desarrollador 1

Tarea 8-2	Diseñar e implementar la interfaz gráfica del editor para buscar una palabra dentro del archivo de texto.	5	Desarrollador 2
Observaciones:			

Tabla 51. Tareas de desarrollo para la HU.8

4. Carga prevista en los desarrolladores

Información final sobre la carga prevista de trabajo de cada uno de los miembros del equipo de desarrollo en base a las tareas asignadas en la iteración.

Desarrollador	Velocidad Inicial (días ideales)	Dedicación (% de tiempo)	Carga de trabajo (días ideales)	Tareas Aceptadas (cantidad)
Desarrollador 1	20	100%	10	4
Desarrollador 2	20	100%	10	4

Tabla 52. Carga para los desarrolladores en la Iteración No.2

5. Planificación temporal de la iteración.

La planificación temporal que se ha utilizado para hacer las estimaciones es la siguiente:

Semana	Desarrollador	Día 1	Día 2	Día 3	Día 4	Día 5
Semana 1	Desarrollador 1	Tarea 5-1	Tarea 5-1	Tarea 5-1	Tarea 5-1	Tarea 5-1
	Desarrollador 2	Tarea 5-2	Tarea 5-2	Tarea 5-2	Tarea 5-2	Tarea 5-2
Semana 2	Desarrollador 1	Tarea 8-1	Tarea 8-1	Tarea 8-1	Tarea 8-1	Tarea 8-1
	Desarrollador 2	Tarea 8-2	Tarea 8-2	Tarea 8-2	Tarea 8-2	Tarea 8-2

Tabla 53. Planificación temporal de la iteración No.2

8.2.4.1 Desarrollo de las tareas

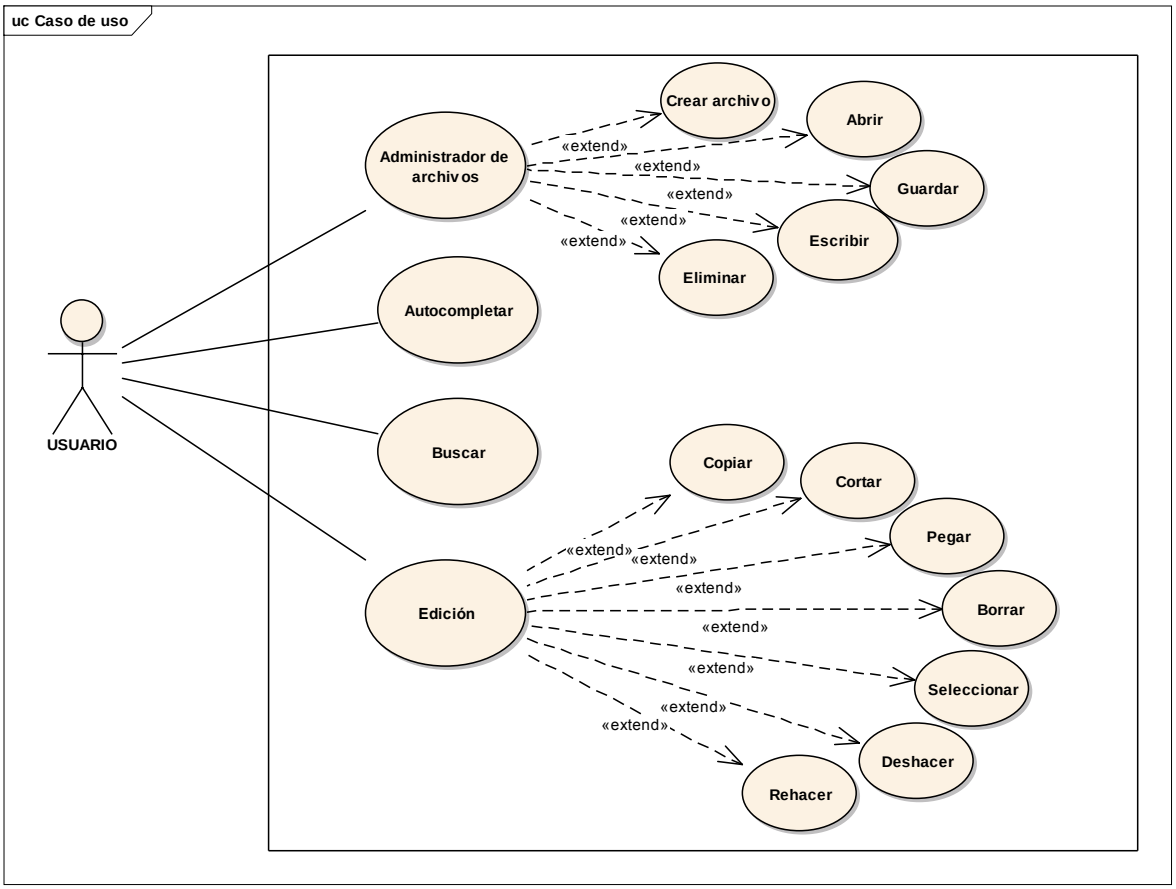

Ilustración 36.Diagrama de casos de uso iteración 4.

CU-005	**Eliminar archivo de texto**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1
Precondición	El usuario debe haber creado el archivo previamente.
Descripción	El usuario puede tener acceso en el menú del editor de texto la función eliminar del sistema operativo para borrar el fichero actual del disco duro.
Referencias	HU.5

Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario selecciona el archivo a eliminar y elige esta opción.	**2.** Elimina el archivo del disco duro
Postcondición	Se elimina el archivo de texto
Comentarios	

Tabla 54. CU: Eliminar archivo de texto

CU-008	**Buscar**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1 HU.2
Precondición	El usuario debe tener texto dentro del archivo.
Descripción	El usuario puede buscar una palabra dentro del archivo texto para encontrarla con facilidad.
Referencias	HU.8
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario selección la opción buscar **3.** El usuario escribe lo que desea buscar	**2.** Desplegar la ventana indicada **4.** Sombrear la palabra buscada por el usuario
Postcondición	
Comentarios	

Tabla 55. CU: Buscar

Diagrama de clase

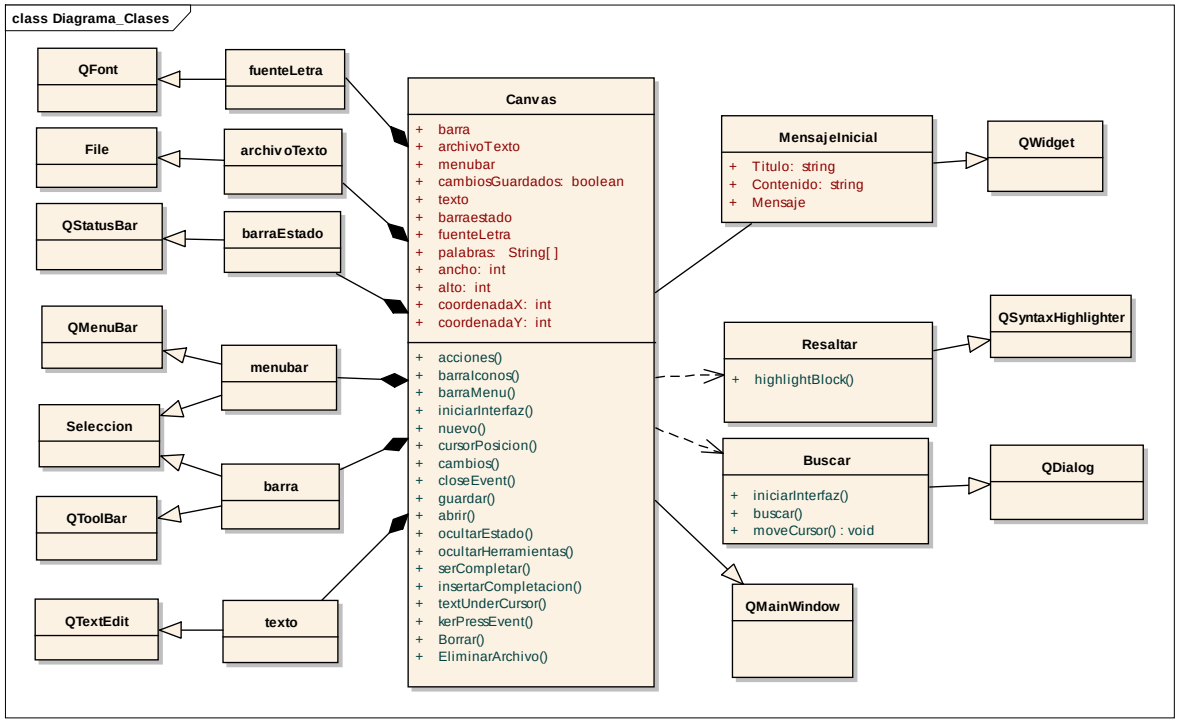

Ilustración 37. Diagrama de clases iteración 2 Entrega No2

Diagrama de secuencia

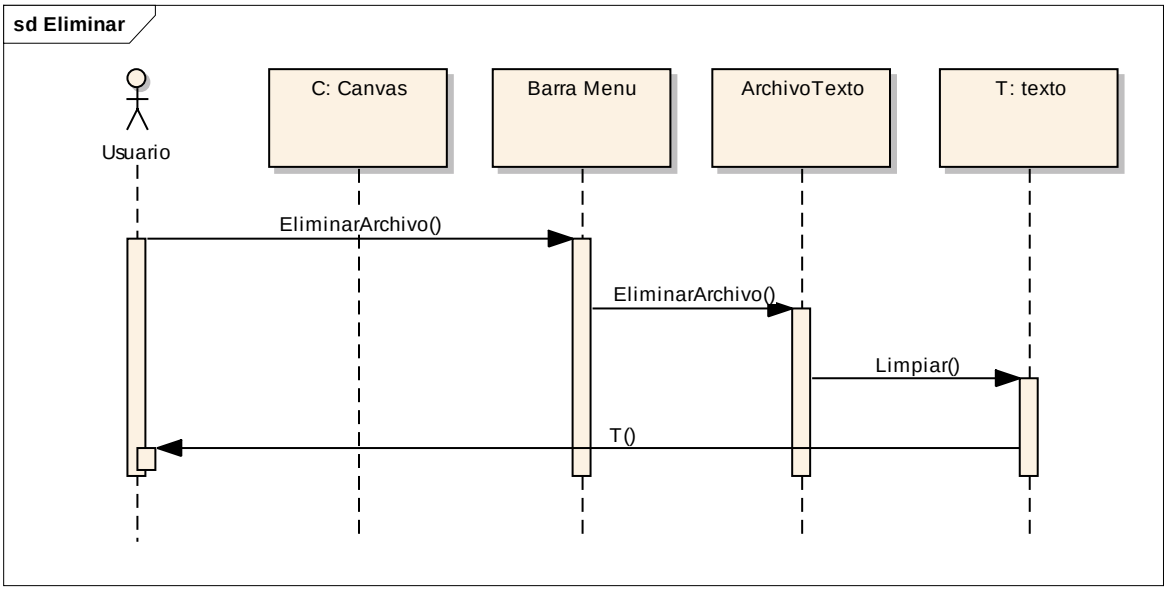

Ilustración 38. Diagrama de secuencia: Eliminar

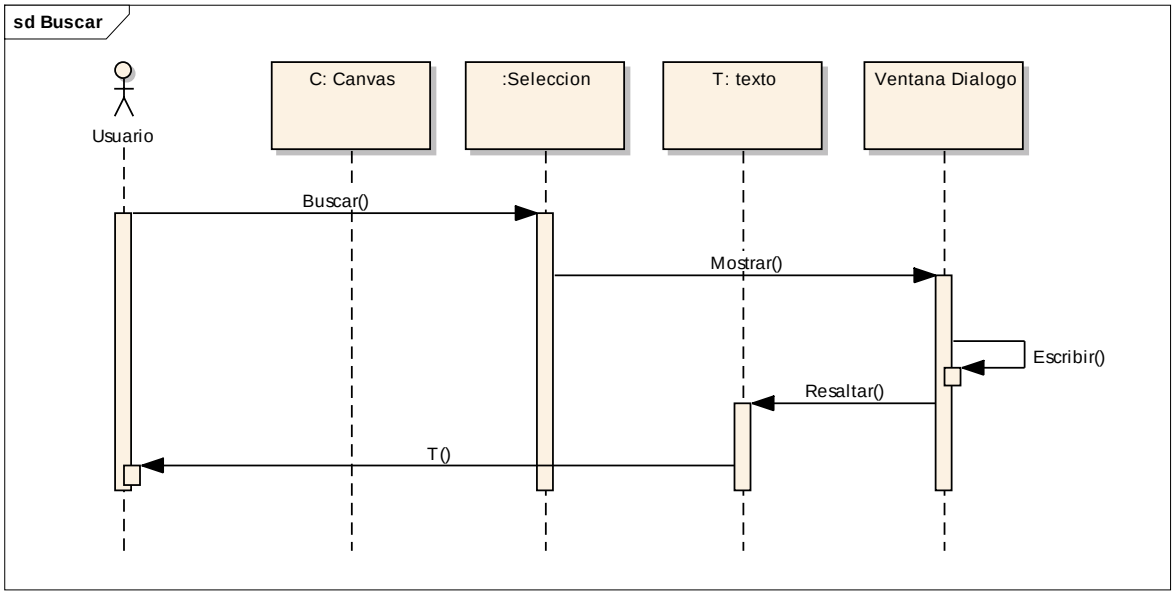

Ilustración 39. Diagrama de secuencia: Buscar

Diagrama de colaboración

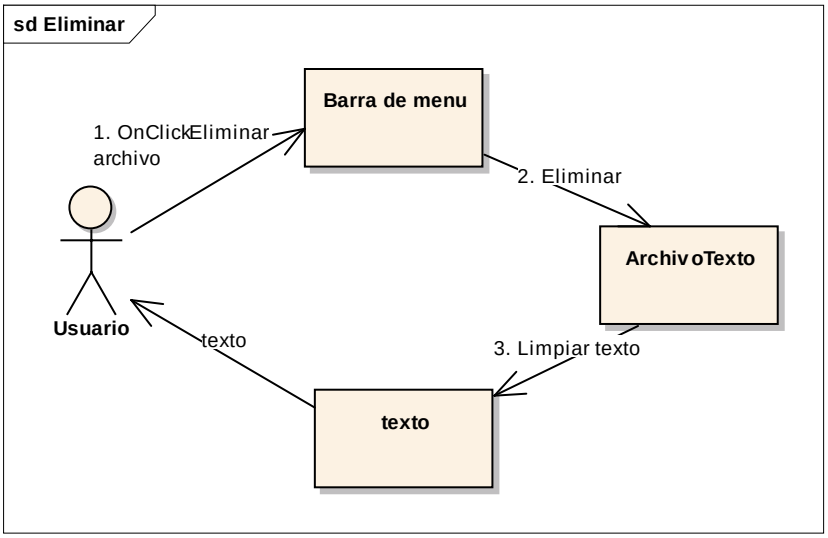

Ilustración 40. Diagrama de colaboración: Eliminar

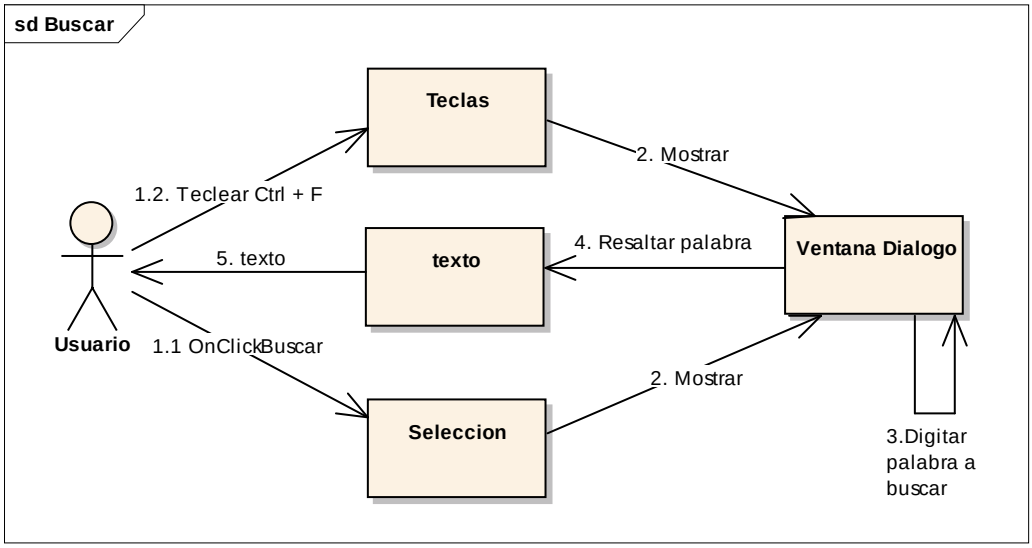

Ilustración 41. Diagrama de colaboración: Buscar

Diagrama de estados

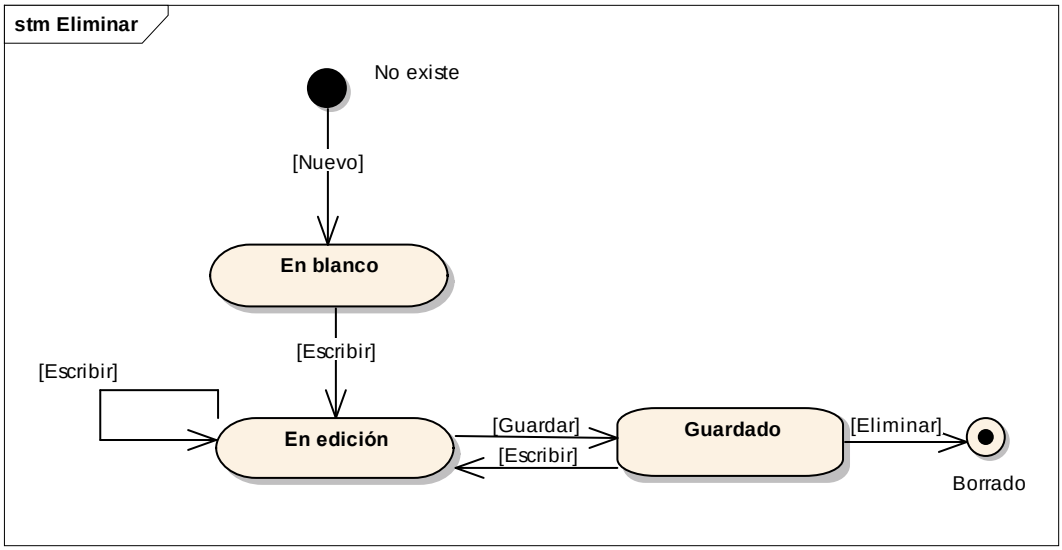

Ilustración 42. Diagrama de estados: Eliminar

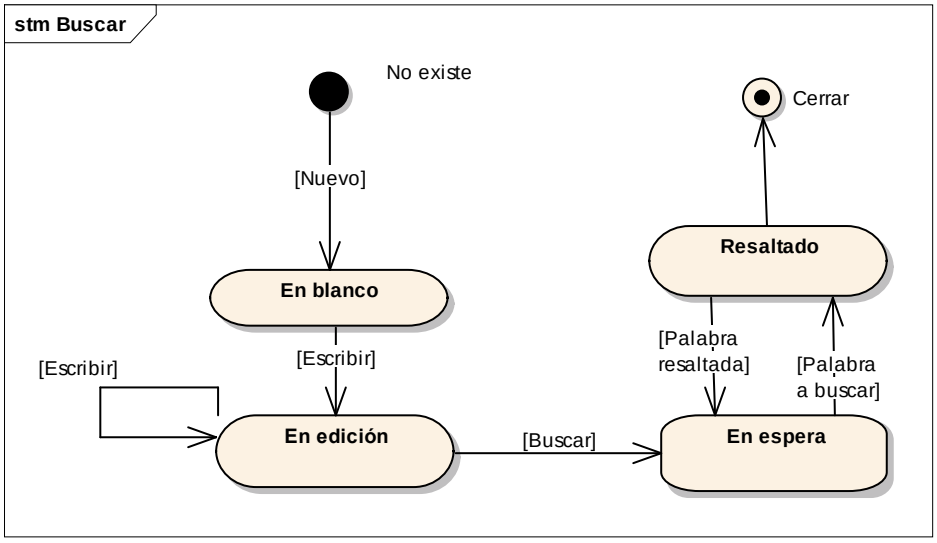

Ilustración 43. Diagrama de estados: Buscar

Diagrama de actividades

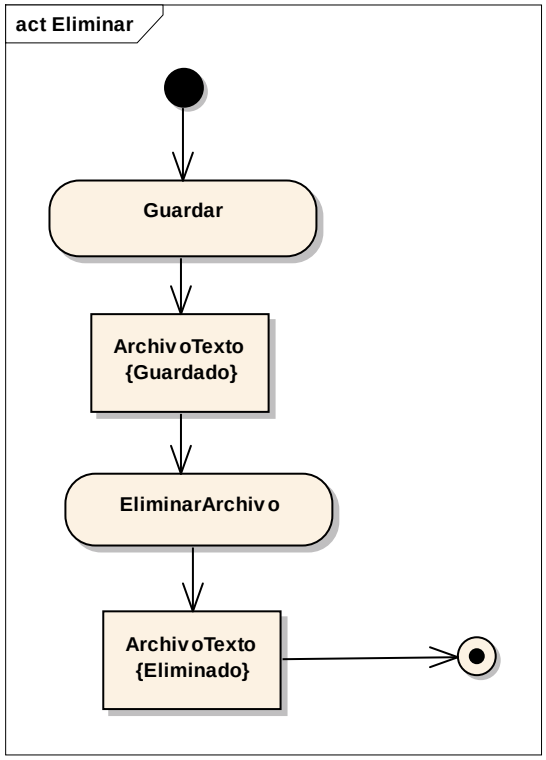

Ilustración 44. Diagrama de actividades: Eliminar

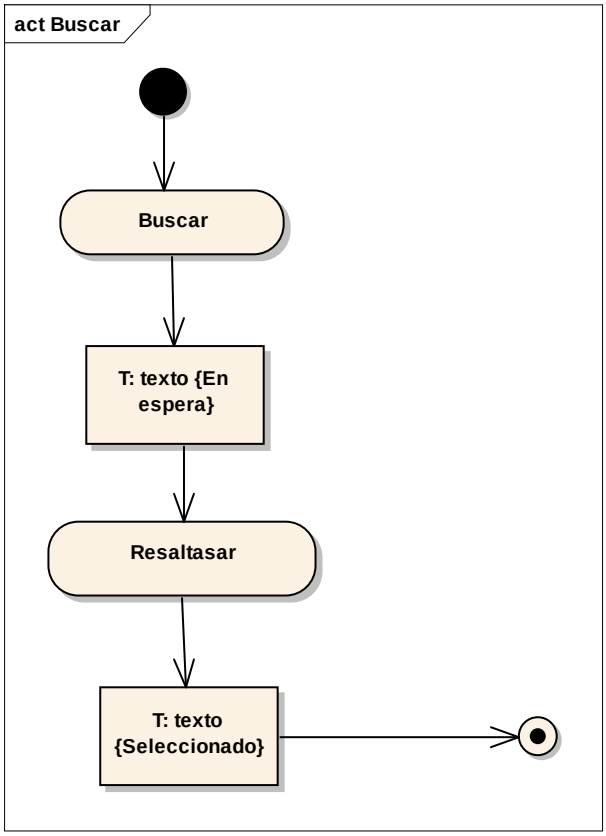

Ilustración 45. Diagrama de actividades: Buscar

8.2.4.1.1 Interfaz Hombre-Máquina

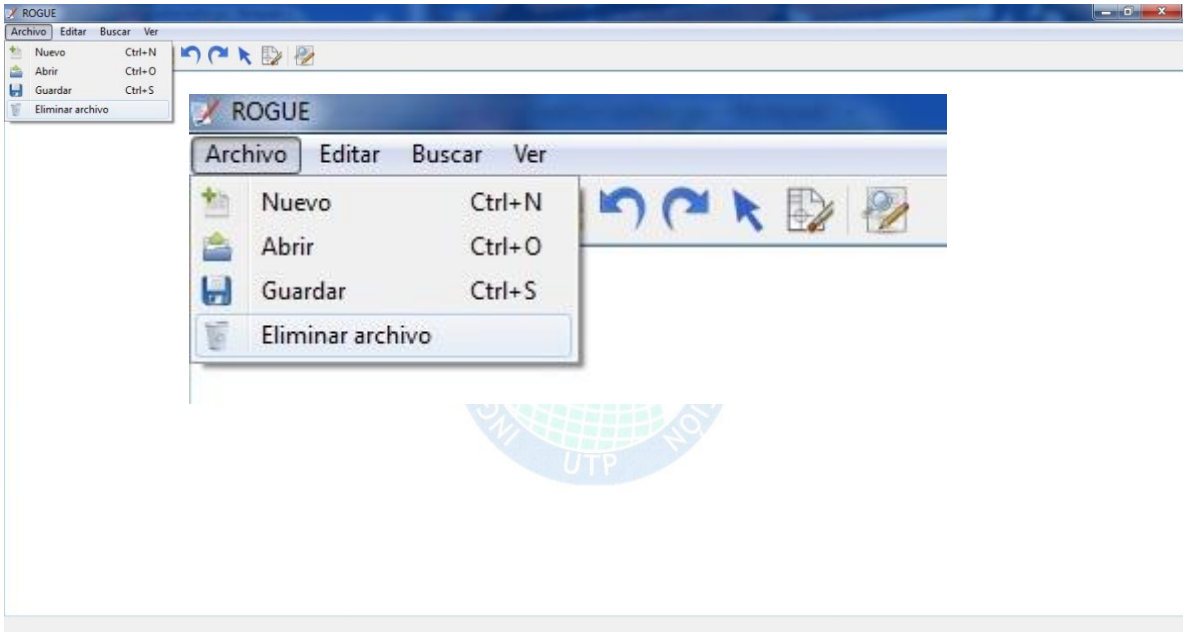

Ilustración 46. Eliminar archivo

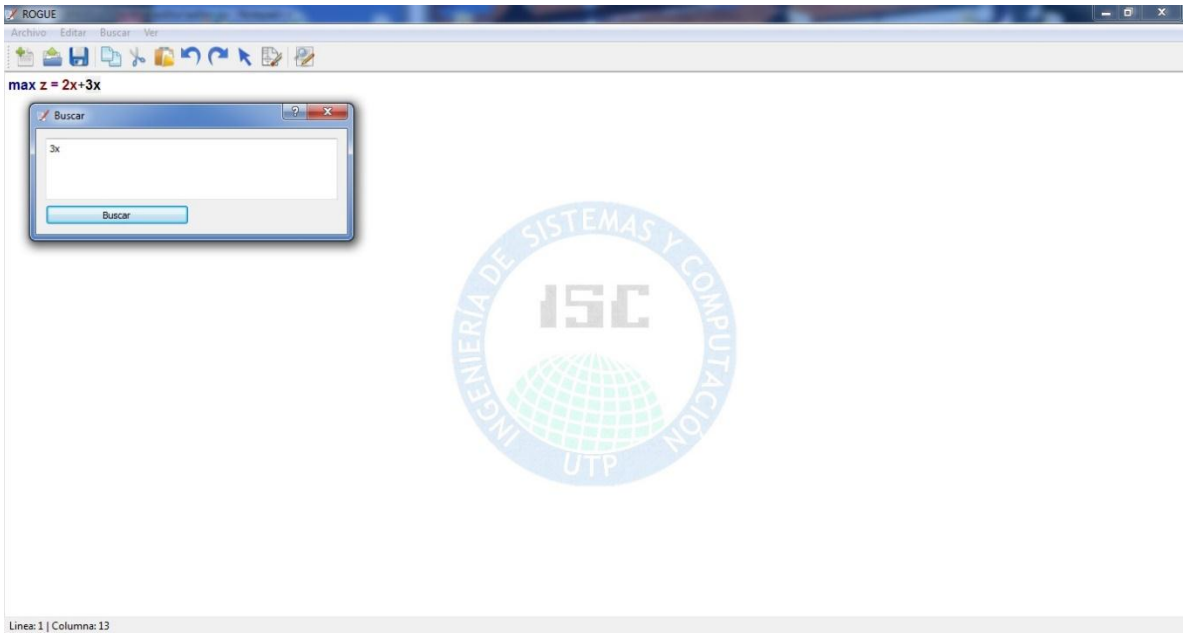

Ilustración 47. Buscar

8.2.4.2 Revisión del SPRINT

Se cumple el objetivo de proporcionar al usuario la herramienta de buscar una palabra dentro del archivo y eliminar un fichero del disco duro.

Se cumple con las pruebas de aceptación de cada una de las HU:

- Eliminamos un archivo y verificamos que no se encuentre en el disco duro
- Buscamos una palabra que se encuentre dentro del archivo y verificamos que se encuentra sombreada
- Buscamos una palabra que no se encuentre dentro del archivo y comprobamos que se informa del error

8.2.4.3 Retrospectiva del SPRINT

¿Qué salió bien en la iteración?

Se cumplió con el objetivo planteado inicialmente, el editor cuenta con la funcionalidad de buscar una palabra en el archivo de texto y eliminar el archivo del disco duro.

Valor agregado

La barra de herramientas del editor puede cambiar de posición.

¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua).

Para las próximas iteraciones es importante adicionar la función imprimir y vista previa del archivo de texto.

8.2.5 Plan iteración No. 1 entrega No. 3

1. Objetivos de la iteración

El objetivo que se ha diseñado para esta primera iteración de la última entrega es Proporcionar al usuario la herramienta de reemplazar una palabra en su archivo de texto.

2. Listado inicial de historias de usuario a desarrollar

La lista de las Historias de usuario que se van a desarrollar en esta iteración son:

Ident	Título	Est
HU.9	Un usuario puede reemplazar una palabra dentro del archivo de texto	3

Tabla 56. HU para la iteración No.1 Entrega No.3

3. Descomposición en Tareas de Desarrollo

Se incluye el resultado de la descomposición de las historias de usuario en tareas de desarrollo, así como la asignación a los desarrolladores y la estimación realizada de su duración.

Identificador: HU.9	Reemplazar una palabra dentro del archivo de texto	3

Identificador	Título de la tarea de desarrollo	Estimación (días Ideales)	Desarrollador Asignado
Tarea 9-1	Diseñar los diagramas UML	4	Desarrollador 1
Tarea 9-2	Diseñar e implementar la interfaz gráfica del editor para reemplazar una palabra dentro del archivo de texto.	4	Desarrollador 2

Observaciones:

Tabla 57. Tareas de desarrollo para la HU.9

4. Carga prevista en los desarrolladores

Información final sobre la carga prevista de trabajo de cada uno de los miembros del equipo de desarrollo en base a las tareas asignadas en la iteración.

Desarrollador	Velocidad Inicial (días ideales)	Dedicación (% de tiempo)	Carga de trabajo (días ideales)	Tareas Aceptadas (cantidad)

Desarrollador 1	20	100%	10	4
Desarrollador 2	20	100%	10	4

Tabla 58. Carga para los desarrolladores en la Iteración No.1

5. Planificación temporal de la iteración.

La planificación temporal que se ha utilizado para hacer las estimaciones es la siguiente:

Semana	Desarrollador	Día 1	Día 2	Día 3	Día 4	Día 5
Semana 1	Desarrollador 1	Tarea 9-1	Tarea 9-1	Tarea 9-1	Tarea 9-1	
	Desarrollador 2	Tarea 9-2	Tarea 9-2	Tarea 9-2	Tarea 9-2	

Tabla 59. Planificación temporal de la iteración No.1

6. Desviaciones previstas.

Para esta iteración se consideró importante adicionar dos nuevas funcionalidades al editor.

Ident	Título	Est
HU.10	Un usuario puede imprimir un archivo de texto	1/2
HU.11	Un usuario puede tener una vista previa de su archivo de texto	1/2

Tabla 60. HU para la iteración No.1

Modificación de la tercera entrega:

Entrega	Objetivo	Fecha de la entrega
3	Permitir que el usuario pueda reemplazar, imprimir y tener una vista previa de su archivo de texto.	30 de Abril de 2015

Iteración	Objetivo
1	Proporcionar al usuario la herramienta de reemplazar una palabra, imprimir y tener una vista previa de su archivo de texto.

Tabla 61. Plan de entrega No.3

Se adiciona una descripción completa de las historias de usuario:

Identificador: HU.10	Imprimir un archivo de texto	
Descripción: Como usuario quiero imprimir el archivo de texto.		
Estimación: 1/2	**Prioridad: 3**	**Entrega: 3**
Pruebas de aceptación: • Seleccionamos la función imprimir y verificamos que se muestre la ventana imprimir del sistemas operativo.		
Observaciones:		

Tabla 62. Tarjeta de HU.10: Imprimir

Identificador: HU.11	Vista previa de un archivo de texto	
Descripción: Como usuario quiero tener una vista previa del archivo de texto.		
Estimación: 1/2	**Prioridad: 3**	**Entrega: 3**
Pruebas de aceptación: • Seleccionamos la función vista previa y verificamos que el archivo de texto se muestre en otra ventana.		
Observaciones:		

Tabla 63. Tarjeta de HU.11: Vista previa

Descomposición en Tareas de Desarrollo

Se incluye nuevamente la descomposición de las historias de usuario en tareas de desarrollo.

Identificador: HU.11	Imprimir un archivo de texto	1/2

Identificador	**Título de la tarea de desarrollo**	**Estimación (días Ideales)**	**Desarrollador Asignado**
Tarea 10-1	Diseñar los diagramas UML	3	Desarrollador 1

Tarea 10-2	Diseñar e implementar la interfaz gráfica del editor para imprimir un archivo de texto.	3	Desarrollador 2

Observaciones:

Identificador: HU.11	Vista previa de un archivo de texto	1/2

Identificador	**Título de la tarea de desarrollo**	**Estimación (días Ideales)**	**Desarrollador Asignado**
Tarea 11-1	Diseñar los diagramas UML	2	Desarrollador 1
Tarea 11-2	Diseñar e implementar la interfaz gráfica del editor para tener la vista previa del archivo.	2	Desarrollador 2

Observaciones:

Carga prevista en los desarrolladores

Información final sobre la carga prevista de trabajo de cada uno de los miembros del equipo de desarrollo en base a las tareas asignadas en la iteración.

Desarrollador	**Velocidad Inicial (días ideales)**	**Dedicación (% de tiempo)**	**Carga de trabajo (días ideales)**	**Tareas Aceptadas (cantidad)**
Desarrollador 1	20	100%	10	4
Desarrollador 2	20	100%	10	4

Planificación temporal de la iteración.

La planificación temporal que se ha utilizado para hacer las estimaciones es la siguiente:

Semana	**Desarrollador**	**Día 1**	**Día 2**	**Día 3**	**Día 4**	**Día 5**

Semana 1	Desarrollador 1					Tarea 10-1
	Desarrollador 2					Tarea 10-2
Semana 2	Desarrollador 1	Tarea 10-1	Tarea 10-1	Tarea 11-1	Tarea 11-1	
	Desarrollador 2	Tarea 10-2	Tarea 10-2	Tarea 11-2	Tarea 11-2	

8.2.5.1 Desarrollo de las tareas

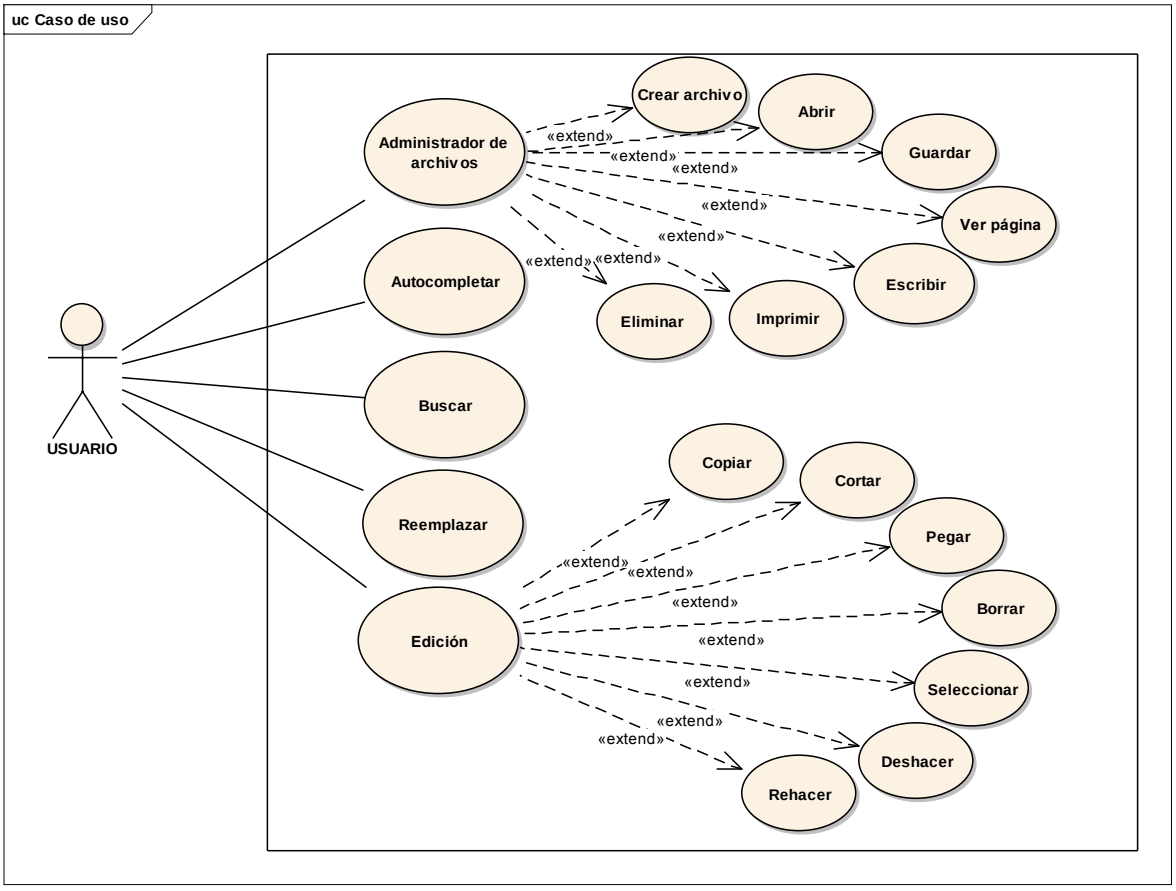

Ilustración 48. Diagrama de casos de uso iteración 5.

CU-009	Reemplazar

Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1 HU.2 HU.8
Precondición	El usuario debe buscar la palabra previamente.
Descripción	El usuario puede reemplazar una palabra dentro del archivo de texto para una fácil modificación.
Referencias	HU.8, HU.9

Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario selección la opción buscar y reemplazar **3.** El usuario escribe lo que desea buscar **5.** Ingresa la palabra por la cual la va reemplazar.	**2.** Desplegar la ventana indicada **4.** Sombrear la palabra buscada por el usuario. **6.** Reemplaza la palabra sombreada por la nueva.

Postcondición	
Comentarios	

Tabla 64. CU: Reemplazar

CU-010	**Imprimir**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1 HU.2
Precondición	Ninguna
Descripción	El usuario puede imprimir el archivo de texto.

Referencias	HU.10
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario selección la opción imprimir o Ctrl+P**.**	**2.** Despliega la ventana con las opciones para imprimir**.**
Postcondición	
Comentarios	

Tabla 65. CU: Imprimir

CU-011	**Vista previa**
Versión	1.0 (23/02/2015)
Actores	Usuario común
Dependencias	HU.1 HU.2
Precondición	Ninguna
Descripción	El usuario puede tener una vista previa del archivo de texto.
Referencias	HU.11
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
1. El usuario selección la opción vista previa o Ctrl+Shift+ P**.**	**2.** Despliega la ventana con la vista previa del archivo**.**
Postcondición	
Comentarios	

Tabla 66. CU: Vista previa

Diagrama de clases

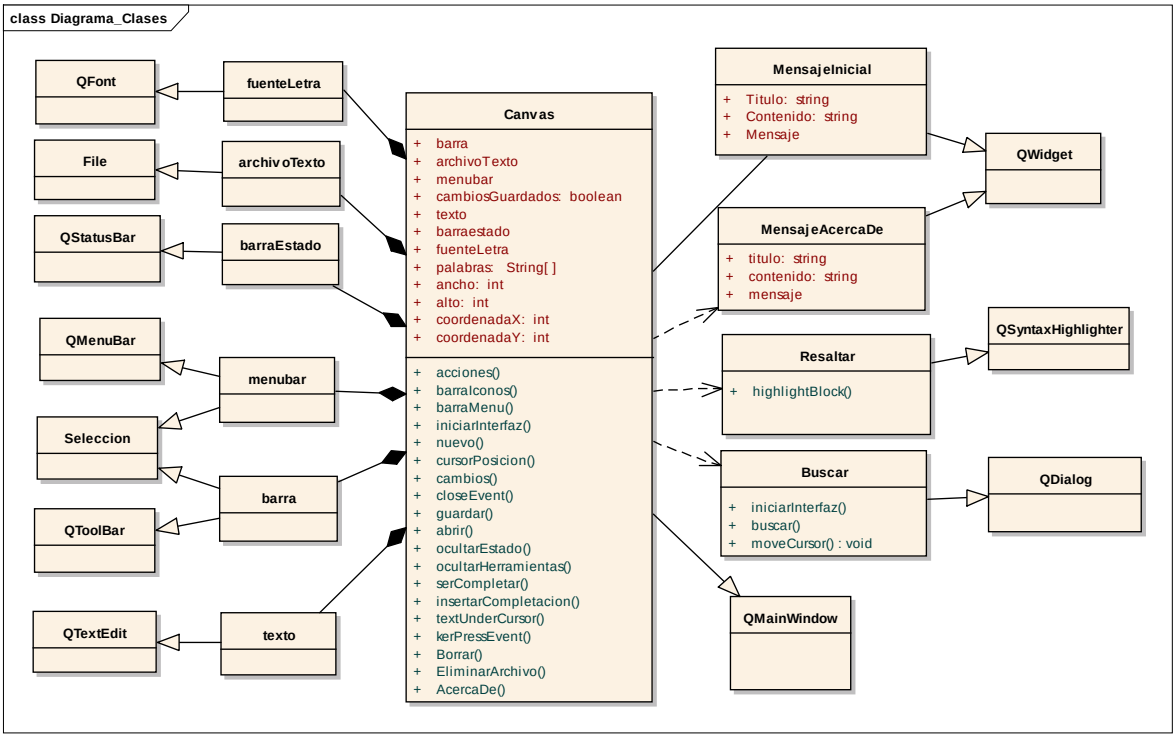

Ilustración 49. Diagrama de clases iteración 1 Entrega No3

Diagrama de secuencia

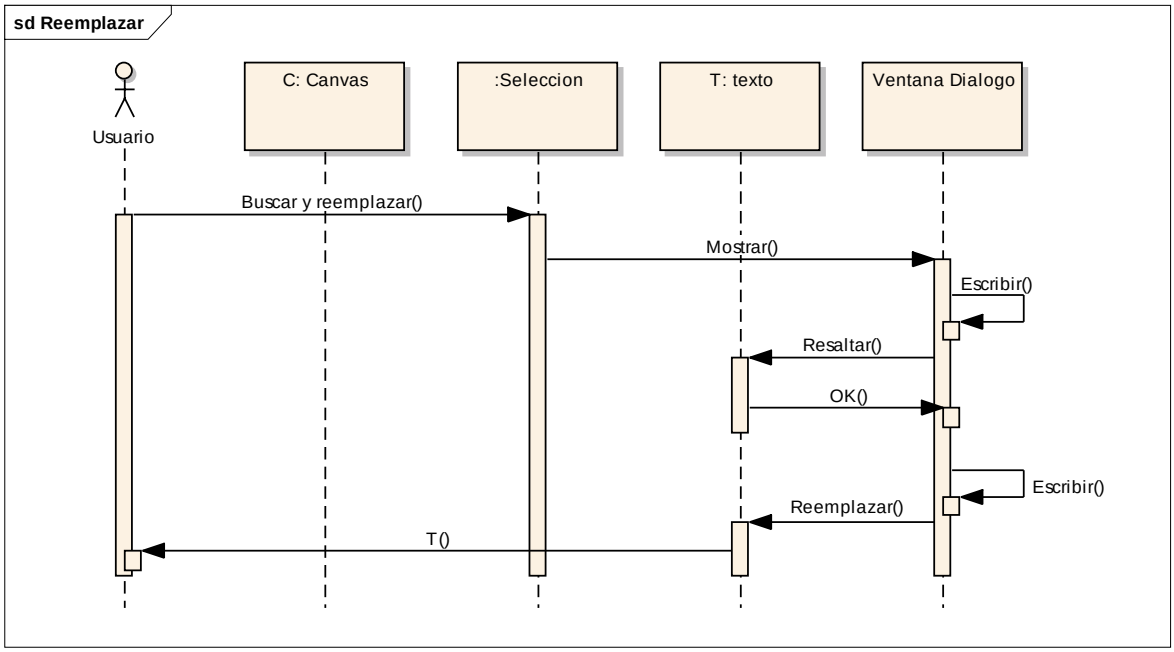

Ilustración 50. Diagrama de secuencia: Reemplazar

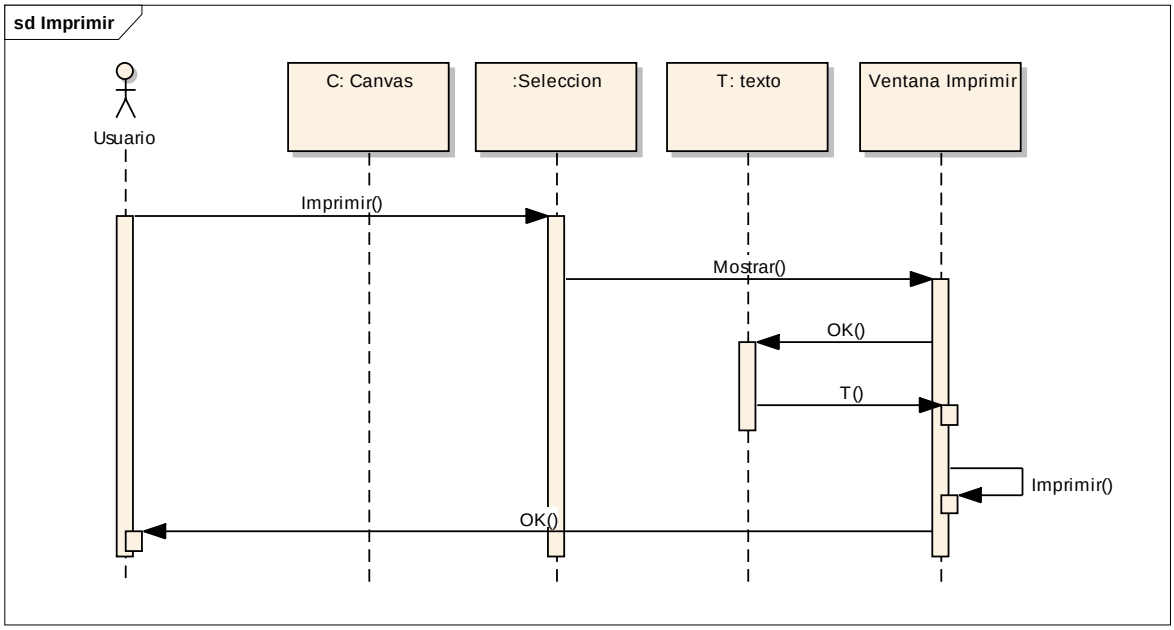

Ilustración 51. Diagrama de secuencia: Imprimir

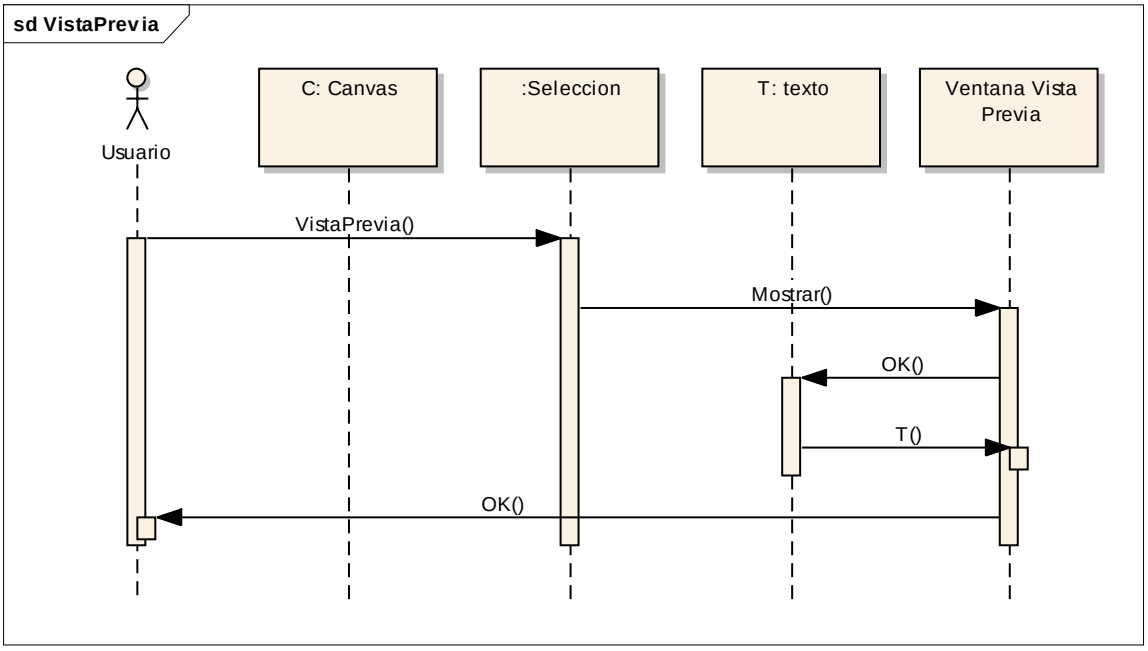

Ilustración 52. Diagrama de secuencia: Vista Previa

Diagrama de colaboración

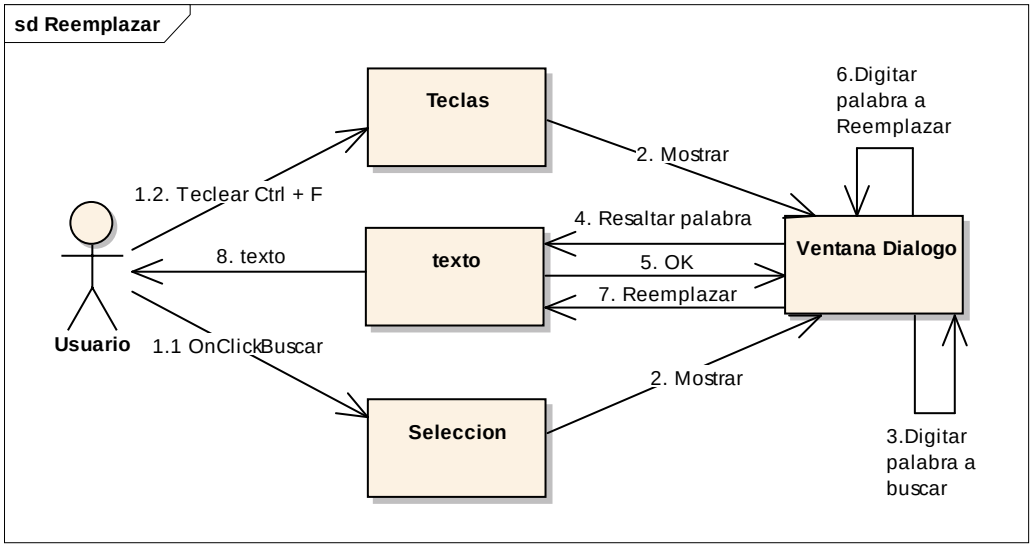

Ilustración 53. Diagrama de colaboración: Reemplazar

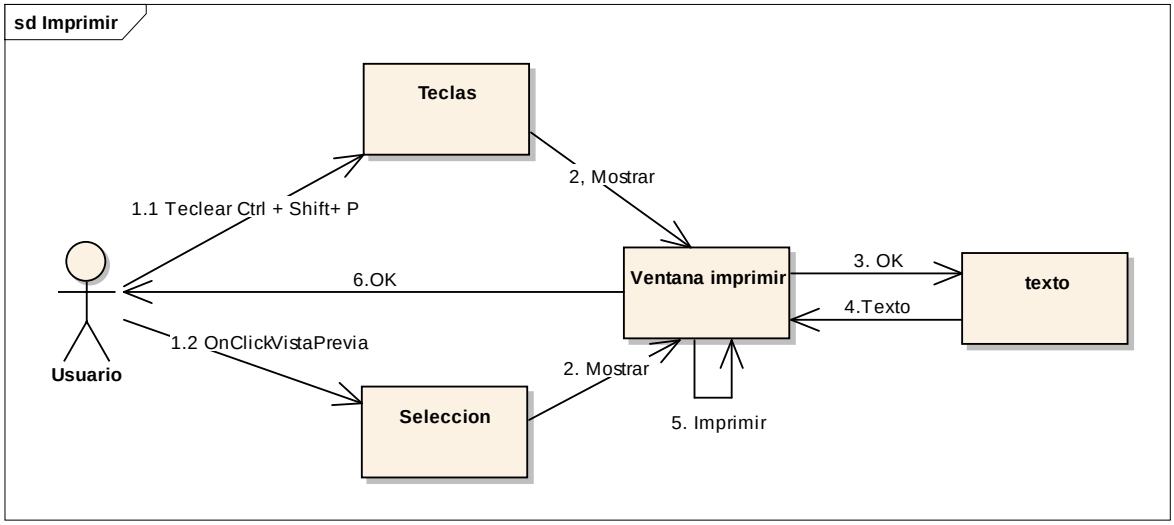

Ilustración 54. Diagrama de colaboración: Imprimir

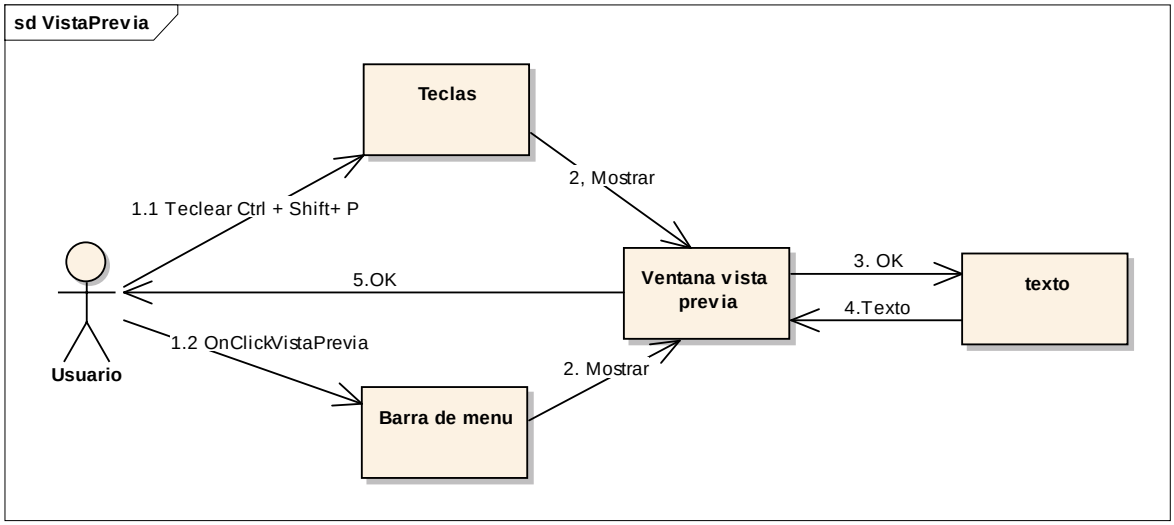

Ilustración 55. Diagrama de colaboración: Vista Previa

Diagrama de estados

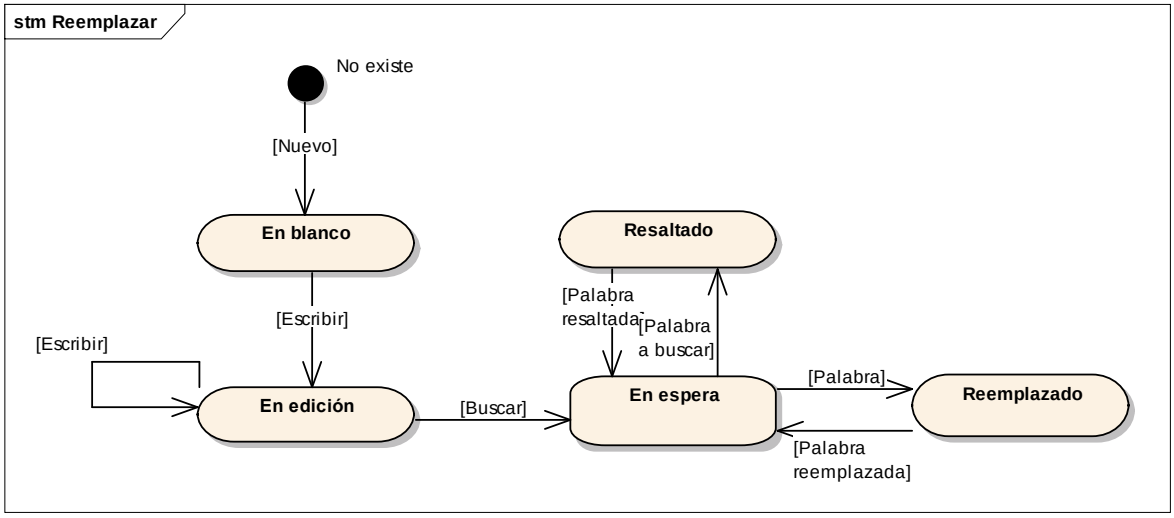

Ilustración 56. Diagrama de estados: Reemplazar

Diagrama de actividades

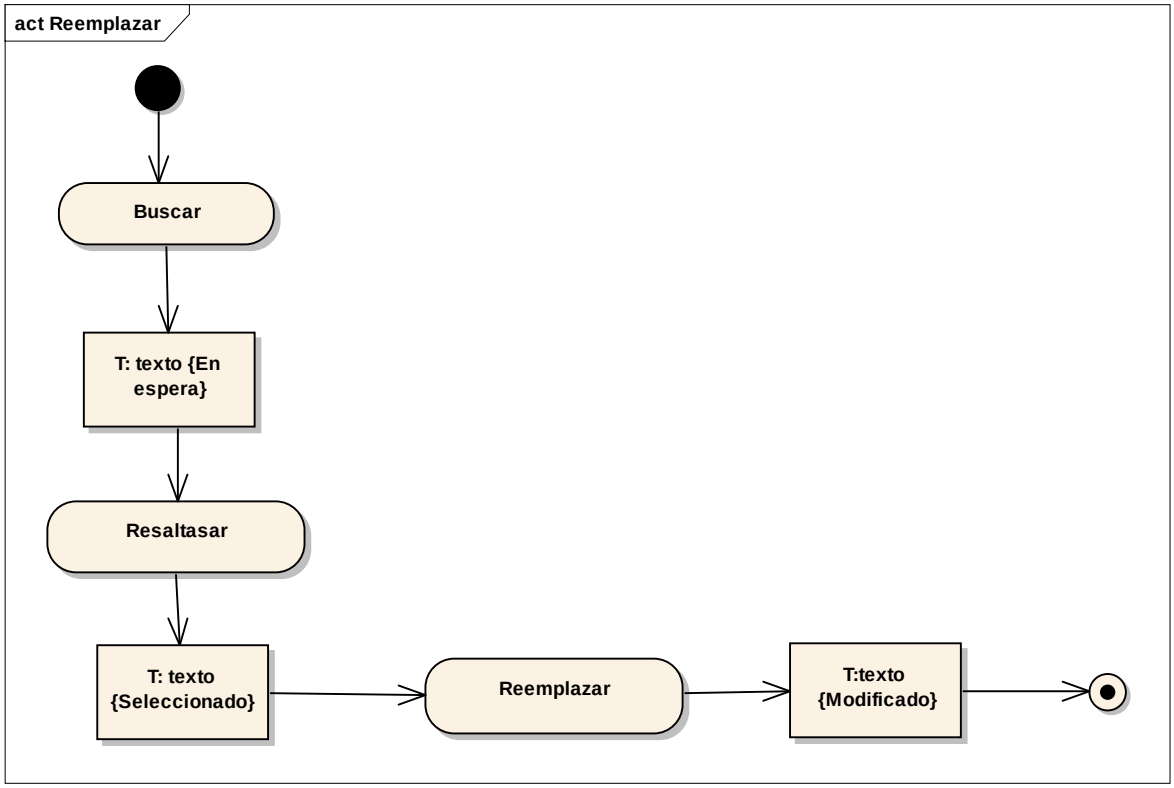

Ilustración 57. Diagrama de actividades: Reemplazar

Los siguientes diagramas muestran la estructura del sistema a un nivel más alto:

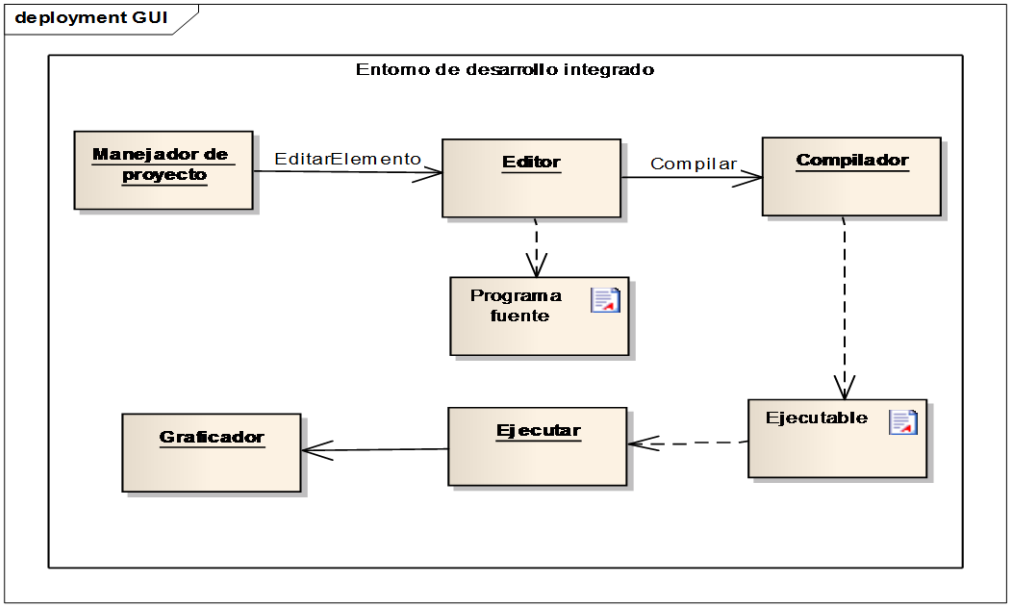

Ilustración 58. Entorno de desarrollo integrado

Diagrama de componentes

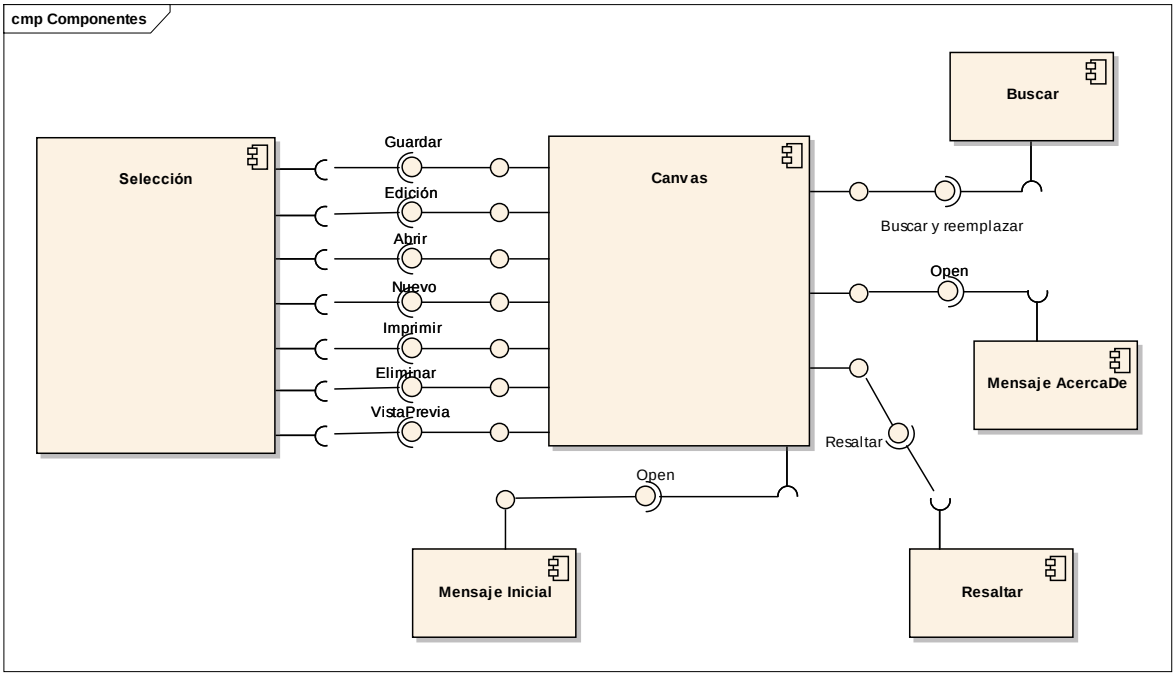

Ilustración 59. Diagrama de componentes

Diagrama de paquetes

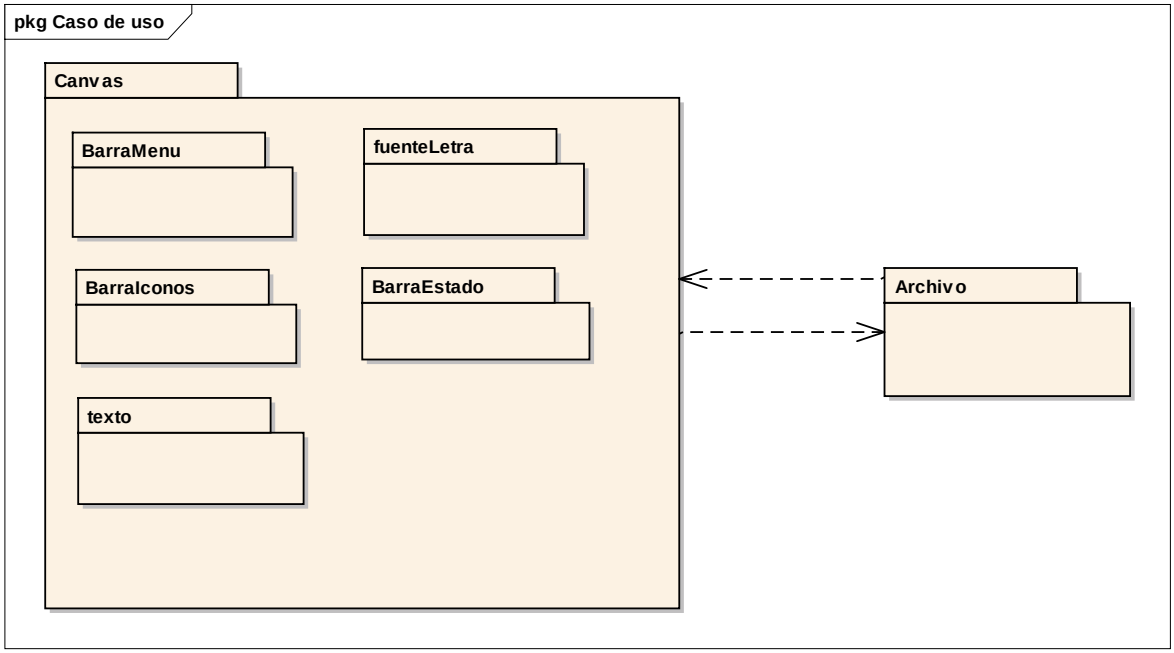

Ilustración 60. Diagrama de paquetes

Diagrama de despliegue

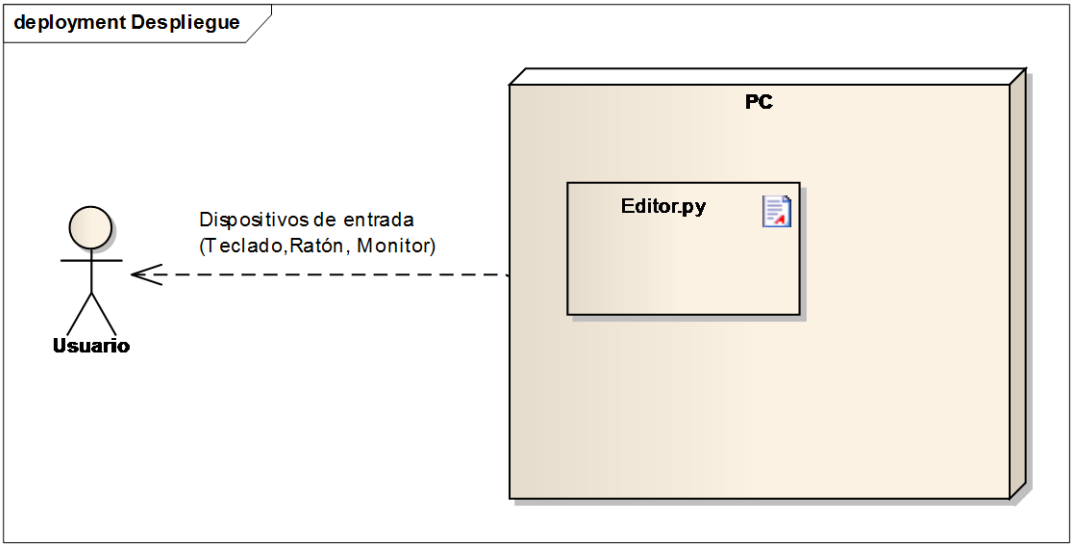

Ilustración 61. Diagrama de despliegue

8.2.5.1.1 Interfaz Hombre-Máquina

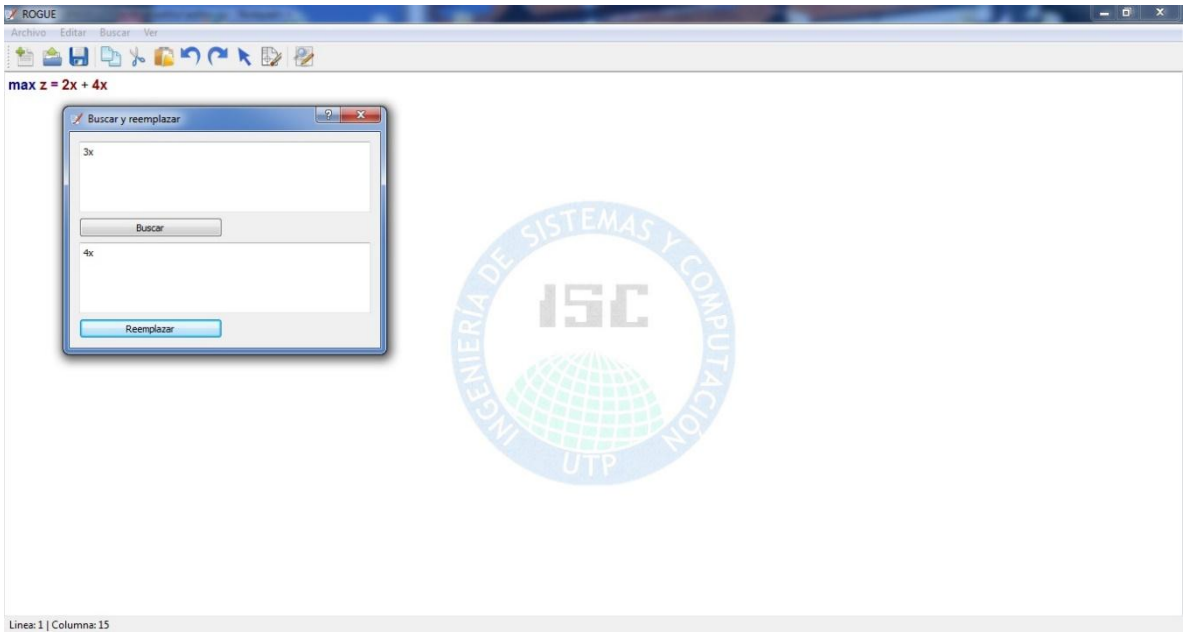

Ilustración 62. Buscar y reemplazar

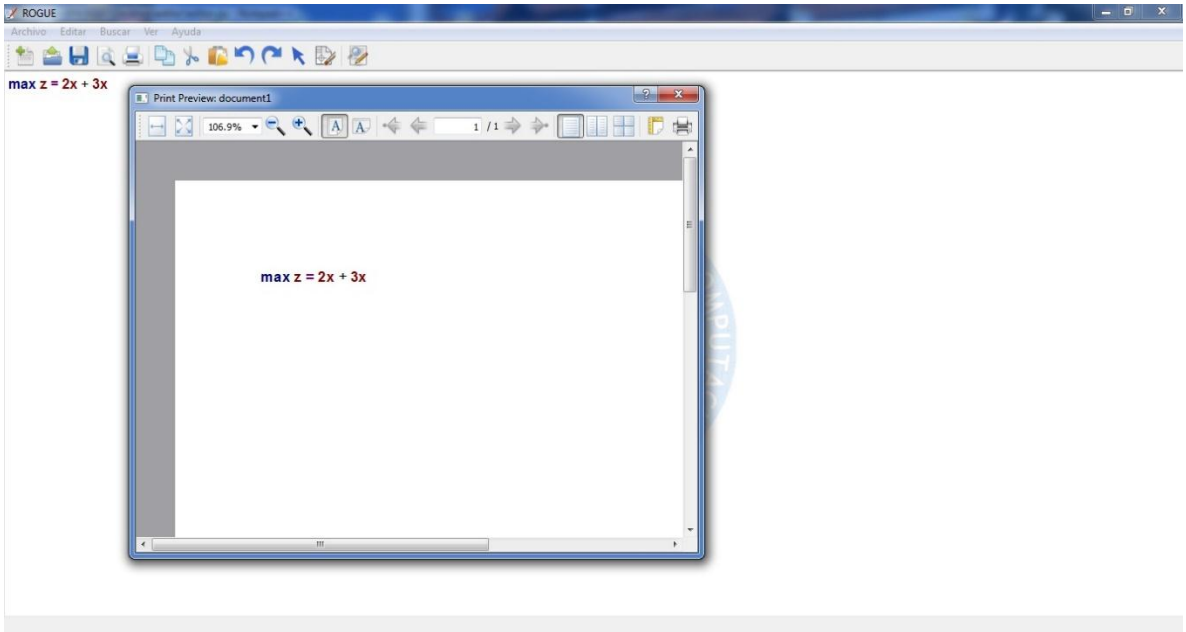

Ilustración 63. Vista previa

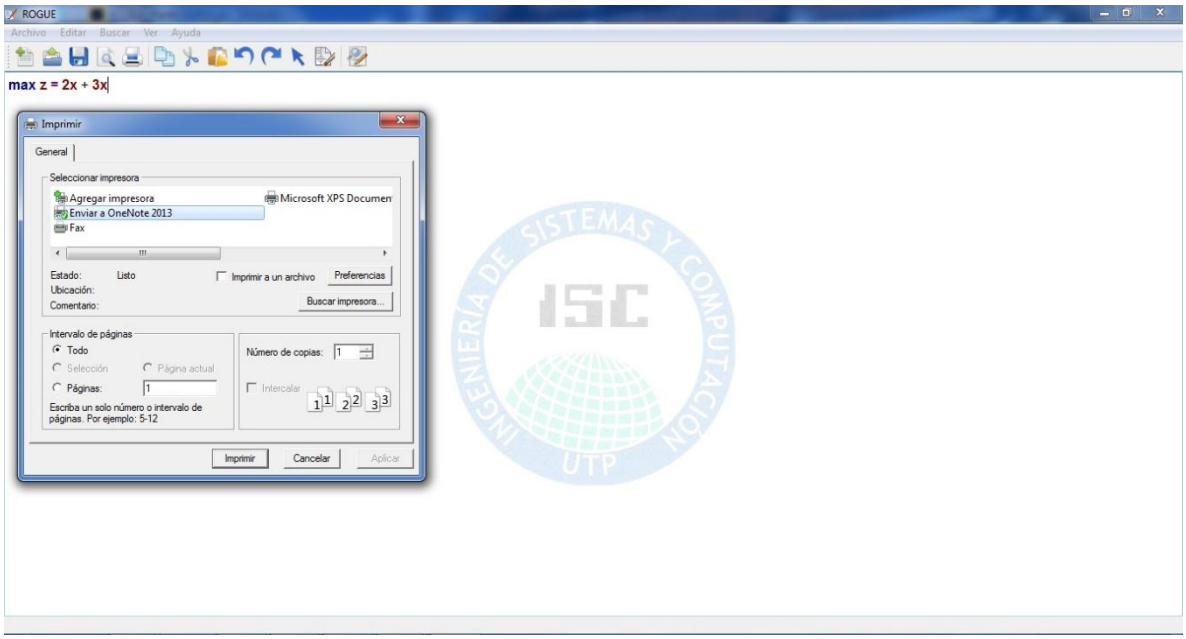

Ilustración 64. Imprimir

8.2.5.2 Revisión del SPRINT

Se cumple el objetivo de proporcionar al usuario la herramienta de reemplazar una palabra en su archivo de texto, imprimir y tener una vista previa de su archivo.

Se cumple con las pruebas de aceptación de cada una de las HU:

- Buscamos la palabra que deseamos cambiar, insertamos la nueva palabra, reemplazamos y verificamos que se haya realizado el cambio
- Seleccionamos la función imprimir y verificamos que se muestre la ventana imprimir del sistemas operativo.
- Seleccionamos la función vista previa y verificamos que el archivo de texto se muestre en otra ventana.

8.2.5.3 Retrospectiva del SPRINT

¿Qué salió bien en la iteración?

Se cumplió con el objetivo planteado inicialmente, el editor cuenta con la funcionalidad de reemplazar una palabra, imprimir y tener una vista previa del archivo de texto.

Valor agregado

Se adiciona en la barra de estado la opción “Acerca de” para que el usuario conozca acerca de las herramientas con las cuales el editor fue diseñado.

¿Qué no salió bien en la iteración? (errores)

Se cumplieron con los objetivos de la iteración.

9. IMPLEMENTACIÓN

Para el desarrollo del editor se escogió la herramienta PyQt, el cual ofrece librerías que facilitan trabajar la interfaces gráficas de Qt a través de Python, específicamente para esta implementación se escogió PyQt4, el cual permite trabajar Qt versión 4 en Python versiones 2 y 3.

Durante el desarrollo del editor de texto se utilizaron los siguientes módulos de PyQt4:

El módulo QtCore que contiene las clases no-GUI fundamentales, incluido el ciclo de eventos y señales y ranura mecanismo de Qt. También incluye la plataforma de abstracciones independientes para Unicode, las discusiones, los archivos mapeados, memoria compartida, las expresiones regulares, y el usuario y la configuración de la aplicación.

El módulo QtGui que contiene la mayoría de las clases de GUI.

El módulo de Qt que consolida las clases contenidas en todos los módulos descritos anteriormente en un único módulo. Esto tiene la ventaja de que no hay que preocuparse de qué módulo subyacente contiene una clase particular y la desventaja de que se carga el conjunto del marco Qt, aumentando así la huella de memoria de una aplicación. Si usted utiliza este módulo consolidado, o los módulos de componentes individuales es a gusto personal[19].

A continuación el código completo del editor de texto con todas sus funcionalidades:

#Librería del sistema

import sys

import os

import time

#Librería grafica PyQt

from PyQt4 import QtGui, QtCore

from PyQt4.QtCore import Qt

from PyQt4.QtGui import QTextCursor

```
#Mensaje de inicio
class MensajeInicial(QtGui.QWidget):
    def __init__(self, parent=None):
        super(MensajeInicial, self).__init__(parent)
        titulo = QtCore.QString('Creadores')
        contenido = QtCore.QString('Michell Angello Guevara Velásquez\Estefanía
          Sánchez Romero')
#Se crea el mensaje de tipo información con los campos título, contenido, un
botón de OK, logo e icono
        mensaje = QtGui.QMessageBox(self)
        mensaje.setIcon(QtGui.QMessageBox.Information)
        mensaje.setText(contenido)
        mensaje.setStandardButtons(QtGui.QMessageBox.Ok)
        mensaje.setDefaultButton(QtGui.QMessageBox.Ok)
        mensaje.setWindowIcon(QtGui.QIcon("iconos/logo.png"))
        mensaje.setWindowTitle(titulo)
        resultado = mensaje.exec_()
#Mensaje Acerca de, del menú Ayuda
class MensajeAcercade(QtGui.QWidget):
    def __init__(self, parent=None):
        super(MensajeAcercade, self).__init__(parent)
        titulo = QtCore.QString('Acerca de RoGue ODSL')
        contenido = QtCore.QString('The MIT License (MIT)\nCopyright (c)
          2015\nMichell Angello Guevara Velasquez\nEstefania Sanchez
          Romero\n\nRoGue ODSL v1.0 (UNICODE) es un editor de texto plano para
          un lenguaje de dominio específico para la determinación de problemas de
          optimización lineal')
#Se crea el mensaje de tipo información con los campos título, contenido, un
botón de OK, logo e icono
        mensaje = QtGui.QMessageBox(self)
```

```
        mensaje.setIcon(QtGui.QMessageBox.Information)
        mensaje.setText(contenido)
        mensaje.setStandardButtons(QtGui.QMessageBox.Ok)
        mensaje.setDefaultButton(QtGui.QMessageBox.Ok)
        mensaje.setWindowIcon(QtGui.QIcon("iconos/logo.png"))
        mensaje.setWindowTitle(titulo)
        resultado = mensaje.exec_()

#Ventana principal donde se crea toda la interfaz del editor de texto
class Canvas(QtGui.QMainWindow):
    def __init__(self, parent=None):
        super(Canvas, self).__init__(parent)
#Inicializa un archivo en blanco
        self.archivoTexto =""
#Variable que lleva el registro de cambios guardados
        self.cambiosGuardados = False
#Inicializa la interfaz
        self.iniciarInterfaz()

#Función donde están todas las acciones del editor de texto
    def acciones(self):
#Crea la acción nuevo y enlaza a su función nuevo
        self.accionNuevo =
          QtGui.QAction(QtGui.QIcon("iconos/nuevo.png"),"Nuevo",self)
        self.accionNuevo.setShortcut("Ctrl+N")
        self.accionNuevo.triggered.connect(self.nuevo)
```

```
#Crea la acción abrir y enlaza a su función abrir
        self.accionAbrir = QtGui.QAction(QtGui.QIcon("iconos/abrir.png"),"Abrir",self)
        self.accionAbrir.setShortcut("Ctrl+O")
        self.accionAbrir.triggered.connect(self.abrir)

#Crea la acción guardar y enlaza a su función guardar
        self.accionGuardar =
          QtGui.QAction(QtGui.QIcon("iconos/guardar.png"),"Guardar",self)
        self.accionGuardar.setShortcut("Ctrl+S")
        self.accionGuardar.triggered.connect(self.guardar)

#Crea la acción eliminar archivo y enlaza a su función eliminarArchivo
        self.accionEliminarArchivo =
          QtGui.QAction(QtGui.QIcon("iconos/eliminar.png"),"Eliminar archivo",self)
        self.accionEliminarArchivo.triggered.connect(self.eliminarArchivo)

#Crea la acción vista previa y enlaza a su función prevista
        self.accionPrevista =
          QtGui.QAction(QtGui.QIcon("iconos/prevista.png"),"Vista previa",self)
        self.accionPrevista.setShortcut("Ctrl+Shift+P")
        self.accionPrevista.triggered.connect(self.prevista)

#Crea la acción imprimir y enlaza a su función imprimir
        self.accionImprimir =
          QtGui.QAction(QtGui.QIcon("iconos/imprimir.png"),"Imprimir",self)
        self.accionImprimir.setShortcut("Ctrl+P")
        self.accionImprimir.triggered.connect(self.imprimir)
```

```
#Crea la acción copiar
        self.accionCopiar =
          QtGui.QAction(QtGui.QIcon("iconos/copiar.png"),"Copiar",self)
        self.accionCopiar.setShortcut("Ctrl+C")
        self.accionCopiar.triggered.connect(self.texto.copy)

#Crea la acción cortar
        self.accionCortar =
          QtGui.QAction(QtGui.QIcon("iconos/cortar.png"),"Cortar",self)
        self.accionCortar.setShortcut("Ctrl+X")
        self.accionCortar.triggered.connect(self.texto.cut)

#Crea la acción pegar
        self.accionPegar =
          QtGui.QAction(QtGui.QIcon("iconos/pegar.png"),"Pegar",self)
        self.accionPegar.setShortcut("Ctrl+V")
        self.accionPegar.triggered.connect(self.texto.paste)

#Crea la acción deshacer
        self.accionDeshacer =
          QtGui.QAction(QtGui.QIcon("iconos/deshacer.png"),"Deshacer",self)
        self.accionDeshacer.setShortcut("Ctrl+Z")
        self.accionDeshacer.triggered.connect(self.texto.undo)

#Crea la acción rehacer
        self.accionRehacer =
          QtGui.QAction(QtGui.QIcon("iconos/rehacer.png"),"Rehacer",self)
        self.accionRehacer.setShortcut("Ctrl+Y")
        self.accionRehacer.triggered.connect(self.texto.redo)
```

#Crea la acción seleccionar todo el contenido

```
self.accionSeleccionarTodo =
  QtGui.QAction(QtGui.QIcon("iconos/seleccionar.png"),"Seleccionar
  todo",self)

self.accionSeleccionarTodo.setShortcut("Ctrl+A")

self.accionSeleccionarTodo.triggered.connect(self.texto.selectAll)
```

#Crea la acción borrar dentro del archivo

```
self.accionBorrar =
  QtGui.QAction(QtGui.QIcon("iconos/borrar.png"),"Borrar",self)

self.accionBorrar.setShortcut("Del")

self.accionBorrar.triggered.connect(self.borrar)
```

#Crea la acción buscar y reemplazar enlazando al archivo buscar

```
self.accionBuscar =
  QtGui.QAction(QtGui.QIcon("iconos/buscaryreemplazar.png"),"Buscar y
  reemplazar",self)

self.accionBuscar.setShortcut("Ctrl+F")

self.accionBuscar.triggered.connect(Buscar(self).show)
```

#Crea la acción ocultar barra iconos y enlaza a su función ocultarHerramientas

```
self.barraHerramientas = QtGui.QAction("Barra herramientas",self)

self.barraHerramientas.triggered.connect(self.ocultarHerramientas)
```

#Crea la acción ocultar barra estado y enlaza a su función ocultarEstado

```
self.barraEstado = QtGui.QAction("Barra estado",self)

self.barraEstado.triggered.connect(self.ocultarEstado)
```

```
#Crea la acción acerca de y enlaza a su función acercaDe
        self.accionAcerca = QtGui.QAction(QtGui.QIcon(""),"Acerca de",self)
        self.accionAcerca.triggered.connect(self.acercaDe)


#Función donde se crea la barra de iconos y se agrega sus respectivos iconos
    def barraIconos(self):
#Crea la barra para los iconos
        self.barra = self.addToolBar("Barra Iconos")
#Agrega la acción nuevo a la barra de iconos
        self.barra.addAction(self.accionNuevo)
#Agrega la acción abrir a la barra de iconos
        self.barra.addAction(self.accionAbrir)
#Agrega la acción guardar a la barra de iconos
        self.barra.addAction(self.accionGuardar)
#Agrega la acción vista previa a la barra de iconos
        self.barra.addAction(self.accionPrevista)

#Agrega la acción imprimir a la barra de iconos
        self.barra.addAction(self.accionImprimir)

#Agrega separador
        self.barra.addSeparator()

#Agrega la acción copiar a la barra de iconos
        self.barra.addAction(self.accionCopiar)
```

```
#Agrega la acción cortar a la barra de iconos
        self.barra.addAction(self.accionCortar)
#Agrega la acción pegar a la barra de iconos
        self.barra.addAction(self.accionPegar)
#Agrega la acción deshacer a la barra de iconos
        self.barra.addAction(self.accionDeshacer)
#Agrega la acción rehacer a la barra de iconos
        self.barra.addAction(self.accionRehacer)
#Agrega la acción seleccionar todo a la barra de iconos
        self.barra.addAction(self.accionSeleccionarTodo)
#Agrega la acción borrar a la barra de iconos
        self.barra.addAction(self.accionBorrar)

#Agrega separador
        self.barra.addSeparator()

#Agrega la acción buscar a la barra de iconos
        self.barra.addAction(self.accionBuscar)

#Función donde se crea el menú y se agrega sus respectivos ítems
    def barraMenu(self):
#Crea barra del menú
        menubar = self.menuBar()

#Crea archivo dentro del menú
        archivo = menubar.addMenu("Archivo")
```

#Crea editar dentro del menú

editar = menubar.addMenu("Editar")

#Crea buscar dentro del menú

buscar = menubar.addMenu("Buscar")

#Crea ver dentro del menú

ver = menubar.addMenu("Ver")

#Crea ayuda dentro del menú

ayuda = menubar.addMenu("Ayuda")

#Agrega las acciones de nuevo, abrir, guardar, eliminar un archivo, vista previa e imprimir en archivo

archivo.addAction(self.accionNuevo)

archivo.addAction(self.accionAbrir)

archivo.addAction(self.accionGuardar)

archivo.addAction(self.accionEliminarArchivo)

archivo.addAction(self.accionPrevista)

archivo.addAction(self.accionImprimir)

#Agrega las acciones de copiar, cortar, pegar, deshacer, rehacer, seleccionar todo y borrar en editar

editar.addAction(self.accionCopiar)

editar.addAction(self.accionCortar)

editar.addAction(self.accionPegar)

editar.addAction(self.accionDeshacer)

editar.addAction(self.accionRehacer)

```
        editar.addAction(self.accionSeleccionarTodo)
        editar.addAction(self.accionBorrar)

#Agrega la acción buscar en buscar
        buscar.addAction(self.accionBuscar)

#Agrega las acciones ocultar barra de iconos y ocultar barra de estado en ver
        ver.addAction(self.barraHerramientas)
        ver.addAction(self.barraEstado)

#Agrega la acción acerca de en ayuda
        ayuda.addAction(self.accionAcerca)

#Inicializa la Interfaz dentro del Canvas
    def iniciarInterfaz(self):
#Crea el objeto que permite ver y editar texto
        self.texto = QtGui.QTextEdit()

#Cambia el de fondo de la ventana de texto
        self.texto.setStyleSheet("background-color: rgba(255, 255, 255); background-
          image: url(iconos/escudo.png); background-repeat: no-repeat; background-
          position: center; opacity: 10.0; ")

 #Cambia el estilo y tamaño de la letra del texto
        fuenteLetra = QtGui.QFont()
        fuenteLetra.setFamily('Arial')
        fuenteLetra.setPointSize(12)
```

```
        self.texto.setFont(fuenteLetra)

#Llama a la función de todas las acciones del editor
        self.acciones()
#Llama a la función de menú del editor
        self.barraMenu()
#Llama a la función de la barra de iconos del editor
        self.barraIconos()
#Crea una barra de estado
        self.barraEstado = self.statusBar()

#Agrega el objeto texto a la ventana principal
        self.setCentralWidget(self.texto)

# x,y coordenadas en la pantalla, ancho, alto
        self.setGeometry(50,50,1000,850)
#Nombre editor
        self.setWindowTitle("RoGue ODSL")
#Icono del editor
        self.setWindowIcon(QtGui.QIcon("iconos/logo.png"))

#Llama a la función de cursorPosicion si se detecta dentro del texto uncambio de la posición del cursor
        self.texto.cursorPositionChanged.connect(self.cursorPosicion)
#Llama a la función de cambios, si se detecta un cambio dentro del  texto
        self.texto.textChanged.connect(self.cambios)
```

```
#Las palabras reservadas del lenguaje
        palabras = ["max","min","sa"]

#Función de autocompletar una lista de palabras de Qt
        self.palabras = QtGui.QCompleter(palabras)

#Función que mueve el cursor al final del texto
        self.texto.moveCursor(QtGui.QTextCursor.End)

#Llama a setCompletar con el resultado de la función autocompletar de Qt
        self.setCompletar(self.palabras)

#Llama a la función resaltar con el texto
        Resaltar(self.texto)

#Crea una nueva ventana
    def nuevo(self):
#Crea una clase tipo Canvas y la muestra
        Canvas(self).show()

#Permite abrir un archivo con extensión .odsl
    def abrir(self):
#Guarda en un archivoTexto archivos con extensión .odsl
        self.archivoTexto = QtGui.QFileDialog.getOpenFileName(self,'Abrir
          archivo','.','(*.odsl)')
 #Verifica si el archivo existe
        if self.archivoTexto:
```

```
#Verifica si los cambios han sido guardados
        if self.cambiosGuardados:
            pass
        else:
#Crea una ventana de tipo warning con un icono, texto y los  botones Yes y No
            mensaje = QtGui.QMessageBox(self)
            mensaje.setIcon(QtGui.QMessageBox.Warning)
            mensaje.setText("El documento se va cerrar")
            mensaje.setInformativeText("Quieres guardar los cambios?")
            mensaje.setStandardButtons(QtGui.QMessageBox.Yes|
            QtGui.QMessageBox.No)
            mensaje.setDefaultButton(QtGui.QMessageBox.Yes)
            mensaje.setWindowIcon(QtGui.QIcon("iconos/logo.png"))
            mensaje.setWindowTitle("RoGue ODSL")
            resultado = mensaje.exec_()
#Verifica si el botan Yes ha sido activado
            if resultado == QtGui.QMessageBox.Yes:
                self.guardar()
            else:
                pass


#Cambia el título de la ventana principal con el nombre del archivo
        self.setWindowTitle(str(self.archivoTexto))


#Abre el archivo en modo lectura y copia el contenido en la ventana de texto
        with open(self.archivoTexto,"rt") as archivo:
```

```
            self.texto.setText(archivo.read())

#Los cambios son guardados
            self.cambiosGuardados=True

#Permite guardar un archivo con extensión .odsl
    def guardar(self):
#Abre solamente si el archivo aún no ha sido guardado
        if not self.archivoTexto:
          self.archivoTexto = QtGui.QFileDialog.getSaveFileName(self, 'Guardar
           archivo')

#Verifica si el archivo existe
        if self.archivoTexto:
#Cambia el título de la ventana principal con el nombre del archivo
            self.setWindowTitle(str(self.archivoTexto))
# Agrega la extensión odsl si aún no la tiene
            if not self.archivoTexto.endsWith(".odsl"):
              self.archivoTexto += ".odsl"
#Abre el archivo en modo escritura y copia el contenido de la ventana de
texto en el archivo
            with open(self.archivoTexto,"wt") as archivo:
                archivo.write(self.texto.toPlainText())

#Los cambios son guardados
            self.cambiosGuardados = True
```

```
#Permite borrar un archivo con extensión .odsl o limpiar el texto en su defecto
    def eliminarArchivo(self):
        #Abre solamente si el archivo aún no ha sido guardado
        if not self.archivoTexto:
            #Limpia todo el texto
          self.texto.clear()
        else:
  #Borra el archivo que se encuentra abierto
          mensaje = QtGui.QMessageBox(self)
          mensaje.setIcon(QtGui.QMessageBox.Warning)
          mensaje.setText("Esta seguro que desea borrar el archivo")
          mensaje.setStandardButtons(QtGui.QMessageBox.Yes|
          QtGui.QMessageBox.No)
          mensaje.setDefaultButton(QtGui.QMessageBox.No)
          mensaje.setWindowIcon(QtGui.QIcon("iconos/logo.png"))
          mensaje.setWindowTitle("RoGue ODSL")
          resultado = mensaje.exec_()

          if resultado == QtGui.QMessageBox.Yes:
                        os.remove(str(self.archivoTexto))
#Cambia el título de la ventana principal con el nombre del editor
                        self.setWindowTitle("RoGue ODSL")
#Limpia todo el texto
                        self.texto.clear()
#Los cambios quedan sin guardar
                        self.cambiosGuardados = False
```

```
        else:
                        pass

#Permite obtener la vista previa del documento
    def prevista(self):
#Abre una venta de dialogo de prevista
        prevista = QtGui.QPrintPreviewDialog()
#Si imprimir es activado, abre la ventana de dialogo de imprimir
        prevista.paintRequested.connect(lambda p: self.texto.print_(p))
        prevista.exec_()


#Permite imprimir el documento
    def imprimir(self):
#Abre la venta de dialogo de imprimir
        dialogo = QtGui.QPrintDialog()
#Verificar si es activado imprimir y captar el texto como documento y lo
          imprime
        if dialogo.exec_() == QtGui.QDialog.Accepted:
           self.texto.document().print_(dialog.printer())


#Permite borrar dentro del texto
    def borrar(self):
#Crea un objeto de tipo cursor sobre el texto
        cursor = self.texto.textCursor()
#Verifica si el texto ha sido seleccionado
        if cursor.hasSelection():
#Borra el texto seleccionado
```

```
            cursor.removeSelectedText()
        else:
#Borra el carácter anterior a la posición del cursor
            cursor.deletePreviousChar()

#Oculta la barra de herramientas
    def ocultarHerramientas(self):
#Almacena si la barra esta visible o no
        estado = self.barra.isVisible()
#Cambia la visibilidad a lo inverso
        self.barra.setVisible(not estado)

#Oculta la barra de herramientas
    def ocultarEstado(self):
#Almacena si la barra esta visible o no
        estado = self.barraEstado.isVisible()
#Cambia la visibilidad a lo inverso
        self.barraEstado.setVisible(not estado)

#Muestra el mensaje acerca de
    def acercaDe(self):
#Llama a la clase MensajeAcercade
        MensajeAcercade()

#Muestra la posición del cursor en la barra de estado
   def cursorPosicion(self):
#Crea un objeto de tipo cursor sobre el texto
```

```
        cursor = self.texto.textCursor()
#Crea un objeto de tipo número de bloque
        linea = cursor.blockNumber() + 1
#Crea un objeto de tipo número de columna
        columna = cursor.columnNumber()
#Muestro el mensaje del número de línea y columna en la barra de estado
        self.barraestado.showMessage("Linea: {} | Columna:
          {}".format(linea,columna))


#Detecta los cambios
    def cambios(self):
#Reinicia los cambios
        self.cambiosGuardados = False


#Detecta cuando activo el evento de cerrar la ventana principal
    def closeEvent(self,event):
#Verifica si los cambios fueron guardados
        if self.cambiosGuardados:
#Acepta el evento de cerrar la ventana
            event.accept()
        else:
#Crea una ventana de tipo warning con un icono,texto y los botones Save,
Cancel y Discard
         mensaje = QtGui.QMessageBox(self)
            mensaje.setWindowTitle("RoGue ODSL")
            mensaje.setWindowIcon(QtGui.QIcon("logo.png"))
            mensaje.setIcon(QtGui.QMessageBox.Warning)
            mensaje.setText("El documento ha sido modificado")
```

```
            mensaje.setInformativeText("Quieres guardar los cambios?")
            mensaje.setStandardButtons(QtGui.QMessageBox.Save|
            QtGui.QMessageBox.Cancel|QtGui.QMessageBox.Discard)
            mensaje.setDefaultButton(QtGui.QMessageBox.Save)
            mensaje.setWindowIcon(QtGui.QIcon("iconos/logo.png"))
            mensaje.setWindowTitle("RoGue ODSL")
            resultado = mensaje.exec_()
#Verifica si el boton Save ha sido activado
            if resultado == QtGui.QMessageBox.Save:
                self.guardar()
#Verifica si el boton Discard ha sido activado
            elif resultado == QtGui.QMessageBox.Discard:
                event.accept()
            else:
                event.ignore()


#Recibe completar que devuelve la función Completer de Qt
    def setCompletar(self, completar):
#Verifica si no se creó el objeto QCompleter de Qt
        if not completar:
            return
#Establece el Widget para que la complementación se prevé para el mismo
        completar.setWidget(self)
#Provee las palabras para completar al usuario
        completar.setCompletionMode(QtGui.QCompleter.PopupCompletion)
#Esta propiedad mantiene como se proveen las palabras para completar al
usuario
```

```
        completar.setCaseSensitivity(QtCore.Qt.CaseInsensitive)
        self.completar = completar
#Llama a insertarCompletar con el objeto Completer para que lo reciba cada que se quiera completar
        self.connect(self.completar, QtCore.SIGNAL("activated(const QString&)"),
          self.insertarCompletar)


#Inserta la palabra reservada al texto
    def insertarCompletar(self, completation):
#Crea un objeto de tipo cursor sobre el texto
        tc = self.texto.textCursor()
#Longitud de lo que se tiene que mover el cursor después de autocompletar
        extra = (completation.length() - self.completar.completionPrefix().length())
#Mueve el cursor a la izquierda
        tc.movePosition(QtGui.QTextCursor.Left)
#Mueve el cursor al final de la palabra
        tc.movePosition(QtGui.QTextCursor.EndOfWord)
#Inserta la palabra de completado
        tc.insertText(completation.right(extra))
#Se cambia el cursor al dónde queda al insertar el texto
        self.texto.setTextCursor(tc)


#Devuelve el texto seleccionado que deseo completar
    def textUnderCursor(self)
#Crea un objeto de tipo cursor sobre el texto
        tc = self.texto.textCursor()
#Selecciona el texto
        tc.select(QtGui.QTextCursor.WordUnderCursor)
```

```
        return tc.selectedText()


# Captura un evento cuando se presiona una tecla
    def keyPressEvent(self, event):
#Verifica si la lista de autocompletado es visible
        if self.completar and self.completar.popup().isVisible():
#Verifica si se activa la tecla enter, return, escape, tab, baktab en la lista que se despliega
            if event.key() in (QtCore.Qt.Key_Enter,
           QtCore.Qt.Key_Return,QtCore.Qt.Key_Escape, QtCore.Qt.Key_Tab,
           QtCore.Qt.Key_Backtab):
                event.ignore()
                return
#Verifica si Ctrl-E ha sido presionado
        atajo = (event.modifiers() == QtCore.Qt.ControlModifier and event.key() ==
          QtCore.Qt.Key_E)
#Selecciona texto a partir de la posición del cursor
        completionPrefix = self.textUnderCursor()
#Verifica si no se ha tecleado Ctrl-E y que no haya teclas modificadoras en memoria
        if (not atajo and (event.text().isEmpty())):
#La lista de autocompletado la oculta
            self.completar.popup().hide()
            return

#Verifica si el texto es diferente las palabras de la pila de autocompletar
        if (completionPrefix != self.completar.completionPrefix()):
#Verifica si el texto tiene indicios a la pila de autocompletar
            self.completar.setCompletionPrefix(completionPrefix)
```

```
            popup = self.completar.popup()
            popup.setCurrentIndex(self.completar.completionModel().index(0,0))
#Devuelve un rectángulo (en coordenadas de ventana grafica) que  incluye el cursor de la edición de texto
        cr = self.texto.cursorRect()
#Ajusta la ventana
        cr.setWidth(self.completar.popup().sizeHintForColumn(0) +
          self.completar.popup().verticalScrollBar().sizeHint().width())
        self.completar.complete(cr)


#Muestra un ventana de dialogo para las opciones de buscar y reemplazar
class Buscar(QtGui.QDialog):
    def __init__(self, parent = None):
        QtGui.QDialog.__init__(self, parent)
#Utiliza los recursos del padre (Canvas)
        self.padre = parent
#Inicializa la interfaz
        self.iniciarInterfaz()


 #Inicializa la Interfaz con los botones y campos de texto
    def iniciarInterfaz(self):
#Botón para buscar texto
        botonBuscar = QtGui.QPushButton("Buscar",self)
        botonBuscar.clicked.connect(self.buscar)

#Botón para reemplazar la última búsqueda
        botonReemplazar = QtGui.QPushButton("Reemplazar",self)
        botonReemplazar.clicked.connect(self.reemplazar)
```

```
#Campo para colocar el texto a buscar
        self.campoBuscar = QtGui.QTextEdit(self)
        self.campoBuscar.resize(250,50)

#Campo para colocar el texto por el que quiero reemplazar
        self.campoReemplazar = QtGui.QTextEdit(self)
        self.campoReemplazar.resize(250,50)

#Ventana de buscar y reemplazar
        ventanaBuscar = QtGui.QGridLayout()

#Se agrega el campo buscar a la ventana
        ventanaBuscar.addWidget(self.campoBuscar,1,0,1,4)
        ventanaBuscar.addWidget(botonBuscar,2,0,1,2)

#Se agrega el campo reemplazar a la ventana
        ventanaBuscar.addWidget(self.campoReemplazar,3,0,1,4)
        ventanaBuscar.addWidget(botonReemplazar,4,0,1,2)

#Ajusto dimensiones y nombre de la ventana
        self.setGeometry(300,300,360,250)
        self.setWindowTitle("Buscar y reemplazar")
        self.setLayout(ventanaBuscar)

#Permite buscar texto
    def buscar(self):
```

```
#Texto del Canvas
    texto = self.padre.texto.document()
#Texto del campo buscar
    textoBuscar = self.campoBuscar.toPlainText()
#Crea un objeto de tipo textCursor sobre el texto del Canvas
    cursor = self.padre.texto.textCursor()
#Posición del cursor dentro del texto
    self.posicion=cursor.position()


#Verifica si el cursor esta al principio del texto
    if(cursor.atStart() == True):
       self.posicion=0
#Verifica si el cursor esta al final del texto
    elif(cursor.atEnd() == True):
       self.posicion=0
    else:
      self.puntero=texto.find(textoBuscar,self.posicion,QtGui.QTextDocument.Find
      dCaseSensitively)
#Verifica si hay una palabra debajo de la posición actual del curso, de lo
contario pone el cursor al comienzo del editor
       if(self.puntero.position()>=0):
            pass
       else:
            self.posicion=0
# Busca en el texto la palabra distinguiendo entre minúsculas y mayúsculas

self.puntero=texto.find(textoBuscar,self.posicion,QtGui.QTextDocument.FindCase
Sensitively)
    if(self.puntero.position()>=0):
```

```
            self.comienzo=self.puntero.position()-len(textoBuscar)
#Llama a la función moveCursor con el inicio y fin de la palabra encontrada
            self.moveCursor(self.comienzo,self.puntero.position())


#Selecciona texto
    def moveCursor(self,inicio,final):
#Crea un objeto de tipo textCursor sobre el texto del Canvas
        cursor = self.padre.texto.textCursor()
#Cambia la posición al inicio de la palabra
        cursor.setPosition(inicio)
#Seleccionamos el texto a la derecha de la longitud de la palabra encontrada
         cursor.movePosition(QtGui.QTextCursor.Right,QtGui.QTextCursor.KeepAnc
          hor,final - inicio)
#Se cambia el cursor en Canvas
        self.padre.texto.setTextCursor(cursor)


#Reemplaza texto
    def reemplazar(self):
#Crea un objeto de tipo textCursor sobre el texto del Canvas
        cursor = self.padre.texto.textCursor()
#Verifica si el texto ha sido seleccionado
        if cursor.hasSelection():
#Inserta el nuevo texto
            cursor.insertText(self.campoReemplazar.toPlainText())
#Se cambia el cursor en Canvas
            self.padre.texto.setTextCursor(cursor)
```

```
#Clase que resaltar texto
class Resaltar(QtGui.QSyntaxHighlighter):
    def __init__(self, parent=None):
        super(Resaltar, self).__init__(parent)
#Formato de las palabras reservadas
        formatoPalabrasReservadas = QtGui.QTextCharFormat()
        formatoPalabrasReservadas.setForeground(QtCore.Qt.darkBlue)
        formatoPalabrasReservadas.setFontWeight(QtGui.QFont.Bold)
 #Formato de las variables
        formatoVariables = QtGui.QTextCharFormat()
        formatoVariables.setFontWeight(QtGui.QFont.Bold)
        formatoVariables.setForeground(QtCore.Qt.darkRed)
#Formato de los operadores
        formatoOperadores = QtGui.QTextCharFormat()
        formatoOperadores.setForeground(QtCore.Qt.darkMagenta)
        formatoOperadores.setFontWeight(QtGui.QFont.Bold)
#Formato de los comentarios de una sola línea
        formatoComentario = QtGui.QTextCharFormat()
        formatoComentario.setForeground(QtCore.Qt.darkGreen)
#Formato de los comentarios de varias líneas
        self.formatoCometarioMultiple = QtGui.QTextCharFormat()
        self.formatoCometarioMultiple.setForeground(QtCore.Qt.darkGreen)

#Palabras reservadas del lenguaje
        palabrasResevadas = ["\\bMAX\\b", "\\bMIN\\b", "\\bSA\\b", "\\bmax\\b",
          "\\bmin\\b", "\\bsa\\b"]
```

```
#Agrega a las reglas de resaltado las palabras reservadas

        self.reglasResaltado =
          [(QtCore.QRegExp(palabra),formatoPalabrasReservadas)

              for palabra in palabrasResevadas]


#Agrega a las reglas de las variables

        self.reglasResaltado.append((QtCore.QRegExp("\\b[0-9]*.[0-9]*[xX]+\\b"),
          formatoVariables))

        self.reglasResaltado.append((QtCore.QRegExp("\\b[0-9]*.[0-9]*[yY]+\\b"),
          formatoVariables))

        self.reglasResaltado.append((QtCore.QRegExp("\\b[zZ]\\b"),
          formatoVariables))


#Agrega a las reglas de resaltado de los operadores

        self.reglasResaltado.append((QtCore.QRegExp("\\<"),formatoOperadores))

        self.reglasResaltado.append((QtCore.QRegExp("\\<="),formatoOperadores))

        self.reglasResaltado.append((QtCore.QRegExp("\\>"),formatoOperadores))

        self.reglasResaltado.append((QtCore.QRegExp("\\>="),formatoOperadores))

        self.reglasResaltado.append((QtCore.QRegExp("\\="),formatoOperadores))


#Agrega a las reglas de resaltado de los comentarios de una línea

        self.reglasResaltado.append((QtCore.QRegExp("--
          [^\n]*"),formatoComentario))


#Indica el comienzo de un comentario de varias líneas

        self.comienzoExpresionComentario = QtCore.QRegExp("/\\*")
#Indica el final de un comentario de varias líneas

        self.finalExpresionComentario = QtCore.QRegExp("\\*/")
```

```
#Se llama cuando un bloque de texto cambia
    def highlightBlock(self, texto):
#Recorro las reglas de estado
        for palabras, formato in self.reglasResaltado:
#Captura la expresión regular de las reglas
            expresion = QtCore.QRegExp(palabras)
#Encuentra la posición de una coincidencia del texto con la expresiónregular
            inicio = expresion.indexIn(texto)
            while inicio >= 0:
#Marca la expresión que encuentra
                longitud = expresion.matchedLength()
#Cambia el formato
                self.setFormat(inicio, longitud, formato)
#Cambia el puntero del inicio después de la palabra analizada
                inicio = expresion.indexIn(texto, inicio + longitud)

        self.setCurrentBlockState(0)
        empiezaInicio = 0

#Verifica si solo tiene un bloque te texto
        if self.previousBlockState() != 1:
            empiezaInicio = self.comienzoExpresionComentario.indexIn(texto)

#Verifica si encuentra la expresión de comienzo de un comentario multi-línea
        while empiezaInicio >= 0:
            finalInicio = self.finalExpresionComentario.indexIn(texto, empiezaInicio)
```

#Verifica si no encuentra la expresión de final de un comentario multi-línea

```
        if finalInicio == -1:
            self.setCurrentBlockState(1)
            longitudComentario = len(texto) - empiezaInicio
        else:
```

#Se analiza la longitud dado que tiene más de una línea

```
            longitudComentario = finalInicio - empiezaInicio +
             self.finalExpresionComentario.matchedLength()
```

#Cambia el formato

```
        self.setFormat(empiezaInicio,
        longitudComentario,self.formatoCometarioMultiple)
        empiezaInicio =
        self.comienzoExpresionComentario.indexIn(texto,empiezaInicio +
        longitudComentario);


def main():
    app = QtGui.QApplication(sys.argv)
    main = MensajeInicial()
    main.show()
    main = Canvas()
    main.show()
    sys.exit(app.exec_())


if __name__ == "__main__":
    main()
```

10. CONCLUSIONES

La finalidad del IDE para el lenguaje de optimización lineal es resolver problemas que surgen diariamente en las empresas para disminuir costos o aumentar productividad acorde a los recursos que tenga la organización, sean de tiempo, material o humano. Este editor es una primera parte de este proceso que pretende resolver dichos inconvenientes.

El proyecto realizado ha contribuido de manera importante en la formación de nosotros como estudiantes, dado que nos permitió aplicar los conocimientos adquiridos a lo largo de la carrera. En este proyecto se abarcaron grandes áreas como son la investigación de operaciones para el modelado matemático, la programación, la definición de expresiones regulares y gramáticas, el diseño de diagramas UML y metodologías de desarrollo ágil. Estas se pueden definir como herramientas que contribuyeron a la creación de un editor de texto acorde a las necesidades y restricciones de tiempo. La meta es poder tener un entorno de desarrollo integrado de un lenguaje de optimización lineal y desarrollado por estudiantes de la Universidad Tecnológica de Pereira.

Muchos de nosotros como desarrolladores nos ha tocado entregar productos, pero durante dicho proceso nos hemos enfrentado a cambios imprevistos, debido a que no se tienen objetivos claros desde el comienzo o porque de alguna manera varían por el ambiente externo, un cambio implica mucho trabajo extra y/o genera costos demasiado altos. El éxito de un proyecto se debe a que se tenga claro lo que se quiere, es decir, que los objetivos sean precisos desde el inicio. El cliente es la persona que tiene más claro lo que se desea, pero como no hace parte de equipo de desarrollo es necesario intermediar con él, pero si esta comunicación falla, nuestro proyecto fracasa, por esto y por muchos otros factores surgió SCRUM como un proceso rápido, iterativo e incremental para el desarrollo de software.

Los diagramas UML son una gran herramienta para visualizar, especificar, construir y documentar un sistema, nos ofrece tres grandes perspectivas. La primera es la estructura la cual indica como cada parte del sistema está distribuida para formar un todo, la segunda perspectiva es el comportamiento que permite ver la forma de proceder de las partes en relación con sus funciones y por último la perspectiva de interacción, acción recíproca entre los objetos. Sin la ayuda de estos diagramas el producto final tomaría más tiempo del previsto y con comportamientos impredecibles.

BIBLIOGRAFIA

MILANO, Matt. ¿Cuáles son las ventajas de un editor de texto?. {En línea}. {28 Abril de 2015}. Disponible en http://www.ehowenespanol.com/cuales-son-ventajas-editor-texto-info_310589

Florez Fernandez, Héctor A. Interpretación dinámica de múltiples lenguajes de dominio específico Facultad Tecnológica, Universidad Distrital Francisco José de Caldas. Bogotá.

Booch, Grady. Object-Oriented Analysis and Design with Applications. Third Edition

Sánchez Cuadrado, Jesús. A Model-Based Approach to Families of Embedded Domain-Specific Languages. Software Engineering, IEEE Transactions on Volume.35, No 6, 2009.

Marjan Mernik, Viljem Zumer. Domain-Specific Languages for Software Engineering. The University of Maribor, 2002

MÉTODOS DE DESARROLLO ÁGILES, Lenguajes y Sistemas Informáticos [diapositivas]. UNIVERSIDAD DE GRANADA. 2014 49 diapositivas.

DESARROLLO ÁGIL DE SOFTWARE, SCRUM, UNIDAD 1, Lenguajes y Sistemas Informáticos [diapositivas]. UNIVERSIDAD DE GRANADA. 2014 85 diapositivas.

DESARROLLO ÁGIL DE SOFTWARE, SCRUM, UNIDAD 1, Sem3. Historias de Usuario, Lenguajes y Sistemas Informáticos [diapositivas]. UNIVERSIDAD DE GRANADA. 2014 73 diapositivas.

DESARROLLO ÁGIL DE SOFTWARE, SCRUM, UNIDAD 1, Sem6. Ejemplo de "Plan de Entregas", Lenguajes y Sistemas Informáticos [diapositivas]. UNIVERSIDAD DE GRANADA. 2014 21 diapositivas.

DESARROLLO ÁGIL DE SOFTWARE, SCRUM, UNIDAD 1, Sem7. Ejemplo de "Plan de la 1er Iteración", Lenguajes y Sistemas Informáticos [diapositivas]. UNIVERSIDAD DE GRANADA. 2014 10 diapositivas.

[1] Wilson, Leslie B. (1993). Comparative Programming Languages, Second Edition. Addison-Wesley. p. 75

[2] Parr Terrence. The definitive ANTLR Reference, Building Domain Specific Languages. United States of America, 2007

[3] Parr Terrence. Language implementation patterns, Create Your Own Domain-Specific and General Programming Languages. United States of America, 2010.

[4] García Enrique, Troyano José. Guía práctica de ANTLR. 2003

[5] XTEXT. {En línea}. {10 Abril de 2014}. Disponible en http://www.eclipse.org/Xtext/documentation

[6] JavaCC. {En línea}. {10 Abril de 2014}. Disponible en http://javacc.java.net

[7] Rehman, Christopher Paul, Christopher R. Paul. "The Linux Development Platform: Configuring, Using and Maintaining a Complete Programming Environment". 2002.

[8] PROGRAMACION LINEAL. {En línea}. {10 Mayo de 2015}. Disponible en http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/programaci%C3%B3n-lineal/

[9] PROGRAMACIÓN LINEAL EN WINQSB. {En línea}. {10 Mayo de 2015}. Disponible en http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/programaci%C3%B3n-lineal-en-winqsb/

[10] PROGRAMACIÓN LINEAL EN SOLVER. {En línea}. {10 Mayo de 2015}. Disponible en http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/programaci%C3%B3n-lineal-en-solver/

[11] PROGRAMACIÓN LINEAL EN TORA. {En línea}. {10 Mayo de 2015}. Disponible en http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/programaci%C3%B3n-lineal-en-tora/

[12] PROGRAMACIÓN LINEAL EN LINGO. {En línea}. {10 Mayo de 2015}. Disponible en http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/programaci%C3%B3n-lineal-en-lingo/

[13] SWEBOOK. California, Versión 2004, IEEE Computer society.

[14] Python. {En línea}. {10 Abril de 2014}. Disponible en https://www.python.org/

[15] Notepad++. {En línea}. {10 Abril de 2014}. Disponible en http://tecnicia.blogspot.com/2011/03/notepad-primera-parte.html

[16] Gedit. {En línea}. {10 Abril de 2014}. Disponible en http://www.freesoftwaremagazine.com/articles/gedit_powerful_underrated_text_editor_everybody

[17] PyQt4. {En línea}. {4 Marzo de 2015}. Disponible en https://pypi.python.org/pypi/PyQt4

[18] Elementos de UML. {En línea}. {10 Noviembre de 2014}. Disponible en https://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html

[19] PyQt4 Components {En línea}. {2 Mayo de 2015}. Disponible en http://pyqt.sourceforge.net/Docs/PyQt4/introduction.html#pyqt4-components

LISTA DE TABLAS

LISTA DE ILUSTRACIONES