PERCEPCIÓN UML

Estudiante

A.P.U. PEREZ, Nicolás Ignacio

Director

Mg. AMATRIAIN, Hernán

TRABAJO INTEGRADOR PRESENTADO PARA OBTENER EL TÍTULO
DE
LICENCIADO EN SISTEMAS

DEPARTAMENTO
DE DESARROLLO PRODUCTIVO Y TECNOLOGICO
UNIVERSIDAD NACIONAL DE LANUS

MARZO, 2019

AGRADECIMIENTOS

De corazón, no paro de dar las gracias a mis tutores y asesores de este trabajo; a Hernán Amatriain, inmensamente agradecido por su paciencia, dedicación, criterio, aliento y sobre todo por esta hermosa motivación que me ha penetrado en la mente para la realización de este proyecto a largo plazo. A la profesora Laura Loidi por tomarse el trabajo de leer y releer este trabajo para hacerme sugerencias.

Agradecimientos totales a todo el personal que hace a esta maravillosa Universidad, principalmente por la atención, amabilidad y el cálido ambiente de trabajo que me brindan tanto a mi como a todos los estudiantes, graduados, docentes, no docentes y trabajadores de la UNLa.
Gracias a todos aquellos que han podido colaborar de mil formas con este trabajo: estudiantes, compañeros y familiares. Tampoco me quiero olvidar de la directora Alejandra Vranic y el director Alejandro Tornay que me han involucrado de lleno en el precioso mundo de la docencia en esta institución.

Gracias a mis amigos, compañeros de trabajo, y allegados a los que les he robado muchas horas de compañía e infinitas actitudes poco amables de mi parte por priorizar siempre este software y obtención del título.

Y desde ya, y por encima de todo a los míos, madre, padre y Gri por estar de forma incondicional.

RESUMEN

Este Proyecto se basa en la creación de una aplicación web que permite crear diagramas de clases (DC) despreocupándonos del diseño estético, haciendo hincapié únicamente en su estructura interna.

Está diseñado para que se pueda trabajar de una forma colaborativa, ágil y rápida.

Este Software posee dos aristas fundamentales, la primera nos permite crear dichos DC por medio de una interfaz simple, para generar no solo una imagen, sino también un texto, audio y un protocolo que se pueda compartir con compañeros no videntes. Mientras que la segunda arista es el camino inverso, por medio de un metalenguaje permitir, con un simple texto, que se puedan generar DC para compartir entre los trabajadores y estudiantes de sistemas.

ABSTRACT

This Project is based on the creation of a web application that allows to create class diagrams (DC) without worrying about the aesthetic design, emphasizing only its internal structure. It is designed so that it can be worked in a collaborative, agile and fast way.

This software has two fundamental edges, the first one allows us to create these DCs through a simple interface, to generate not only an image, but also a text, audio and a protocol that can be shared with blind colleagues. While the second edge is the inverse path, by means of a metalanguage allow that through a simple text DC can be generated to share between workers and system students.

ÍNDICE

1. INTRODUCCIÓN 11

11- Contexto de la tesis

13- Objetivo de la tesis

14- Visión general de la tesis

15- Visión general de los resultados obtenidos por el TFL

2. ESTADO DEL ARTE 16

17- Presente de la problemática

18- Teoría de protocolos

19- Bases teóricas sobre el desarrollo de web accesible

22- Nuestro protocolo

3. SOLUCIÓN 27

28- Metodología de la investigación y elaboración de la solución

29- Proceso de selección de un MCV

33- Proceso de iniciación, planificación y estimación del proyecto

39- Proceso de seguimiento y control del proyecto

43- Proceso de gestión de calidad del SW

48- Proceso de exploración de conceptos

55- Proceso de asignación del sistema

61- Proceso de análisis de requisitos

64- Proceso de diseño

69- Proceso de implementación e integración

81- Proceso de verificación y validación

88- Proceso de gestión de la configuración

90- Proceso de desarrollo de la documentación

91- Proceso de formación

4. CONCLUSIONES **92**

92- Conclusiones del creador

5. REFERENCIAS **95**

6. ANEXO – MANUAL DE USUARIO **97**

NOMENCLATURA

AP	Análisis de Protocolo
APP	Aplicación
BD	Base de datos
CU	Casos de uso
DC	Diagrama de clases
DCU	Diagrama de casos de uso
DER	Diagrama de Entidad relación
DFD	Diagrama de flujo de datos
LDC	Líneas de código
MVC	Modelo de ciclo de vida
OO	Orientación a Objetos
RF	Requisitos funcionales
RNF	Requisitos no funcionales
RS	Requisitos de software
RU	Requisitos de Usuario
SW	Software
TFL	Trabajo Final de Licenciatura
UML	Lenguaje Unificado de Modelado
UNLa	Universidad Nacional de Lanus
W3C	World Wide Web Consortiun
WAI	Web Accessibility Initiative

ÍNDICE DE FIGURAS

27- Figura 1: DC generado por Percepción UML

33- Figura 2: Ciclo de vida prototipado evolutivo

34- Figura 3: COCOMOII

42- Figura 4: PERT con camino critico

52- Figura 5: DC para determinar prototipos viables

53- Figura 6: Prototipo desechable

56- Figura 7: Opciones de Percepción UML

57- Figura 8: Modificar DC ya existente

57- Figura 9: Ver tu DC

58- Figura 10: Crear un nuevo DC

58- Figura 11: Maquetado

59- Figura 12: Ver clase especifica

59- Figura 13: Web en su nuevo estilo simplificado

60- Figura 14: Diagrama de contexto

61- Figura 15: Diagrama de casos de uso

63- Figura 16: Casos de usos esenciales

64- Figura 17: Diagrama de escenarios principales

65- Figura 18: Diagrama HIPO

66- Figura 19: Diagrama de entidad relación

70- Figura 20: Casos de pruebas

76- Figura 21: Prueba de stress

81- Figura 22: Mejora continua

87- Figura 23: Prueba de rendimiento

ÍNDICE DE TABLAS

35- Tabla 1: Mapa de actividades

40- Tabla 2: Gestión de riesgos

44- Tabla 3: Niveles de calidad del SW

45- Tabla 4: Métricas de McCall

46- Tabla 5: Métricas del maquetado

46– Tabla 6: Métricas prototipo evolutivo

47- Tabla 7: Métricas de versión final

69- Tabla 8: Interfaz grafica

77- Tabla 9: Casos de pruebas

78- Tabla 10: Prueba de interfaz web

85- Tabla 11.a: Casos de pruebas y sus impactos

86- Tabla 11.b: Casos de pruebas y sus impactos

1. INTRODUCCIÓN

En este Capítulo se plasman los motivos por los cuales se realizó este Trabajo Final de Licenciatura (TFL), quienes fueron los principales participes, como surgió el proyecto y como esperamos que este trabajo sea el primero de muchos que ayuden a fortalecer las raíces y frutos de este proyecto de accesibilidad en la carrera de sistemas. Y fundamentalmente como ha quedado el proyecto al finalizar este trabajo.

1.1. CONTEXTO DE LA TESIS

Este trabajo surge, principalmente, de la mano del inconmensurable trabajo que realiza día a día la Universidad Nacional de Lanús - UNLa para lograr una mayor inclusión, para que todos podamos estudiar lo que nos gusta y en un lugar por demás hermoso.
Allá por el 2013 la Universidad se topó con el ingreso de un nuevo estudiante (Cristian), estudiante que nos alimentaria de gran forma a todos nosotros con las metas, vallas y desafíos que nos ha regalado.

Una vez que Cristian ingresó a la Universidad, a la Licenciatura en Sistemas, para ser precisos, se tornó evidente de que gran parte del dictado de la carrera, programas y métodos de enseñanza están únicamente estructurados para personas sin disminuciones en alguno de los sentidos. Cristian, tenía la particularidad que no podía ver, característica que no lo detuvo a la hora de querer seguir con sus estudios, así que nos preguntamos, “Si, él no se detiene. ¿Por qué vamos a detenerlo nosotros con nuestra forma de enseñar?”.

Gran parte de los docentes hicieron esfuerzos muy sobresalientes para adaptar sus materias de forma que Cristian pudiera mostrar sus aptitudes, aprender, dar exámenes, aprobar, desaprobar, etc.; Es decir para que Cristian pudiera tener una vida como la de cualquier otro alumno. Se incorporaron muchas tecnologías ya existentes, se le brindaron horas extras de estudio, se le dio apoyo de todo tipo, pero nada de esto fue gratis, él nos pagaba día a día con todo su esfuerzo y responsabilidad, los dos principales motores que permitieron que estas ganas de mejorar no se detengan, todo gracias a su voluntad.

Llego el momento en el cual Cristian llegaba a las materias meramente de sistemas, programación en todos sus niveles, la rama de ingeniería en software, las materias teóricas de sistemas y las

materias un poco más específicas. Es allí cuando surge la idea y motivación de realizar un software que permita a Cristian no solo interactuar con sus compañeros a la hora de debatir un programa, un diseño o una posible resolución de un problema, sino también, a futuro, ayudarlo a poderse zambullir en el ámbito laboral, donde él deberá defender sus ideas, debatir sobre las de otros y modificar antiguas ideas de terceros.

Hernán Amatriain tomo por las astas este desafío y empezó el camino que llevaría a un horizonte repleto de herramientas que facilitarían a Cristian y a cualquier otro estudiante o trabajador de sistemas a superar estos escoyos. La primera idea que surgió fue crear un mecanismo que permita crear diagramas de entidad relación, y desencriptarlos en un texto de fácil interpretación que pueda ser leído por un software de manera que cualquier usuario pueda "percibir" el DER por medio de un sonido en un lenguaje seudotécnico. Dicho trabajo fue tomado por otro tesista del cual esperamos ansiosos su desempeño que de seguro será altamente útil.

Luego, aparecí yo, enterándome de esta situación en una de las clases de Hernán, y decidí introducirme a este proyecto. Partimos de lo antes mencionado sobre el desarrollo de un DC decodificado en un audio, lo que hoy yo llamo PERCEPCION UML.

Como último aspecto a detallar en este apartado, un aspecto bastante personal e incluso hasta egoísta, es mencionar otro factor clave que puso en contexto a mi trabajo. Por "casualidad" en mi ejercicio como docente en la UNLa, en estos dos años que diagrame este proyecto me tope con 3 alumnos, los cuales estaban en situaciones muy parecidas a las de Cristian, alumnos que no mencionaré específicamente en este trabajo, pero quiero resaltar ya que fueron de gran ayuda para estructurar este trabajo, Tomas y Agustin de la Licenciatura en Tecnologías Ferroviarias y Luciano de la Licenciatura en Sistemas. Ellos, como alumnos con dificultades visuales me han planteado cosas sobre las clases en general que me han ayudado a mejorar aspectos de este trabajo.

También quiero resaltar brevemente el trabajo vivaz que realiza la Dirección de Bienestar Universitario, que, junto al Programa de Inclusión Universitaria para Personas con Discapacidad, me han dado un importante carril por el cual circular para lograr lo que plantea PERCEPCION UML.

1.2. OBJETIVO DE LA TESIS

Percepción UML, es una aplicación web, que permite a cualquier persona crear DC, plasmados en una base de datos, y poder seguir trabajando sobre esta BD en cualquier otro lugar, crear distintas versiones y modificaciones de este DC, compartirlos por mail, permitir que cualquier persona con un usuario y contraseña puedan acceder a la misma para que sigan trabajando o para que simplemente perciban el diseño.

Ahora bien, todo lo dicho en el párrafo superior no parece ser nada nuevo ni ligado a la introducción sobre Cristian y el desafío que queríamos afrontar. La particularidad de Percepción UML es que el DC es transformado a audio y a texto, de manera que una persona puede crear un DC y cualquier otra (o la misma) puede ver el DC, "escucharlo" o "leerlo", lo que de ahora en más diremos "percibirlo". En un principio también habíamos pensado generar también dichos diagramas en Braille, pero esta funcionalidad dejo de estar en un primer plano y será dejada para futuros trabajos o para integrarlo a este o cualquier otro en futuros proyectos.

Otra de las "virtudes" que tiene Percepción UML, es que el usuario solo tiene que encargarse de la lógica y estructura interna del DC, no tiene que preocuparse por la estética. ¿Qué quiere decir esto?, muchas veces hemos usado otras tecnologías que nos permiten crear un DC, pero siempre nos topábamos con lo mismo: "Uh, esta clase tiene mil relaciones, ¿Dónde la pongo?, ¿Se entiende así?, son demasiadas Clases ¿cómo las distribuyo?", todas estas preguntas y pseudoproblemas de diseño están resueltos por la web de Percepción UML, integrando la tecnología "open source, PlantUML", de manera que el programador o analista no tendrá que gastar energías en la parte visual o estética, solamente deberá enfocarse en lo que es realmente importante, el diseño interno, además la percepción no puede estar ligada a una imagen y no puede depender de la ubicación ni de las clases, ni de ningún otro componente.

De manera que se puede dar un grosero resumen de cuáles fueron los objetivos de este trabajo:

- Crear una plataforma que permita a los docentes o trabajadores de Sistemas crear DC en un formato que pueda compartirse con personas de capacidades visuales disminuidas.
- Crear una plataforma que le permita a personas ciegas o de capacidades visuales disminuidas a generar DC visuales, por medio del teclado, con un lenguaje de fácil asimilación y simple a la hora de tipear.

- Poder generar los diagramas en Braille.
- Transformar los DC en audios de fácil manipulación.
- Permitir un control de versiones super ágil para que todos puedan trabajar sobre un mismo DC.
- Integrar todos estos objetivos en una aplicación WEB disponible desde cualquier dispositivo con conexión a internet.

1.3. VISIÓN GENERAL DE LA TESIS

A modo de resumen, Percepción UML, no es más que una web de armado de Diagramas de Clases, que con solo darle las Clases, Atributos, métodos y relaciones la web genera una imagen del DC, una descripción auditiva del DC y una descarga un texto en lenguaje coloquial de este.

Además, el software dispone de una muy amigable forma de controlar las versiones y últimos movimientos sobre un diagrama de clases. La creación del DC se traduce a escribir muy poco, de manera que facilita la tarea de alguien que no puede apoyarse en su visión; de forma que con un simple clic puede elegir entre opciones preestablecidas, y así facilitar la navegación web.
No se puede dejar de subrayar también la otra faceta de este programa que es generar el diagrama de clases por medio del pseudo-lenguaje o protocolo para la creación de DC. Cómo se diagramó este protocolo, cómo se lo probó y cómo se lo pulió se verá en los capítulos posteriores; al igual que la generación del texto y el audio, los mecanismos para generar este texto fueron desarrollado entre varios docentes y estudiantes avanzados de la carrera.

Para llevar a cabo este nuevo proyecto hemos encarado el desafío de acuerdo con el Estándar IEEE 830-1998, por medio del paradigma de Orientación a Objetos y lenguaje Java, utilizando un patrón MVC, con la parte visual desarrollada en HTML5, JavaScript y JSP, los datos son guardados en una base de datos con distribución MYSQL hospedada en un servidor brindado por Jelastic; dichos procesos y etapas fueron diagramadas a lo largo de un año y medio, los cuales serán descriptos en esta documentación.

1.4. VISIÓN GENERAL DE LOS RESULTADOS OBTENIDOS POR EL TFL

La mejor manera de evaluar o comentar los resultados de un trabajo, es plasmando la relación entre los objetivos y el desenlace, de manera que a continuación se dejan los objetivos iniciales del trabajo y cómo fueron o no cumplidos y/o modificados.

a. Crear una plataforma que permita a los docentes o trabajadores de Sistemas generar DC en un formato que pueda compartirse con personas de capacidades visuales disminuidas:

Este fue el objetivo primordial del trabajo, se puede compartir información gráfica en formato texto, protocolo y audio con cualquier usuario.

b. Crear una plataforma que le permita a personas ciegas o de capacidades visuales disminuidas generar DC visuales, por medio del teclado, con un lenguaje de fácil asimilación y simple a la hora de tipear:

Este objetivo fue cumplido puesto que se generó una web accesible, con la posibilidad de crear una imagen por medio de un protocolo, esta imagen se puede compartir con cualquier trabajador o estudiante de sistemas, esta web no necesita el uso del mouse, se puede acceder a todas las opciones por medio de atajos de teclado o por medio de la navegación con las teclas CONTROL, SHIFT, TAB y ENTER.

c. Poder generar los diagramas en Braille.

Si bien el Braille es muy utilizado y útil aún en estos años, las primeras entrevistas con Cristian revelaron que suele ser mucho más útil tener un buen documento en formato txt, ya que hoy por hoy se tienen muchas tecnologías para manipular esto. Además, sin transcribirla el Braille de nuestro producto dejamos de lado un problema que sigue siendo trabajado, que es la dificultad para escribir un texto en Braille, es muy simple reemplazar cada letra por su equivalente en Braille, pero es muy complejo hacer que el receptor comprenda dicho texto con la cohesión y coherencia que le brinda un escrito u comunicación verbal. Por ejemplo, una de las principales desventajas de este gran sistema es la falta de símbolos, difícil aprendizaje, velocidad de lectura muy lenta, etc.

d. Transformar los DC en audios de fácil manipulación.

Si bien nuestro SW, brinda un audio para reproducir por internet, o incluso para descargar, este objetivo quedó en segundo plano, puesto que se llegó a la conclusión que el texto es mucho mas rico, y en caso de que algún usuario desee una devolución auditiva se les brinda un texto en un español coloquial que se reproducción es muy natural.

e. Permitir un control ágil de versiones para que todos puedan trabajar sobre un mismo DC.

Este objetivo no era primordial, pero fue alcanzado en la versión final de nuestro producto. Permite que cualquier usuario que tengo la contraseña de un DC pueda trabajar sobre él.

f. Integrar todos estos objetivos en una aplicación WEB disponible desde cualquier dispositivo con conexión a internet.

Todo el potencial brindado por la aplicación esta disponible, provisoriamente en:

http://node24730-uml-dc.jelastic.saveincloud.net/UML-DC/

Con la particularidad que esperamos que próximamente este disponible en el servidor de la Universidad Nacional de Lanus.

2. ESTADO DEL ARTE

En este capítulo se presenta el estado de la cuestión sobre distintas teorías acordes a este proyecto. Se presenta la situación de esta problemática, como fue abordada en los últimos años, Presente de la problemática (sección 2.0, Teoría de Protocolos (sección 2.1), Bases Teóricas sobre el desarrollo web accesible (sección 2.2) y Nuestro protocolo (sección 2.3).

2.0 PRESENTE DE LA PROBLEMÁTICA

En la sociedad actual, la universidad es una de las opciones que muchos jóvenes eligen al finalizar sus estudios obligatorios, por lo que debería ser también una posibilidad factible para cualquier joven con discapacidad. Si bien es cierto que cada vez más jóvenes con discapacidad acceden a estudios universitarios y finalizan sus carreras con éxito, no siempre encuentran los apoyos o servicios necesarios para superar las dificultades de un sistema diseñado sin pensar en personas con capacidades diferentes.

En este contexto, el término inclusión se refiere a la participación en la escuela común de las personas con discapacidad y aquellas con necesidades educativas especiales. La Unesco define la educación inclusiva como un proceso orientado a considerar a la diversidad de los estudiantes, incrementando su participación y reduciendo la exclusión en y desde la educación.

La mayoría de los países del mundo han adoptado en sus políticas y leyes los principios de la Declaración de Educación para Todos, pero en la práctica es posible constatar que la educación es para casi todos o para la mayoría y que los excluidos son, precisamente, quienes más necesitan de ella para compensar su situación de desventaja social.

En esta situación nos encontramos ahora. ¿Qué herramientas existen para la educación inclusiva? ¿Qué herramientas necesitamos? Y, ante todo, qué herramientas existen o son necesarias para lograr la accesibilidad e inclusión en la carrera de sistemas, particularmente en nuestra universidad.

En la actualidad hay disponibles varios softwares que permiten hacer más fácil el estudio para personas con discapacidades visuales, como, por ejemplo, el Gestor ONCE de Libros Digitales (GOLD), que posibilita la descarga y gestión accesible de libros digitales en formato Daisy (sistema que permite desplazarse y escuchar las opciones); el Comunicador Táctil ONCE (CTO), esta aplicación les facilita emitir mensajes de manera presencial usando el sistema dactilológico, e incluso disponer de frases ya hechas para distintas situaciones cotidianas; Be my eyes, es una aplicación que permite interactuar a la persona con discapacidad visual con cualquier otra persona, es una buena herramienta para aquellas personas con problemas de visión que necesitan solicitar ayuda remota mediante una videollamada; Super Visión para CardBoard, dirigida a personas con baja visión, esta app funciona como una lupa virtual que permite hacer del smartphone un

magnificador electrónico, con ella los estudiantes de poca visión pueden leer textos y ver objetos lejanos con mayor nitidez.

Y así podría proseguir con varias aplicaciones que hacen más ameno el estudio de alumnos ciegos. ¿Pero cuántas aplicaciones encontramos vinculadas en el estudio de Sistemas, Informática o Ciencias de la Computación?
Allí el universo de productos se ve muy acotado, de hecho, casi nulo para la creación de diagramas o códigos fuentes.
Existen varias apps como NVDIA o Window-Eyes, que permiten navegar la pantalla por medio de sonido, pero nada que ayude a crear tus propios diagramas o líneas de código. Hacia ese territorio nos embarcamos con este trabajo.

2.1 TEORIA DE PROTOCOLOS

Para poder resolver el problema planteado de antemano y en los capítulos siguientes, es notorio que se tuvo que pasar por varios protocolos o reglas de texto para generar información. Entonces surge la necesidad de definir que es en verdad un protocolo y como utilizarlo.

Protocolos pueden ser de muchos tipos, desde los clásicos protocolos sociales, protocolos económicos, protocolos criptográficos o muchos otros que se nos puedan ocurrir. En este apartada hablaremos de los protocolos de comunicación, donde nos referimos a comunicación, en no solo el pasaje de bite a bite, sino algo mucho más ligado al ser humano, como transferirnos información, en nuestro caso, visual, a texto o viceversa.

Un **protocolo de comunicaciones,** como afirma Licesio J. Rodríguez-Aragón en Internet y Teleinformática: "es un sistema de **reglas** que permiten que dos o más entidades de un sistema que se comuniquen entre ellas para transmitir información por medio de cualquier tipo de variación de una magnitud física. Se trata de las **reglas o el estándar** que define la **sintaxis, semántica y sincronización** de la comunicación, así como también los posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos".

Entonces, haciendo referencia a este último párrafo, para poder compartir diagramas de clases entre personas no videntes y cualquiera de todos nosotros, vamos a necesitar definir con total certeza reglas, sintaxis, semántica, una sincronización; es decir estandarizar nuestra comunicación. De esto se encargará PERCEPCIÓN UML.

El primer problema que solucionar es el de construir un protocolo que transforme nuestro texto a una imagen de un DC, esto claramente ya está solucionado por un sin fin de softwares, ahora bien, si uno se plantea o pregunta que tan simples son esos protocolos se puede apreciar que carecen de naturalidad, tienen una sintaxis compleja y difieren mucho entre sí, además que muchas veces las herramientas para crear DC son meramente visuales. Esto lo trabajará PERCEPCIÓN UML, brindando una interfaz simple para construir estos DC dejando de lado la parte visual en su construcción.

Una vez superado el escollo de la imagen, surge la pregunta de cómo comentar dicha imagen, o como transferirla a una persona no vidente, aquí nuevamente PERCEPCIÓN UML nos dará una solución, por medio de esta aplicación, se transformará las imágenes a un texto coloquial, a un audio de fácil interpretación y además a un protocolo simple y comprensible para reconstruir esta imagen.

Entonces a modo de resumen y sin más preámbulos, PERCEPCIÓN UML trabajara con tres protocolos, uno para la creación de imágenes por medio de una interfaz visual, para esto usare el protocolo trabajado por PLANTUML.COM, otro protocolo para crear imágenes con un lenguaje coloquial y un último protocolo que transforma las imágenes a un texto de muy fácil interpretación.

2.2 BASES TEÓRICAS SOBRE EL DESARROLLO DE WEB ACCESIBLES

Construir este software depende íntegramente de la construcción de una página web para dos grandes usuarios, los primeros son aquellos que no tienen ninguna imposibilidad óptica por lo cual la web podría considerarse estándar o sin grandes cuestiones a tener en cuenta, y luego una interface web para Cristian o para cualquier otra persona allegada a sistemas que quiera usar nuestro protocolo para compartir DC. Para esta última interfaz es cuando hay que toparse totalmente con el desarrollo accesible.

Según Lic. Gabriela A. Toledo en Manual de prácticas de accesibilidad digital: "La ´accesibilidad´ puede definirse como la condición a cumplimentar por los entornos, procesos, bienes y servicios, de manera que resulten comprensibles, utilizables y practicables en condiciones de seguridad y comodidad, de la forma más autónoma y natural posible.

La accesibilidad Web consiste en un acceso universal a la Web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y **capacidades de los usuarios."**

Entonces, este SW debe respetar al máximo algunas cuestiones de accesibilidad para que pueda sacar adelante los objetivos básicos de este trabajo.

Para esto es fundamental comprender los principios básicos para un desarrollo estandarizado que torne accesible una aplicación.

Por esto, la Lic. Gabriela A. Toledo (2011, p.10), en su Manual de prácticas de accesibilidad digital. Recomendaciones para facilitar las páginas Web a las personas con limitaciones en la visión, nos define que:

> Considerar la e-accesibilidad al desarrollar un instrumento informático implica advertir las ventajas derivadas de entender y construir el ciberespacio según los principios del Diseño Universal, entre los que puedo destacar:
>
> - Cumplir al derecho ciudadano a la participación y no discriminación por razón de discapacidad.
> - Respetar las disposiciones legislativas internacionales y nacionales.
> - Acrecentar el número de usuarios potenciales, con mayor alcance de la comunicación, servicios o mercado.
> - Garantizar la equivalencia de los contenidos entre distintos navegadores y dispositivos, pues se diseña considerando estándares generales de accesibilidad.
> - Mejorar la indexación en los motores de búsqueda. El cumplimiento de las pautas, tanto en código como en contenidos semánticos (por ejemplo, la presentación de vínculos o enlaces) con sentido permite a los motores de búsqueda una mejor identificación de la información, y, en consecuencia, mayores posibilidades de posicionamiento en los buscadores.

En Argentina, entre las razones por las cuales la accesibilidad se evalúa como una necesidad pueden nombrarse las de índole ético, social, político, económico y legal, así como también las propuestas por la Usabilidad Web, como encontrabilidad (findability), funcionalidad, utilidad y credibilidad.

Para mi facilidad y la de este proyecto existen organismos encargados de tipificar estos asuntos. El Consorcio de la Word Wide Web (W3C - World Wide Web Consortium) es una organización internacional donde las entidades miembros, personal a tiempo completo y público en general, trabajan conjuntamente para desarrollar estándares Web. Está dirigido por el creador de la Web Tim Berners-Lee. Su misión es lograr de la Web su máximo potencial. Es la entidad encargada de proponer las Pautas de accesibilidad.

A partir de 1998 comienza a desarrollarse la Iniciativa de Accesibilidad a la Web (WAI – Web Accesibility Initiative) que se enfoca en extender los protocolos y formatos de datos para hacer la Web más accesible.
El trabajo de la WAI se centra en el desarrollo de las pautas, mejora de herramientas para la evaluación y reparación de accesibilidad Web.
Lleva a cabo una labor educativa y de concientización en relación con la importancia del diseño accesible de páginas Web, abriendo nuevos campos en accesibilidad a través de la investigación.

Para ir resumiendo, entonces, se puede definir que: **Una página web es accesible si** su contenido puede ser operado y recibido de múltiples modos y si cumple con las pautas propuestas por la WAI. Son 14, conformadas por 65 puntos de verificación que ayudan a detectar posibles errores, divididos en tres niveles de accesibilidad.

Prioridad 1 o Nivel A: Puntos que TIENE que cumplir un desarrollo Web para que uno o más grupos de usuarios encuentren posible acceder a su sitio Web.
Prioridad 2 o Nivel AA: Puntos que DEBE cumplir un desarrollo Web para que uno o más grupos de usuarios no encuentren dificultades para acceder al sitio.
Prioridad 3 o Nivel AAA: Puntos que PUEDE cumplir un desarrollo Web para que uno o más grupos de usuarios no encuentren dificultad para acceder a la página.

2.3 NUESTRO PROTOCOLO

Esta entonces claro que, la web de PERCEPCION UML, debe ser accesible, por lo menos su parte desarrollada para la transformación de texto a imagen, y que nos debe brindar dos protocolos principales y uno secundario.

a- Un protocolo principal que permita guardar los inputs generados por la interfaz gráfica, para guardar los dato en la base de datos y luego poderlos transformar a texto, audio, imagen y al protocolo "c".
b- Este protocolo, secundario, debe permitir la lectura coloquial de la imagen generada por PERCEPCION UML, principalmente utilizado para la enseñanza y primeros pasos con los diagramas de clases.
c- Otro protocolo principal de fácil sintaxis y simple a la hora de recordar para generar imágenes o compartirlo entre usuarios no videntes, o de un usuario no vidente con su docente o par a la hora de trabajar.

Protocolo a: Este protocolo quizás sea el más fuertemente utilizado por la aplicación, ya que transformara todos los datos guardados en la base de datos a una imagen, un audio y dos textos. Este protocolo es el proporcionado por plantuml.com, una herramienta de código abierto que permite diseñar diagramas de clases y que permite integrar nuevas aplicaciones a su página web. Dicho protocolo es bastante simple y entendible, pero no tiene la naturalidad que se está buscando, pero será por demás útil a la hora de la generación de las imágenes.
Entonces pasando en limpio, la información almacenada en la base de datos se transformará en imagen gracias a plantuml.com, luego resta transformarla a un texto coloquial y a un nuevo protocolo.

Protocolo b: En este apartado solo hablare de la transformación a texto coloquial. Este texto es principalmente útil para describir el diagrama de clases como si lo estuviéramos explicando, guardando detalles en cada palabra, tiene un tinte literario y enfocado en el uso del español para describirlo. Explicar un diagrama de clases parece una tarea simple y única, pero me he topado con la dificultad de plantearme: ¿Cómo se lee este diagrama de clases? ¿Es lo mismo decir Clase publica Persona, con atributos privados int dni y String Persona, que decir Clase publica denominada Persona, cuyos atributos son, un atributo llamado dni con permisos privado y del tipo int? Claramente estoy diciendo lo mismo en ambos ejemplos, pero ¿cuál es el más acertado?,

claramente no hay una única forma y no hay una lectura correcta única, para esto es que se les dio un diagrama de clases complejo a algunos docentes y alumnos avanzados para diseñar este lenguaje. En los próximos capítulos se hablará un poco de como es este lenguaje coloquial y como genero cambios en el aplicativo.
Una vez generado este lenguaje, claramente será el utilizado para generar el audio que explicaría la imagen a trabajar.

Protocolo c: Es el protocolo quizás más importante de esta aplicación, ya que no sólo es para PERCEPCION UML sino también para toda la vida académica, también es basado en plantuml.com solo que mucho más simple y totalmente en español. En los primeros pasos se planteó usar el protocolo de PlantUML, pero luego de entrevistas con el usuario estrella, se muto y se lo transformo en el que se mencionara a continuación.

Cabe destacar, que como este es un protocolo que tiene como finalidad última transformarse en algún lenguaje de programación, para resolver algún problema, no se utilizan ni tildes, ni diéresis, y simbología no aceptada por los lenguajes de programación ordinarios.

El protocolo tiene el siguiente esquema:

a- Titulo (optativo): En caso de quererle dar un título al diagrama de clases se debe escribir "Titulo este es el titulo del diagrama de clases"
b- Clases: Se define el tipo de la clase por medio de los comandos "Clase, ClaseAbstracta o Interface", luego un espacio y el nombre de la clase, a continuación se da inicio a la clase con una llave abierta,InicioClase, ejemplo:
   ClaseAbstracta EmpleadoInicioClase
c- Atributos(opcionales): Una vez definida la clase, con su nombre y tipo, y abierta con la llave, se pasa a describir los atributos, uno por renglón, utilizando la notación de UML para la privacidad, es decir +,-,# o vacio, dejando un espacio el tipo del atributo, por ejemplo int, String, char, etc, y dejando espacio el nombre del atributo. Por ejemplo, quedaría algo así:
   -String nombre
   -String direccion
   #String teléfono
d- Métodos(opcionales): Nuevamente se comienza por la privacidad, usando la sintaxis de UML, luego el nombre del método, se abre un paréntesis y dentro de dicho paréntesis,

separado por comas los argumentos con su tipo y nombre; una vez cerrado el paréntesis se pone dos puntos y el tipo del retorno, por ejemplo:

+pedirAumento (Calendar fecha , ListaDePrecio precios) :float

+hacerCheckIn ( ) :Void

e- Cierre de la clase: una vez terminado de poner los atributos y métodos, o no poner ninguno de ellos, se cierra la clase con la llave, FinClase. Ejemplo:

Clase PersonaInicioClase

…(atributos y métodos)

FinClase

f- Relaciones(opcionales): Las relaciones se definen luego de tener todas las clases creadas, y la relación con su sintaxis es simple, la mejor manera de comprenderla es con los siguientes ejemplos, para conocer todas las relaciones posibles y los comandos se aconseja leer el manual del usuario.

Ejemplo de relaciones:

AeroPuerto **UnoAMuchos ComposicionDesde UnoAMuchos** Avion

AeroPuerto **UnoAMuchos AgregacionDesde UnoAMuchos** Persona

Avion **UnoAMuchos ComposicionDesde UnoAMuchos** Componente

Componente **UnoAMuchos Implementa CeroAUno** Precio

Pasajero **Implementa** PantallaPasajero

Persona **HeredaDe** Empleado

Persona **HeredaDe** Pasajero

FuncionesPiloto **Implementa** Empleado

Lo que aparece en negrita en los ejemplos de arriba son los comandos asignados a las relaciones, notar lo fácil de la lectura, y que es opcional la cardinalidad.

Para cerrar este apartado se deja un ejemplo completo del protocolo b, y a continuación la figura 1 generada por la aplicación PERCEPCION UML.

Protocolo:

Titulo Prueba para TFL - Creada por: Nico - 18/06/2018
Clase AeroPuerto InicioClase

-int idAeropuerto
-String nombre
-String direccion
#String telefono
+mostrarPrecios (String fecha , Int puntos) :String
FinClase
Clase Avion InicioClase

-int idAvion
+int cantidadDePasajeros
+float pesoMaximo
+repararComponente (Componente componente) :String
FinClase
Clase Componente InicioClase

-int idComponente
-String nombre
+calcularPrecio ( ) :Int
FinClase
Clase Precio InicioClase

-int idPrecio
-float descuento
-int tipo
FinClase
Clase Empleado InicioClase

-int legajo
-int anioDeIngreso

FinClase

Clase Pasajero InicioClase

-int numeroViajero

-float precioPagado

+hacerCheckIn ( ) :Void

+subirAAvion (Int numeroAvion) :Void

FinClase

Clase Persona InicioClase

-int dni

-String nombre

-String apellido

+String celular

+pedirAumento (Calendar fecha , ListaDePrecio precios) :float

FinClase

Interface PantallaPasajero InicioClase

-String fechaActualizacion

+mostrarArrivos ( ) :String

+mostrarTiempoDeDespegue ( ) :String

FinClase

ClaseAbstracta FuncionesPiloto InicioClase

+mostrarRuta ( ) :String

+mostrarTripulacion ( ) :String

FinClase

AeroPuerto UnoAMuchos ComposicionDesde UnoAMuchos Avion

AeroPuerto UnoAMuchos AgregacionDesde UnoAMuchos Persona

Avion UnoAMuchos ComposicionHacia UnoAMuchos Componente

Componente UnoAMuchos Implementa CeroAUno Precio

Pasajero Implementa PantallaPasajero

Persona HeredaDe Empleado

Persona HeredaDe Pasajero

FuncionesPiloto Implementa Empleado

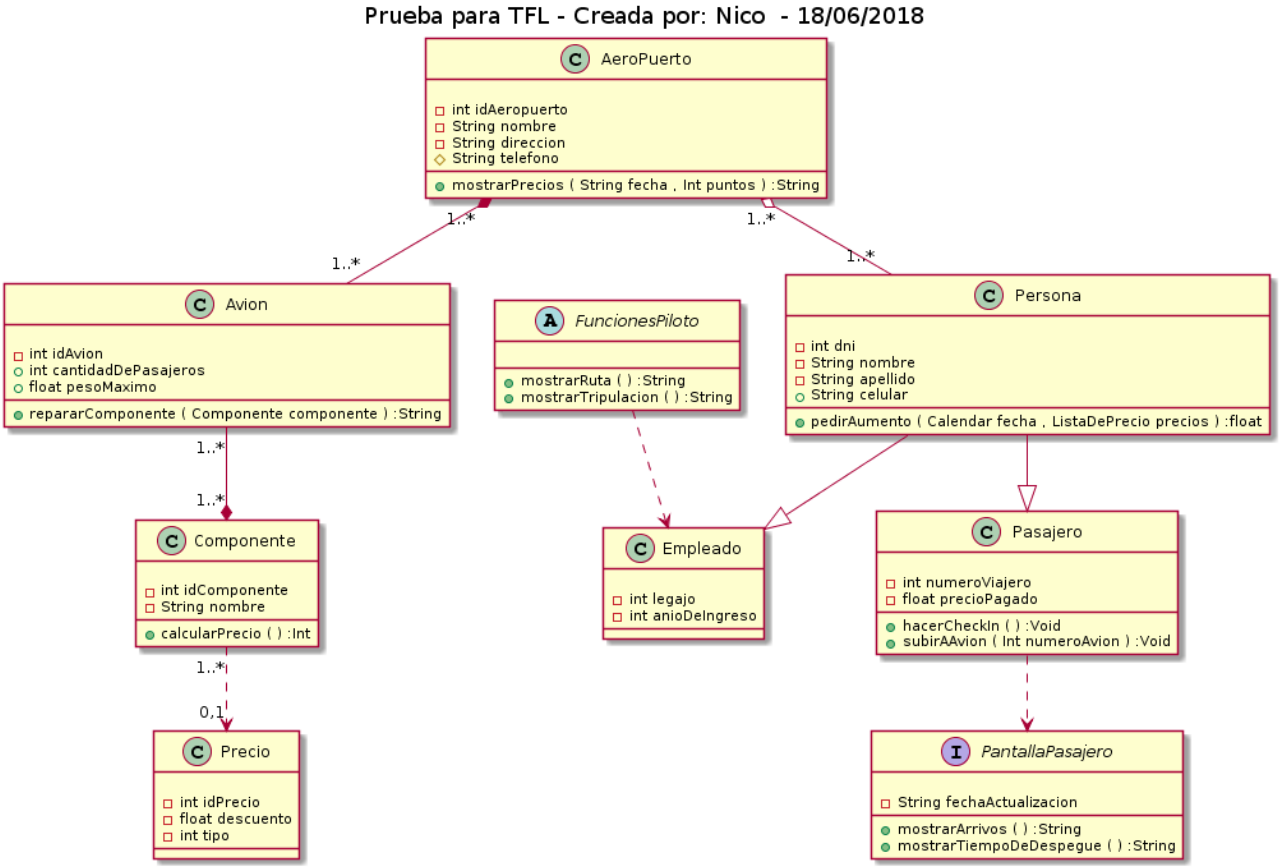


Figura 1: Diagrama de clases generado por PERCEPCION UML al recibir el protocolo expresado en este documento.

3. SOLUCIÓN

En este capítulo se presenta este nuevo proyecto encarado de acuerdo con el Estándar IEEE 830-1998, por medio del paradigma de orientación a objetos e implementado lenguaje Java, utilizando un patrón MVC, con la parte visual desarrollada en HTML5, JavaScript y JSP, los datos son guardados en una base de datos con distribución MYSQL hospedada en el servidor Jelastic. Estos procesos y etapas serán expuestas en este segmento.

3.0 METODOLOGÍA DE LA INVESTIGACION Y ELABORACION DE LA SOLUCIÓN

El estudio que hemos realizado para llevar a cabo este Trabajo Final de Licenciatura se puede definir como de investigación y desarrollo.

Es decir, como define Bisquerra (1989): "(…) este estudio se llevó a cabo para mejorar una situación dada en un contexto, en el que se realizan actividades sistemáticas de investigación de campo, que sirven para el desarrollo de nuevo conocimiento aplicado a nuevas tecnologías."

Específicamente este TFL, siguió esta metodología para diseñar un nuevo producto tecnológico para satisfacer una necesidad no cubierta en los estudios universitarios, en particular en la carrera de sistemas.

Según Bisquerra (1989) nos dice que "Se entiende por investigación y desarrollo un tipo de investigación orientado a la innovación. Sigue un proceso que suele presentar dos etapas: 1) investigar hasta crear un nuevo producto; y 2) después mejorarlo ". (Bisquerra: 1989, pág. 288).

La metodología de este proyecto se basó en los siguientes pasos:

- Investigación y revisión
- Planeación: Definir las habilidades de los futuros usuarios
- Desarrollo del producto: incluye la preparación instruccional de materiales, técnicas, tecnologías y procedimientos a utilizar.
- Pruebas operaciones.
- Pruebas de campo
- Implementación

3.0.1. Investigación y revisión

Esta etapa se basó, principalmente, en la revisión de literatura ya existente, prueba y uso de tecnologías y asesoramiento.

Se consultaron innumerables páginas web dedicadas a la interacción con personas no videntes, se profundizo sobre el concepto de accesibilidad y todo el marco teórico que rodea a este concepto.

Se pidió asesoramiento, a quien sería nuestro primer usuario, Cristian, para no cometer errores que salieron a la luz muy rápido al hablar con él, como, por ejemplo, la importancia de la utilización de productos en formato txt, la necesidad de teclas abreviadas, pensar el SW para que no tenga que utilizarle el mouse ni un solo instante y para fundamentalmente ignorar la estética web, y poner muy por encima el acceso a la información.

3.0.2. Planeamiento y desarrollo

En este ítem, se destaca el cómo de la construcción de la aplicación. Si bien se menciona en este documento paso a paso, se utilizó un modelo de ciclo de vida prototipado evolutivo para la generación del software, su, validación, verificación, integración y todas las etapas de su desarrollo, cada una de estas etapas formara parte del próximo capítulo.

De manera que los siguientes 4 ítems:

- Desarrollo del producto: incluye la preparación instruccional de materiales, técnicas, tecnologías y procedimientos a utilizar.
- Pruebas operaciones.
- Pruebas de campo
- Implementación

Estarán detallados en la solución de esta problemática por medio del ciclo de vida elegido.

3.1 PROCESO DE SELECCIÓN DE UN MCVS

En este segmento hare una vista de las actividades que ocurrieron durante el desarrollo de Percepción UML, donde intento determinar el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas, es decir el momento en el cual elijo el modelo de ciclo de vida más adecuado para el diseño de esta aplicación.

A modo introductorio, y de forma muy breve, en los siguientes puntos daré una minúscula explicación de cómo identificar los posibles MCVS y la elección que he tomado con su debida justificación.

3.1.1. Identificar los posibles MCVS

Dedicare estos primeros renglones a la comprensión y entendimiento de porque es necesario construir un ciclo de vida pertinente desde el puntapié inicial de cualquier proyecto de software, en particular para Percepción UML.

Todo proyecto de ingeniería tiene unos fines ligados a la obtención de un producto, proceso o servicio que es necesario generar a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al conjunto de las fases empleadas se le denomina "ciclo de vida". Percepción UML no es un caso particular.
Sin embargo, la forma de agrupar las actividades, los objetivos de cada fase, los tipos de productos intermedios que se generan, etc. pueden ser muy diferentes dependiendo del tipo de producto o proceso a generar y de las tecnologías empleadas. Por ello es necesario definir de antemano y casi a primera instancia el modelo de ciclo de vida que más se ajustará a nuestro proyecto. El alcance, las características y la estructura son los aspectos fundamentales a tener en cuenta para la selección del modelo de ciclo de vida.

Las principales diferencias entre distintos modelos de ciclo de vida están en:

-El alcance del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto, o su desarrollo completo o, llevando la cosa al extremo, toda la historia del producto con su desarrollo, fabricación, y modificaciones posteriores hasta su retirada del mercado.

-Las características (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propio tema al que se refiere el proyecto (no son lo mismo las tareas que deben realizarse para proyectar un avión que un puente), o de la organización (interés de reflejar en la división en fases aspectos de la división interna o externa del trabajo).

-La estructura de la sucesión de las fases que puede ser lineal, con prototipado, o en espiral.

Veámoslo con más detalle:

Hay una infinidad de modelos de ciclo de vida para el desarrollo de proyectos, a continuación, describo a modo de preámbulo alguno de los más conocidos y utilizados.

Ciclo de vida en cascada

Es el más utilizado, siempre que es posible, precisamente por ser el más sencillo. Consiste en descomponer la actividad global del proyecto en fases que se suceden de manera lineal, es decir, cada una se realiza una sola vez, cada una se realiza tras la anterior y antes que la siguiente. Con un ciclo lineal es fácil dividir las tareas entre equipos sucesivos, y prever los tiempos (sumando los de cada fase).

Requiere que la actividad del proyecto pueda descomponerse de manera que una fase no necesite resultados de las siguientes (realimentación), aunque pueden admitirse ciertos supuestos de realimentación correctiva. Desde el punto de vista de la gestión (para decisiones de planificación), requiere también que se sepa bien de antemano lo que va a ocurrir en cada fase antes de empezarla.

Dicho modelo de ciclo de vida requiere que conozcamos desde el inicio los requisitos, y que ellos estén estáticos es decir no cambien a lo largo de todo el proyecto, o que cambien muy poco. No es nuestro caso ya que inicialmente solo teníamos una idea y algunas funcionalidades obligatorias, pero gran parte de los requisitos eran desconocidas.

Ciclo de vida con prototipado

A menudo ocurre en desarrollos de productos con innovaciones importantes, o cuando se prevé la utilización de tecnologías nuevas o poco probadas, que las incertidumbres sobre los resultados realmente alcanzables, o las ignorancias sobre el comportamiento de las tecnologías, impiden iniciar un proyecto lineal con especificaciones cerradas.

Si no se conoce exactamente cómo desarrollar un determinado producto o cuáles son las especificaciones de forma precisa, suele recurrirse a definir especificaciones iniciales para hacer un prototipo, o sea, un producto parcial (no hace falta que contenga funciones que se consideren triviales o suficientemente probadas) y provisional (no se va a fabricar realmente para clientes, por

lo que tiene menos restricciones de costo y/o prestaciones). Este tipo de procedimiento es muy utilizado en desarrollo avanzado.

El análisis anterior basta para fundamentar y comprender por qué parecería ser el modelo de ciclo de vida más adecuado para Percepción UML.

La experiencia del desarrollo del prototipo y su evaluación deben permitir la definición de las especificaciones más completas y seguras para el producto definitivo.

A diferencia del modelo lineal, puede decirse que el ciclo de vida con prototipado repite las fases de definición, diseño y construcción dos veces: para el prototipo y para el producto real.

Ciclo de vida en espiral

El ciclo de vida en espiral puede considerarse como una generalización del anterior para los casos en que no basta con una sola evaluación de un prototipo para asegurar la desaparición de incertidumbres y/o ignorancias. El propio producto a lo largo de su desarrollo puede así considerarse como una sucesión de prototipos que progresan hasta llegar a alcanzar el estado deseado. En cada ciclo (espirales) las especificaciones del producto se van resolviendo paulatinamente.

A menudo la fuente de incertidumbres es el propio cliente, que, aunque sepa en términos generales lo que quiere, no es capaz de definirlo en todos sus aspectos sin ver como unos influyen en otros. En estos casos la evaluación de los resultados por el cliente no puede esperar a la entrega final y puede ser necesaria repetidas veces.

El esquema del ciclo de vida para estos casos puede representarse por un bucle en espiral, donde los cuadrantes son, habitualmente, fases de especificación, diseño, realización y evaluación (o conceptos y términos análogos).

3.1.2. Seleccionar un modelo para el proyecto

A modo de canalización de todo lo descripto anteriormente y debido a que no se conoce exactamente cómo desarrollar inicialmente el producto ni cuáles son las especificaciones de forma

precisa, suele recurrirse a definir especificaciones iniciales para hacer un prototipo, o sea, un producto provisional, o varios. Entonces se elige seguir el modelo de ciclo de vida prototipado.

Donde desarrolle un prototipo desechable, en el cual probé las tecnologías, verificare la viabilidad de las funcionalidades y veré que tan tangibles pueden ser las primeras soluciones a los intríngulis a solucionar con el software, dicho prototipo desechable será únicamente privado es decir solo yo lo veré y usare, luego de estar familiarizado con las características será mostrado a los clientes externos y asesores para recibir sus críticas.

Una vez mostrada la "enseñanza" de la maqueta desechable, realice dos prototipos evolutivos; el primero tuvo un gran porcentaje de las características finales funcionales, y fue el primer prototipo mostrado y probado, con el mismo se tratará de definir detalles estéticos, operacionales y nuevas funcionalidades que deberán estar presentes en una segunda y última evolución del prototipo.
Puede comprenderse las fases de este ciclo en la figura 2.

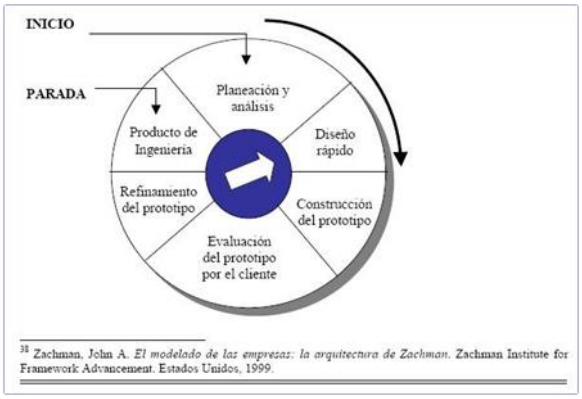


Figura 2: Ciclo de vida prototipado evolutivo

3.2 PROCESO DE INICIACIÓN, PLANIFICACIÓN Y ESTIMACIÓN DEL PROYECTO

La gestión del proyecto presupone establecer condiciones para el desarrollo de este proyecto. Involucra actividades de planificación, estimación de recursos, seguimiento y control, y evaluación del proyecto.

3.2.1 Estimaciones por medio de COCOMOII

Escenario más pesimista con el riesgo de Monte Carlo activo, subestimación mi experiencia con el lenguaje a emplear (Java) y peor disponibilidad de tiempos y recursos. La figura 3 muestra los resultados.

Results

Software Development (Elaboration and Construction)

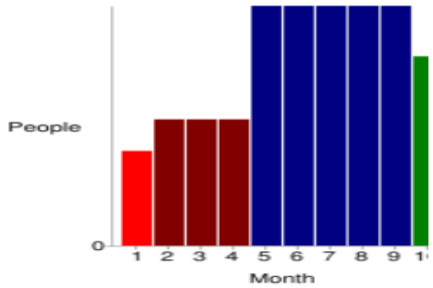
Schedule = 8.3 Months
Cost = $106

Total Equivalent Size = 2650 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.3	1.0	0.3	$6
Elaboration	1.3	3.1	0.4	$26
Construction	4.1	5.2	0.8	$81
Transition	0.6	1.0	0.6	$13

Staffing Profile



Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.0	0.2	0.4	0.1
Environment/CM	0.0	0.1	0.2	0.0
Requirements	0.1	0.2	0.3	0.0
Design	0.1	0.5	0.6	0.0
Implementation	0.0	0.2	1.4	0.1
Assessment	0.0	0.1	1.0	0.2
Deployment	0.0	0.0	0.1	0.2

Software Size Sizing Method Function Points
Unadjusted Function Points 50 Language Java

Software Scale Drivers
Precedentedness Nominal
Development Flexibility Nominal
Architecture / Risk Resolution Nominal
Team Cohesion Extra High
Process Maturity Nominal

Software Cost Drivers
Product
Required Software Reliability High
Data Base Size Low
Product Complexity Nominal
Developed for Reusability Nominal
Documentation Match to Lifecycle Needs Nominal

Personnel
Analyst Capability Nominal
Programmer Capability Nominal
Personnel Continuity Very High
Application Experience High
Platform Experience High
Language and Toolset Experience Nominal

Platform
Time Constraint Nominal
Storage Constraint Nominal
Platform Volatility Nominal

Project
Use of Software Tools Nominal
Multisite Development Nominal
Required Development Schedule High

Maintenance Off

Software Labor Rates
Cost per Person-Month (Dollars) 20

Figura 3: Escenario pesimista obtenido con COCOMOII

En esta estimación, más pesimista del proyecto, puede resumirse que se necesitarían alrededor de 11 meses trabajando de forma individual en condiciones de subestimación de las capacidades de trabajo y de aptitudes. Por lo que podría decirse que será una cota máxima de tiempo de trabajo. Además, se aprecia un costo de 106 dólares, suponiendo que la aplicación es hospedada en un servidor pago (Jelastic, un servidor que cumple con todos los requisitos) desde el día uno.

Dicho costo es ampliamente afrontable y superior al que será en realidad puesto que la aplicación solo será hospedada en la etapa final del proyecto.

Así que por medio de CocomoII obtenemos un tiempo estimado levemente inferior al año y un costo monetario de como mucho 106 dólares. De manera que se puede decir, por medio de esta herramienta que será una producción viable.

En conclusión, siendo que el proyecto inicio en septiembre 2017, esta estimación nos da un resultado que definía que el proyecto se terminaría en septiembre 2018 o antes.

3.2.2 Establecer la matriz de actividades para el MCVS

El diseño constará de un prototipo desechable y 2 prototipos evolutivos, puede comprenderse esto en la tabla 1, mapa de actividades.

		sep-17			hasta abril 2018				hasta sept. 2018		
	Prel.	Prototipo des.			Prototipo Evolutivo x 2				Inst. y Operación		
	AP	ERI	DEI	VIU	ERS	DAS	C	PP	IF	OM	R
Proceso de Selección de un MCVS											
- Identificar los posibles MCVS	X										
- Seleccionar un modelo para el proyecto.	X										
Proceso de Iniciación, Planificación y											
Estimación del Proyecto											
- Establecer la matriz de actividades para el MCVS	X										
- Asignar los recursos del proyecto.	X	X	X	X	X	X	X	X			
- Definir el entorno del proyecto.	X										
- Planificar la gestión del proyecto.	X										
Proceso de Seguimiento y Control del Proyecto											
- Analizar riesgos.	X	X	X	X	X	X	X				
- Realizar la planificación de contingencias.		X	X	X	X	X	X	X			
- Gestionar el proyecto.		X	X	X	X	X	X	X			
- Implementar el sistema de informes de problemas.		X	X	X	X	X	X	X			
- Archivar registros.		X	X	X	X	X	X	X			
Proceso de Gestión de Calidad del Software											
- Planificar la garantía de calidad del software.		X	X		X	X	X				
- Desarrollar métricas de calidad.		X	X		X	X	X				

- Gestionar la calidad del software.		X	X	X	X	X	X	X			
- Identificar necesidades de mejora de la calidad.		X	X	X	X	X	X	X			
Proceso de Exploración de Conceptos											
- Identificar las ideas o necesidades.	X	X			X						
- Formular las soluciones potenciales.	X	X			X						
- Dirigir los estudios de viabilidad.	X	X			X						
- Planificar la transición del sistema (si se aplica).	X	X			X						
- Refinar y Finalizar la idea o necesidad.	X	X			X						
Proceso de Asignación del Sistema											
- Analizar las funciones del sistema.			X			X					
- Desarrollar la arquitectura del sistema.						X					
- Descomponer los requisitos del sistema.						X					
Proceso de Análisis de Requisitos											
- Definir y Desarrollar los requisitos del software.					X						
- Definir los requisitos de interfaz.		x									
- Priorizar e Integrar los requisitos del software.					X						
Proceso de Diseño											
- Realizar el diseño preliminar.			X			X					
- Analizar el flujo de información.						X					
- Diseñar la base de datos (si se aplica).						X					
- Diseñar las interfaces.						X					
- Seleccionar o Desarrollar algoritmos (si se aplica).						X					
- Realizar el diseño detallado.						X					
Proceso de Implementación e Integración											
- Crear los datos de prueba.						X	X				
- Crear el código fuente.							X				
- Generar el código objeto.							X				
- Crear la documentación de operación.			X			X	X				

- Planificar la integración.						X	X				
- Realizar la integración.							X				
Proceso de Instalación y Aceptación											
- Planificar la instalación.									X		
- Distribuir el software.									X		
- Instalar el software.									X		
- Cargar la base de datos (si se aplica).									X		
- Aceptar el software en el entorno de operación.								X	X		
- Realizar las actualizaciones.									X		
Proceso de Operación y Soporte											
- Operar el sistema.										X	
- Proveer de asistencia técnica y consultas.										X	
- Mantener el histórico de peticiones de soporte.										X	
Proceso de Mantenimiento											
- Realizar el mantenimiento correctivo.										X	
- Reaplicar el ciclo de vida del software.										X	
Proceso de Retiro											
- Notificar al usuario.											X
- Conducir operaciones en paralelo (si se aplica).											X
- ~~Retirar el sistema.~~											X

Tabla 1: Mapa de actividades

3.2.2.1 Asignar los recursos del proyecto

En esta sección discriminaré el uso de los recursos para la ejecución del proyecto.

Recursos humanos:

El software que desarrollar consta de un único programador, es decir el creador de este trabajo final de licenciatura, el mismo será el encargado además de la interfaz gráfica y el diseño arquitectónico.

La funcionalidad del software será definida no sólo por el usuario "estrella" Cristian, sino también por el conocimiento de los diagramas de clases y la orientación a Objetos de Hernán Amatriain y los docentes a cargo de orientación a objetos de la UNLa.

Gestores superiores: Hernán Amatriain
Gestores (técnicos) del proyecto: Nicolás Perez
Profesionales: Nicolás Perez
Clientes: Cristian Juárez
Usuarios: Cristian y cualquier interesado en desarrollar un DC.

Recursos tecnológicos:

La programación será íntegramente en Java, por medio de la plataforma eclipse. Los datos serán almacenados en una base de datos en MYSQL. Y la interface web estará diseñada en HTML, con Ajax, JSP y JavaScript embebidos.

La aplicación será una aplicación web, testeada profundamente en Google Chrome, por lo que, recomiendo este navegador, si bien se lo hará compatible con todos los otros conocidos.

3.2.2.2 Definir el entorno del proyecto

El proyecto está orientado a resolver un problema planteado por la UNLa y el Ing. Hernán Amatrian junto al sector de accesibilidad de la UNLa; los cuales se plantean la necesidad de introducir de una manera más grata y planeada a los alumnos con dificultades visuales a la carrera y al mercado laboral. En esta primera instancia, todo el grupo de trabajo se enfocará a darle esta herramienta que le va a permitir a todos los usuarios crear DC de una forma sencilla y sin prestarle mayor atención al diseño estético, sino solo a la estructura interna; y de ser necesario poder escuchar y/o ver el respectivo diagrama.

3.2.2.3 Planificar la gestión del proyecto

El proyecto fue planificado en primera instancia para que sea terminado el 1-8-2018, la distribución de las tareas en cada una de las etapas esta descripta en el diagrama de Gantt, además vemos plasmados las 3 maquetas pensadas para el proyecto.

3.3 PROCESO DE SEGUIMIENTO Y CONTROL DEL PROYECTO

En este apartado evaluaremos el proceso iterativo de seguimiento, registro y gestión de costos, problemas y rendimiento de un proyecto durante su ciclo de vida. En este proceso se realiza un análisis de riesgo que permite identificar los problemas potenciales, determinar su probabilidad de ocurrencia y su impacto y establecer los pasos para su gestión. Los riesgos identificados y su gestión se documentan en el Plan de Contingencia. El progreso se revisa y mide con respecto a los hitos establecidos en el Plan de Gestión del Proyecto Software.

3.3.1 Análisis de riesgos

En una primera instancia, por medio del análisis del entorno se pueden identificar 3 riesgos, potenciales e inevitables, descritos en la tabla 2.

4.3.2 Realizar la planificación de contingencias

Se realizó un plan de respaldo y un plan de emergencia para extirpar los problemas en la generación y subsistencia del proyecto.

El **plan de respaldo**. Para evitar la pérdida de datos o para contemplar cambios abruptos en los ideales del cliente, se respaldan todos los archivos y documentos del proyecto en la plataforma GitHub y One Drive, en:

Soporte en One Drive: https://1drv.ms/f/s!AgB0dw0E7wKakIIU682sdWEyghwxHQ
Proyecto GitHub: https://github.com/gruposeminario/PercepcionUML-UNLa
Web: http://node24730-uml-dc.jelastic.saveincloud.net/UML-DC/

Descripción	***Impacto***	***Probabilidad***	***Acción a seguir***
Llenado de campos por medio de choice o texto	Si bien Cristian no tiene problemas para poder tipear e interactuar con una pantalla, y mucho menos los usuarios sin limitaciones visuales; puede que gran parte de los usuarios con menos destrezas visuales y de tecnologías no puedan llenar estos campos por medio de un teclado.	Media	Se realiza todo con las librerías pertinentes para que las herramientas de accesibilidad visual de los navegadores puedan describir bien la pantalla. Además, se incorporan indicativos extras a los menúes.
Dependencia en otras tecnologías	Tanto el graficado como la lectura acústica queda en manos de PlantText y responsivevoice, respectivamente, librerías totalmente gratuitas, pero que no son controladas por Percepción UML.	Media	Se creará también una versión de escritorio que no dependa de estas librerías online.
Traducción a Braille	El Braille no es otra cosa más que un idioma, por lo que es complejo una traducción textual entre el Braille y el español sin un experto a cargo. Si bien se puede traducir letra por letra al Braille y se tornaría leíble y entendible no es la mejor opción.	Baja	Se deja como un recurso a futuro, y se lo descarta en esta aplicación.

Tabla 2: Gestión de Riesgos.

Para combatir los riesgos descriptos en la Tabla de arriba, propongo las siguientes soluciones:

Se disminuye al máximo la cantidad de opciones a elegir por medio de clics, y se programa la aplicación para que todas las opciones sean asistidas por medio de audio.

Depender de otras tecnologías no ligadas a mi software puede ser crítico en caso de que dichos servicios webs dejen de existir, por ello se eligen 2 servicios estables cuya probabilidad de desaparición es mínima. Además, plantText es de código abierto por lo que podría ser ejecutado por mi aplicación por medio de librerías de JAVA, en un futuro que deje de ser un servicio gratuito. Mientras que responsivevoice es una herramienta utilizada por Google, cosa que también le da una gran disponibilidad en el tiempo, pero en caso de que deje de estar disponible en el segmento de exploración de conocimiento propongo otras alternativas semejantes.

El **plan de emergencia**. Suponiendo que la transformación del diagrama de clases a Braille se torna muy compleja y fuera de mi alcance, o si se logra, pero con un nivel de traducción poco literal; se optará por dejarlo de lado y para que sea tratado en otro proyecto que se enfoque únicamente en eso, para luego poderlo incorporar a esta aplicación.

4.3.3 Gestionar el proyecto

Para una mejor gestión y administración, principalmente de los tiempos, se empleó un diagrama PERT, figura 4, desde el momento de la creación de un primer prototipo evolutivo, en el mismo se aprecian las fechas teóricas pactadas y las fechas reales dadas en el proyecto, además que se distingue el camino critico a transitar para evitar problemas groseros de calendarios

Dicho diagrama, hecho con Microsoft Project, se aloja en:

Link: https://1drv.ms/u/s!AgB0dw0E7wKajsQl1G9ZkoGyZwR8MQ

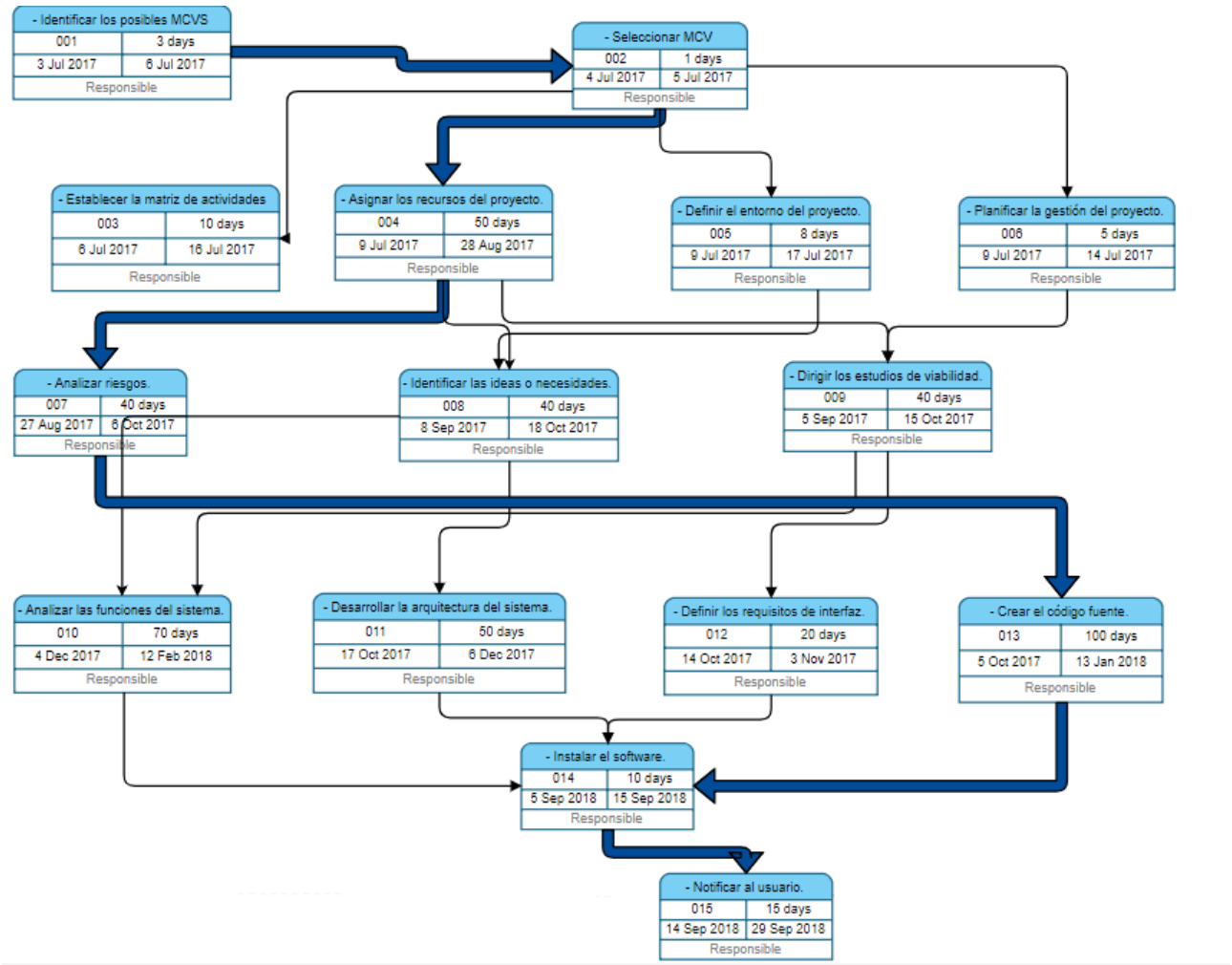


Figura 4: PERT con camino crítico, con actividades principales.

Implementar el sistema de informes de problemas

El primer problema crítico con el que me tope fue el poco conocimiento del cliente real al que estará destinado el producto. El mismo fue solucionado parcialmente por el contacto vía mail con el mismo. Diálogos vía WhatsApp y una posterior primera entrevista allá por mediados de abril, de allí en más el dialogo se tornó más fluido y ayudó mucho mostrar la primera maqueta para mejorar el feed back.

Para impedir que cualquier persona pueda editar o borrar un diagrama de clases que no debe, se crea una autentificación vía email o por un simple usuario y pass generado por el cliente, para solucionar dicho problema sin utilizar la lógica compleja del Login que puede resultar engorrosa a la hora de una aplicación que queremos que sea destinada a cualquier persona en cualquier momento, sumado

a la dificultad extra e innecesaria para las personas no videntes. Así que se optó por solucionar este problema con un simple código enviado vía email en caso de no elegir una contraseña para él ingreso, cualquier que tenga ese código de validación podría modificar el diagrama de clases o las versiones disponibles sin la necesidad de un Login.

4.3.4 Archivar registros

Todos los registros de avances, o los que describen en profundidad lo ilustrado por este documento puede encontrárselo en el segmento Documentos de Registros de nuestro directorio online en:

Link Registros: https://1drv.ms/f/s!AgB0dw0E7wKakIIZj8Gy4U0pu7_AXA

3.3 PROCESO DE GESTIÓN DE CALIDAD DEL SOFTWARE

En este segmento nos centraremos en la calidad del producto, servicio o la satisfacción del cliente, sino en los medios para obtenerla. Por lo tanto, la gestión de calidad utiliza al aseguramiento de la calidad y el control de los procesos para obtener una calidad más consistente.

3.3.1 Planificar la garantía de calidad del software

Los modelos de calidad del software vienen a ayudar en la puesta en práctica del concepto general de calidad que vimos en el apartado anterior, ofreciendo una definición más operacional. Unos de los modelos de calidad más antiguos y extendidos es el de McCall [McCall, 1977], y de él han derivado otros modelos, como el de Boehm [Boehm, 78] o el SQM [Murine, 1988]. En los modelos de calidad, la calidad se define de forma jerárquica. Es un concepto que se deriva de un conjunto de subconceptos, cada uno los cuales se van a evaluar a través de un conjunto de indicadores o métricas. Para el estudio de calidad de nuestro Software nos basaremos en el diagrama descripto en la tabla 3.

Tabla 3: Estructura de 4 niveles de la calidad del SW

En el nivel más alto de la jerarquía se encuentran los FACTORES de calidad, que representan la calidad desde el punto de vista del usuario. Son las características que componen la calidad. También se les llama Atributos de Calidad Externos. Cada uno de los factores se descompone en un conjunto de CRITERIOS de calidad. Son atributos que, cuando están presentes, contribuyen al aspecto de la calidad que el factor asociado representa. Se trata de una visión de la calidad desde el punto de vista del producto software. También se les llama Atributos de Calidad Internos. Para cada uno de los criterios de calidad se definen entonces un conjunto de MÉTRICAS, que son medidas cuantitativas de ciertas características del producto que, cuando están presentes, dan una indicación del grado en que dicho producto posee un determinado atributo de calidad.

Nota: Una desventaja es que aún no ha sido demostrada la validez absoluta de ninguno de estos modelos. Las conexiones que establecen entre características, atributos y métricas se derivan de la experiencia, y de ahí que existan múltiples modelos.

3.3.2 Desarrollar métricas de calidad

Para desarrollar las métricas pertinentes a mi proyecto opto por utilizar el modelo de McCall. El modelo de McCall se basa en 11 factores de calidad, que se organizan en torno a los tres ejes de la siguiente forma (Tabla 4):

PUNTO DE VISTA	FACTORES
Operación del producto	- Facilidad de uso - Integridad - Corrección - Fiabilidad - Eficiencia
Revisión del producto	- Facilidad de mantenimiento - Facilidad de prueba - Flexibilidad
Transición del producto	- Facilidad de reutilización - Interoperabilidad - Portabilidad

Tabla 4: Estructura de las métricas de McCall

Los 4 factores de McCall tomados se definen como sigue:

1**. Fiabilidad:** Hasta qué punto se puede confiar en el funcionamiento sin errores del programa. Por ejemplo, si el programa anterior suma dos números, pero en un 25% de los casos el resultado que da no es correcto, es poco fiable.

2. **Facilidad de mantenimiento:** El coste de localizar y corregir defectos en un programa que aparecen durante su funcionamiento.

3. **Flexibilidad:** El coste de modificación del producto cuando cambian sus especificaciones.

4. **Portabilidad (o Transportabilidad):** El coste de transportar o migrar un producto de una configuración hardware o entorno operativo a otro.

Las métricas que desarrollaré en este proyecto son las siguientes:

- **Fiabilidad** = 1 - (número de errores/ número de líneas de código)
- **Facilidad de mantenimiento** = 1 - 0.1 (número medio de días-hombre por corrección)
- **Portabilidad** = 1 - (esfuerzo para portar / esfuerzo para implementar)
- **Flexibilidad** = 1 - 0.05 (número medio de días-hombre por cambio)

Trabajando con las dos primeras iteraciones de nuestro maquetado, obtuve los siguientes índices, expresados en la tabla 5.

Datos:	**Métricas:**
Número de errores= 21 Número de líneas de código = 2092 Número medio de días hombre por corrección = 3 Esfuerzo = 2 Esfuerzo para implementar = 10 Número medio de días-hombre por cambio = 15	Fiabilidad = 0,9899 Facilidad de mantenimiento = 0,7 Portabilidad = 0,8 Flexibilidad = 0,75

Tabla 5: Obtención de métricas. Maquetado.

Reitero que lo anterior fue únicamente trabajando con las maquetas, es decir con un software casi estéticos sin las funcionalidades al 100%, a posteriori se realiza el mismo estudio, pero con la versión final del nivel evolutivo, (Tabla 6):

Datos:	**Métricas:**
Número de errores= 37 Número de líneas de código = 15833 Número medio de días hombre por corrección = 5 Esfuerzo = 4 Esfuerzo para implementar = 14 Número medio de días-hombre por cambio = 3	Fiabilidad = 0,997 Facilidad de mantenimiento = 0,5 Portabilidad = 0,71 Flexibilidad = 0,85

Tabla 6: Obtención de métricas. Maquetado evolutivo.

Luego hice el mismo análisis para el prototipo evolutivo final, donde obtuve los siguientes resultados (Tabla 7):

Datos:	**Métricas:**
Numero de errores= 7 Número de líneas de código = 18033 Número medio de días hombre por corrección = 2 Esfuerzo = 4 Esfuerzo para implementar = 14 Número medio de días-hombre por cambio = 2	Fiabilidad = 0,902 Facilidad de mantenimiento = 0,7 Portabilidad = 0,9 Flexibilidad = 0,8

Tabla 7: Obtención de métricas. Versión final.

Donde puedo destacar que se sigue teniendo una importante fiabilidad más allá del aumento sustancial de las LDC y los últimos errores encontrados, el mantenimiento sigue en un nivel aceptable, con menos de dos días de trabajo por arreglo, y la aplicación sigue siendo portable gracias a su uso online.

3.3.3 Gestionar la calidad de SW

Un modelo de calidad del software es un conjunto de buenas prácticas para el ciclo de vida del software, enfocado en los procesos de gestión y desarrollo de proyectos. Construir un modelo de calidad es bastante complejo y avanzando en el proyecto vamos a ir mejorando los procesos.

3.3.4 Identificar la necesidad de mejoras en la calidad

Con el primer maquetado evolutivo terminado se identificaron 3 necesidades de mejoras de:

- Agregar valores por defecto a los campos, para facilitar el llenado de campos.
- Permitir que se pueda volver atrás en todo momento y borrar incluso campos llenados desde un primer momento, además de permitir la previsualización constante.
- Incorporar al sistema un manual no solo de la aplicación, sino también de los diagramas de clase, para que puedan consultarlo en todo momento, ya sea para quitarse dudas como para asegurarse poder realizar el mejor diagramado.

Luego con el segundo maquetado se agregaron nuevas mejoras como:

- Permitir él envió de txt por mail, no sólo con el texto coloquial, sino también con un link de descarga y otro adjunto con el protocolo que define a dicho diagrama.
- Se pide una solapa de ayuda que se pueda consultar en todo momento.
- Eliminar las pantallas largas para evitar tener que utilizar el mouse y su rueda.

3.5 PROCESO DE EXPLORACIÓN DE CONCEPTOS

Este proceso incluye la identificación de una necesidad, la formulación de soluciones potenciales, su evaluación y refinamiento a nivel de sistema. Una vez establecidos sus límites, se genera el Informe de la Necesidad del sistema a desarrollar. Este informe inicia el Proceso de Asignación del Sistema y, o, el Proceso de Requisitos, y alimenta los Procesos de Gestión del Proyecto.

El Informe de la Necesidad es un documento que constituye la base de todo el trabajo de ingeniería posterior.

3.5.1 Identificar ideas o necesidades

Necesidades que satisfacer:

- La aplicación debe ejecutarse en cualquier lugar o entorno de la manera más fácil y rápida.
- La interfaz debe ser lo más amigable para el usuario posible, no se debe dejar muchas libertades o focos de inconsistencias.
- Los diagramas de clases ya están bien definidos por la UML por lo que es necesario que el software permita llevar al cabo esta estandarización.
- Debe quedar un respaldo en caso de que querer volver atrás con algún cambio.

3.5.2 Formular soluciones potenciales

Ideas del equipo de software:

- Crear una aplicación web que se pueda ejecutar desde cualquier navegador, sin grandes demandas de velocidad y ancho de internet, es decir formularios a llenar simples.
- La aplicación podría ser ejecutable desde los celulares. Para poder modificar y ver los diagramas de tu equipo de trabajo.
- No se adjuntarán archivos pesados ni complejos, se enviarán archivos de texto.
- Se capacitará de forma constante a todos los que quieran usar esta aplicación.
- Se instalará de forma local, además, a quien lo desee la aplicación.

3.5.3 Dirigir los estudios de viabilidad

Los riesgos descriptos en la tabla 2, página 40, y las soluciones planteadas en el apartado anterior se consideran viables para el proyecto, por los siguientes ítems:

Viabilidad operacional: Los lenguajes de programación a implementar, el tiempo disponible, y las condiciones de producción se consideran totalmente viables para los integrantes del equipo, por el conocimiento previo de cada una de las partes y el tiempo disponible por el creador Nicolas Perez, el tutor Hernán Amatriain y la buena predisposición de Cristian.

Viabilidad del mercado: El proyecto tiene una ventaja que muchos otros proyectos desearían, el mismo no será sumergido a un mercado desconocido o inestable, el software será diseñado previo a una necesidad por demás tangible en esta universidad y en el ámbito de sistemas. Hoy por hoy es cuasi imposible el desarrollo de maquetas y la interacción entre compañeros con distintas capacidades visuales por medio de mecanismos ya diseñados puesto que no hay ningún software conocido que traduzca DC a audio sin la asistencia de un humano. Cabe destacar que el mercado será constante, pero no muy numeroso, pero desde el comienzo se supo que el software no sería masivo si no para usuarios muy específicos.

Viabilidad conceptual: Si bien hay riesgos (tabla 2, página 40), y los requisitos fueron, en primera instancia muy volátiles, la comprensión del problema y la factibilidad de la solución han sido comprendidas, aceptada y trabajada sin problemas mayores.

3.5.4 Refinar y Finalizar la idea o necesidad

Uno de los principales problemas sobre el entendimiento del problema a resolver, fue desde un primer momento, el diagramado del DC, puesto que, si queríamos que el usuario no se fije en la estética del DC sino más que nada en su estructura y armado lógico, esto significaba un gran esfuerzo de programación para tener en cuenta la complejidad que puede tener un DC y la posterior implementación en un lenguaje de programación para abarcar todo el abanico de posibilidades.

Este problema fue resuelto en gran parte implementando un software ya existente y compatible con Java, una aplicación de código abierto utilizado por la página PlantText y PlantUML,

La otra cuestión por demás relevante era la transformación de una imagen a sonido, el DC creado con Percepción UML, se transforma a texto con un lenguaje preestablecido, altamente coloquial. Este lenguaje se obtuvo gracias a la ayuda de varios docentes de la Universidad los cuales por

medio de entrevistas online describían con sus palabras, de la mejor manera posible un DC. Con eso se obtuvo un lenguaje estandarizado para el "dictado" de los DC. Este lenguaje fue el introducido a mi aplicación. Una vez transformado el DC a texto, este texto es reproducido gracias a la aplicación embebida "responsivevoice".

A continuación (Figura 6, página 54), puede observarse el primer lenguaje de traducción de DC, previo a la interacción con docentes y luego la última versión de traducción posterior a la ayuda de los docentes y expertos en la creación de diagramas de clases.

La entrevista a docentes puede encontrarse en:

Link PDF con encuesta: https://1drv.ms/b/s!AgB0dw0E7wKakIIfmoGnYiS2BCL1Ig

En base a estas tres respuestas y otras 4 más que me parecían bien ejemplificadas por estas tres, se puede sacar la conclusión que el orden en el que se describen las clases no es primordial, pero si lo es el orden de su estructura interna, se ve que definen primero sus atributos, luego sus métodos y por último sus relaciones. También ha salido a la luz que en caso de que no tenga atributos, métodos o relaciones es bueno remarcarlo para que no parezca que es un dictado incompleto.
Es fundamental realizar pausas entre las clases para definir rupturas en la lectura que describan mejor la lógica.

Una vez analizado esto, dado el DC de la figura 5, se determinó el siguiente lenguaje para describirlo: (notar luego la diferencia con el lenguaje planteado en los inicios, previo a la interacción con docentes y alumnos, figura 6)

…"Diagrama de clases, creado por Nicolas y Hernan el día 22/04/2018 ésta compuesto por:

Clase pública denominada Empleado, donde sus atributos son:
Un atributo privado del tipo int denominado legajo.
Un atributo publico del tipo String denominado nombre.
Un Método privado con nombre calcularSueldo. Con un argumento del tipo Int con nombre anioDeIngreso. , este método retorna un Int.

Clase pública denominada Contratado, sin Atributos definidos.

Interface denominada Tercerizado, donde sus atributos son:

Un atributo privado del tipo int denominado idEmpleado.

Un Método publico con nombre terminarContrato., este método retorna un Void.

Enum denominado Jefe, donde sus atributos son:

Un atributo privado del tipo int denominado idJefe.

Un Método protegido con nombre calcularSueldoNeto., este método retorna un Int.

Clase abstracta denominada Limpieza, sin Atributos definidos.

El Diagrama de clase posee las siguientes relaciones entre las clases:

La clase Empleado con una relación de composición con la clase Tercerizado. Además esta relación va desde cero a uno hasta 1 a muchos.

La clase Contratado con una relación de dependencia con la clase Contratado. Además esta relación va desde cero a muchos hasta cero a muchos.

La clase Contratado con una relación de herencia con la clase Jefe.

La clase Tercerizado con una relación de composición con la clase Jefe. Además esta relación va desde cero a uno hasta cero a muchos.

La clase Jefe con una relación de dependencia con la clase Tercerizado. Además esta relación va desde cero a uno hasta 1 a muchos.

La clase Jefe con una relación de composición con la clase Empleado. Además esta relación va desde 1 a muchos hasta 1 a muchos.

La clase Limpieza con una relación de implementación con la clase Jefe. Además esta relación va desde cero a uno hasta cero a uno.“…

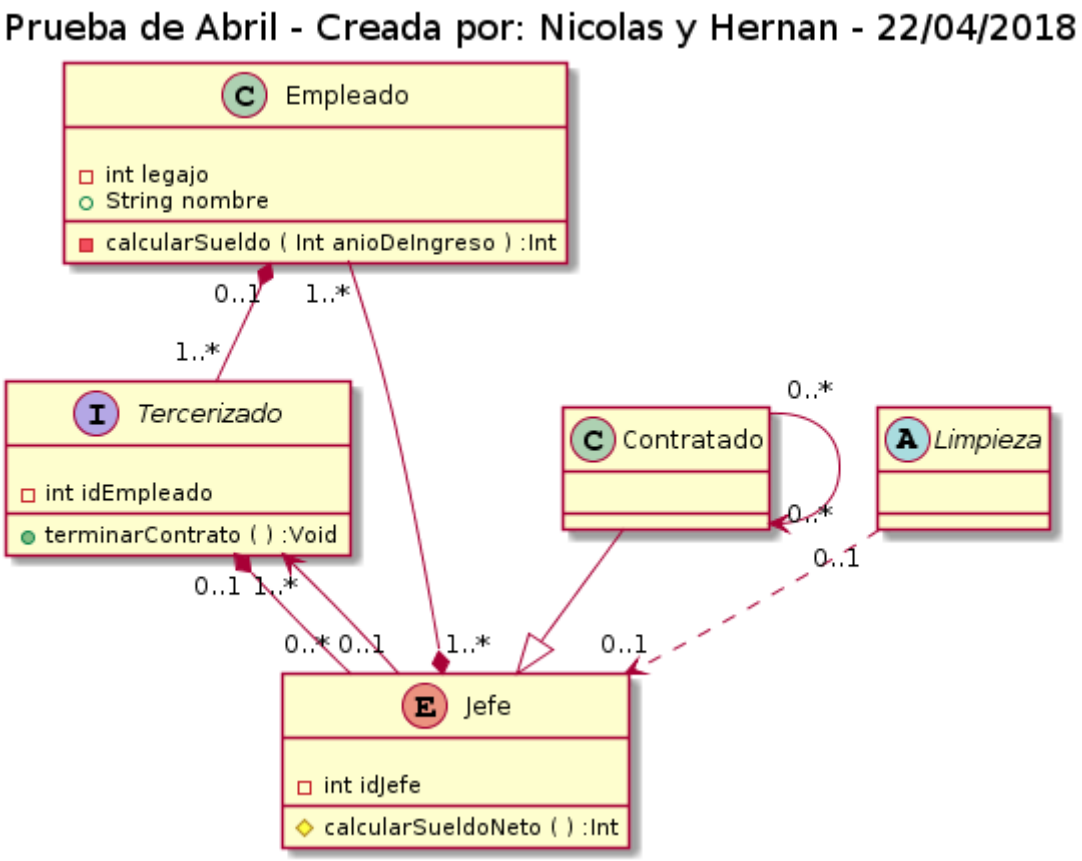


Figura 5: DC utilizado para la descripción y determinación de los protocolos.

Este es uno de los primeros diagramas de los que hemos visto, donde se ven prácticamente todas las relaciones y potencial opciones disponibles, si bien las relaciones en la figura 5, no son todas totalmente correctas a nivel lógico, son perfectas para visualizar como se vería un DC con una gran variedad de relaciones, tipos de datos, clases y estructuras de datos.

Mientras que el protocolo final para describir el diagrama quedaría así:

…"Titulo Prueba de Abril - Creada por: Nicolas y Hernan - 22/04/2018
Clase Empleado InicioClase

-int legajo
+String nombre

-calcularSueldo (Int anioDeIngreso) :Int
FinClase
Clase Contratado InicioClase

FinClase

Interface Tercerizado InicioClase

-int idEmpleado

+terminarContrato ( ) :Void
FinClase
enum Jefe InicioClase

-int idJefe

#calcularSueldoNeto ( ) :Int
FinClase
ClaseAbstracta Limpieza InicioClase

FinClase

Empleado CeroAUno ComposicionDesde UnoAMuchos Tercerizado

Contratado CeroAMuchos Implementa CeroAMuchos Contratado
Contratado HeredaDe Jefe

Tercerizado CeroAUno ComposicionDesde CeroAMuchos Jefe

Jefe CeroAUno Implementa UnoAMuchos Tercerizado
Jefe UnoAMuchos ComposicionDesde UnoAMuchos Empleado

Limpieza CeroAUno Implementa CeroAUno Jefe"...

Clase publica cuyo nombre es Estudiante y sus atributos son: -Un atributo privado del tipo String con nombre nombre -Un atributo privado del tipo int con nombre edad Ademas esta compuesta por los metodos: -Un metodo publico con un retorno del tipo boolean cuyo nombre es esMayorDeEdad ,el mismo necesita de los argumentos: -un int con nombre edad Y tiene las siguientes relaciones: -Esta relacionada con la clase Materia por medio de una Agregacion.

A Su diagrama de clases imagen:

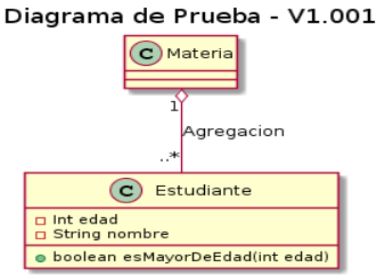


Figura 6. Prototipo desechable sin interacción con expertos y docentes.

3.5.5 Encuestas al usuario final

Primera encuesta donde queda inmortalizada la necesidad de la transformación a audio de una imagen de un DC. Se descarta por completo el hecho de que la imagen sea cualquiera, es decir, se quita de las ideas primarias que se transforme al español "cualquier" DC, sino que el DC debería ser creado por un único software para poder así establecer una única matriz que estructure a dicho DC. Puesto que si no se conocen y no están prefijados varios parámetros que definen a la estructura visual del DC habría que agregar una inteligencia artificial que logre interpretar dicha imagen, inteligencia que excede de este proyecto.

Las preguntas realizadas en esta encuesta a al Mg. Hernán Amatrian está en el siguiente acceso:

Link: https://1drv.ms/w/s!AgB0dw0E7wKakIIorQf2DUBimWmD1Q

En dicha encuesta o entrevista al equipo docente se establece el lenguaje de programación y las prestaciones que deberá tener la aplicación. Puesto a mi experticia en el desarrollo de aplicaciones bajo la plataforma Android, se plantea la posibilidad de que "Percepción UML" sea desarrollado para la plataforma Android, y así descargado por quien lo desee. El primer impedimento era el hecho de que Cristian quizás no podría utilizar un celular con una pantalla estándar, dicha premisa fue descartada puesto que Cristian usa los celulares con la misma fluidez que cualquier otra persona sin limitaciones visuales, incluso nos ha sorprendido con varias de sus respuestas en esta pseudo entrevista. Puesto que esto no era un impedimento para que sea desarrolla en esta plataforma, pero la tangible experiencia de Hernan Amatrian afirmaba que era una dificultad innecesaria resolver este problema por medio de una aplicación para celulares, sumado a la ya conocida incompatibilidad en varios de los dispositivos móviles, estos dos motivos fueron más que suficientes para establecer que la mejor alternativa encontrada era realizar una aplicación web que se puede ejecutar desde cualquier browser.

Enlace a PERCEPCION UML: http://node24730-uml-dc.jelastic.saveincloud.net/UML-DC/

Luego y como broche de oro a las encuestas, se realizaron encuestas a docentes y estudiantes avanzados de la UNLa, como ya se mencionó anteriormente, donde se les brindaba un DC simple pero completo para que ellos, por medio de su experiencia y relación con la orientación a objetos, me describan con un lenguaje técnico que es lo que veían en el DC.

Con esta última encuesta que se pulió el prototipo número dos, prototipo que incorpora las respuestas de los expertos a su diseño.

La última serie de encuestas se las aprecia en el siguiente link:

Link; https://1drv.ms/b/s!AgB0dw0E7wKakIIfmoGnYiS2BCL1Ig

3.6 PROCESO DE ASIGNACIÓN DEL SISTEMA

Este proceso se realiza cuando el sistema requiere tanto del desarrollo de hardware como de software, o cuando no se puede asegurar que sólo se necesita desarrollo de software.

El Informe de la Necesidad se analiza para identificar las entradas, el procesamiento que se aplica a la entrada, las salidas requeridas y las funciones del sistema total, que permiten desarrollar la arquitectura del sistema e identificar las funciones del hardware, del software y de los interfaces.

Este proceso culmina con la Especificación de Requisitos del Software, la Especificación de requisitos del Hardware y la Especificación del Interfaz del Sistema.

3.6.1 Analizar las funciones del sistema

En este apartado analizaré las funciones que posee la segunda iteración del maquetado evolutivo, es decir la versión final. Y haré algunas menciones al primer prototipo, que solo poseía alguna de las funciones básicas. Hare este recorrido mostrando las funciones operativas que tiene la aplicación, en este segundo bucle de operaciones de software.

El programa Percepción UML, consta de una primera pantalla (Inicio), donde uno puede:

1- Crear un diagrama de clases totalmente nuevo.
2- Trabajar con un DC ya existente, requiere de tener de antemano un usuario y pass, es decir un DC ya creado.
3- Ver o escuchar la documentación asociada al software.
4- Ver los últimos movimientos realizados por usuarios de esta tecnología.
5- Además de opciones no raíces, como ver las noticias sobre la aplicación y sus actualizaciones, o una pestaña de ayuda.

6- Además, dispone de la opción de escribir un DC con nuestro protocolo de generación de DC.

Estas opciones son visibles en la figura 7.

En caso de que uno quiere iniciar como un editor de un DC ya existente, deberá presionar dicho botón, ingresar los datos correctos de acceso, figura 8 de la página 57, esto despliega un menú con todas las versiones existentes del DC seleccionado, se puede trabajar con cualquiera de ellas, borrarlas, duplicarles, editarlas o crear una totalmente nueva.

Al elegir una de las versiones, figura 9 de la página 57, pasamos al menú donde pueden verse todas las clases que tiene esta versión del DC, nuevamente se pueden editar las clases, borrarlas o crear nuevas. Este producto fue diseñado para que en el DC primero se generen todas las clases y luego se vayan rellenando cada una de ellas con sus atributos, métodos, relaciones y características; si bien no es una limitación excluyente del software. En esta pestaña aparece una de las opciones claves de esta aplicación, y es la opción "Percibir DC", donde podremos VER y ESCUCHAR el DC creado o su estructura preliminar.

El resto de los usos de esta aplicación podrán consultarse en el manual de usuario, como por ejemplo la creación desde cero de un DC (figura 10, página 58), que puede descargarse desde cualquier vista de la web de Percepción UML, desde el link a continuación, o desde el anexo de este documento.

Link de manual de usuario: https://1drv.ms/f/s!AgB0dw0E7wKakII-Et0RiYy4BXlChw

Puede apreciarse un producto terminado, bajo el maquetado número 1, en la figura 11 y 12, luego se verá como fue evolucionando dicha interfaz.

Figura 7: Opciones de Percepción UML

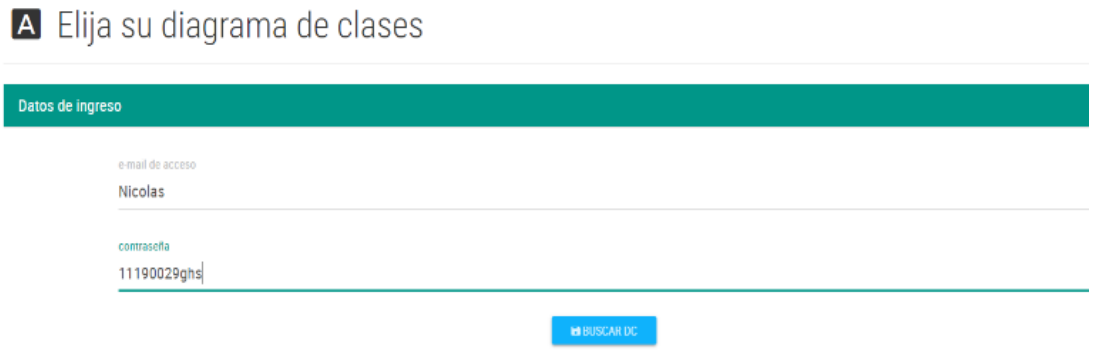


Figura 8: Trabajar con un DC que ya existe.

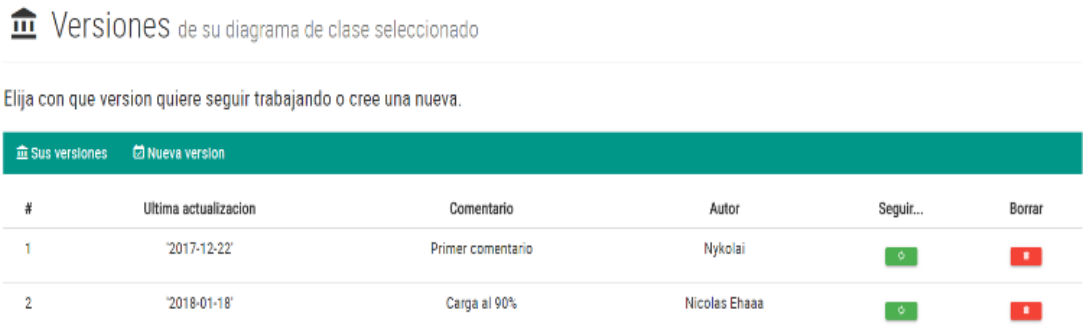


Figura 9: Ver las versiones del DC.

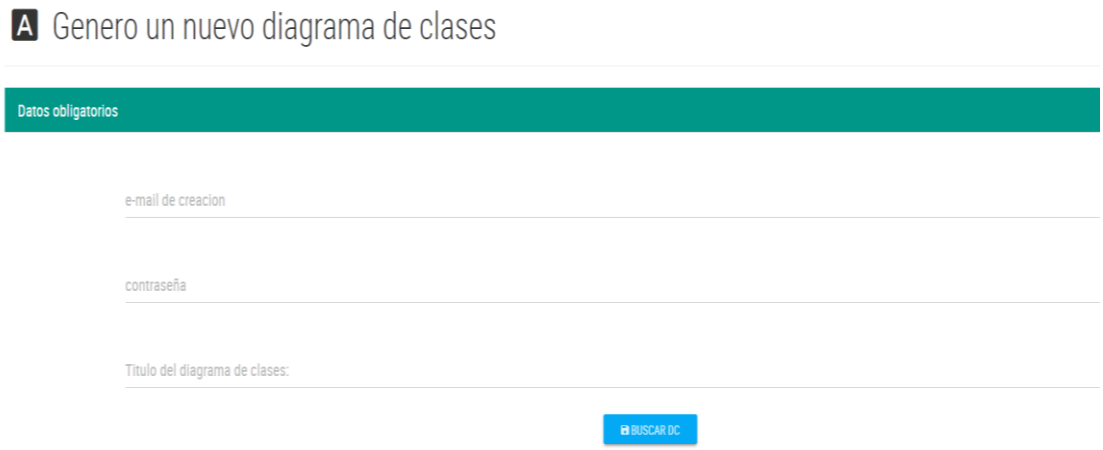


Figura 10: Nuevo DC.

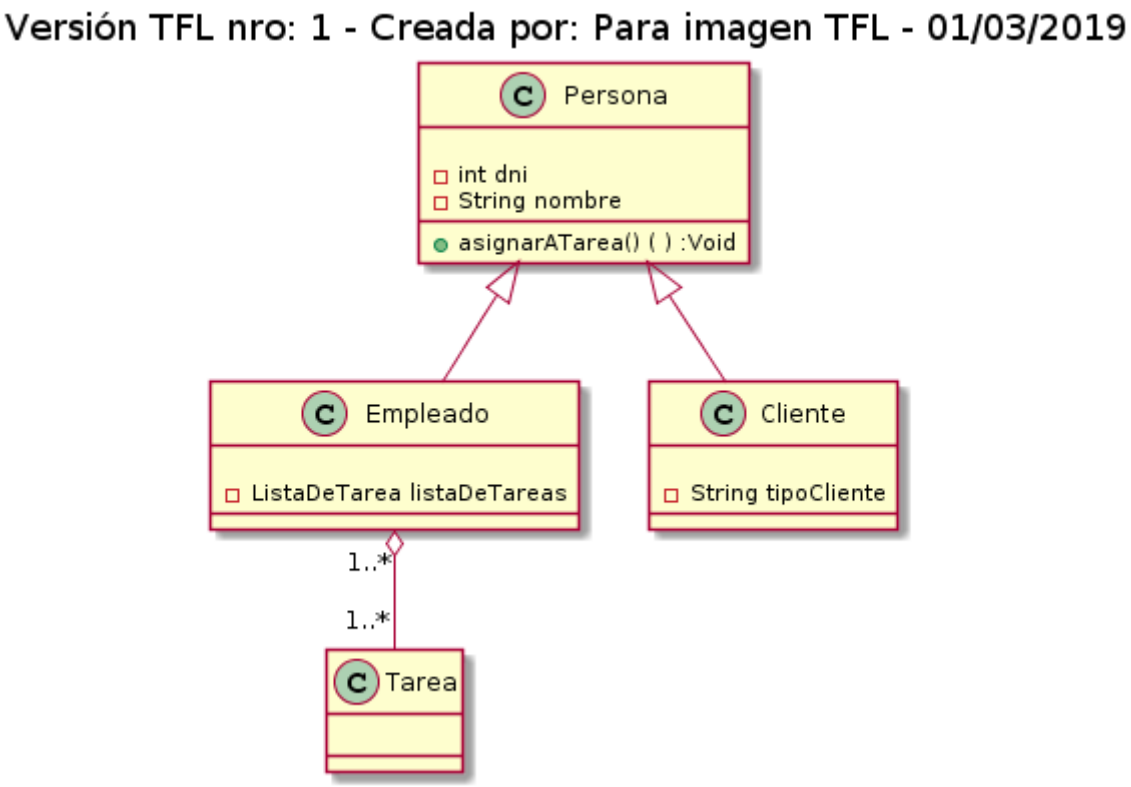


Figura 11: Maquetado 1, vista de un DC

Figura 12: Trabajar con una clase determinada.

Luego del desarrollo casi terminado se definió que no era una complejidad necesaria y que nos brindara algún beneficio el historial de actividades, por eso en la última versión fue eliminado, además de la dificultad que significaba a la hora de la accesibilidad. Con estas mejoras, el ultimo maquetado y posterior versión final se vería como lo adelante la figura 13.

Figura 13: Imagen con estilo simplificado para mejorar la accesibilidad, última versión, ejecutada desde un Browser.

3.6.2 Desarrollar la Arquitectura del Sistema

Si bien, como ya se destacó más atrás, se ha elegido resolver el problema mediante la orientación a objetos, en este punto paso a mostrar la interacción que tendrá el sistema Percepción UML con los diferentes actores externos. Y así ver cómo actúa la Arquitectura del sistema con su entorno. Para

esto realizo un Diagrama de Contexto. En él se pueden ver las entidades y sus flujos de datos con el sistema, figura 14.

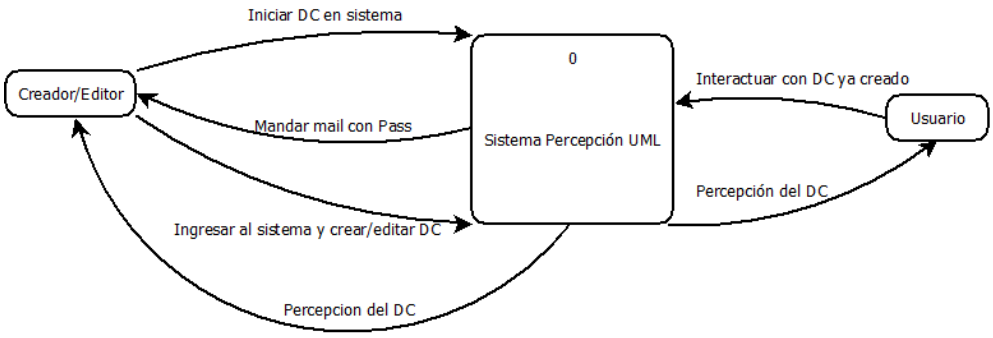


Figura 14. Diagrama de Contexto. Ilustrativo para el flujo de datos.

3.6.3 Descomponer los requisitos del sistema

Esto significa descomponer el sistema en sus componentes para estudiar cada uno de ellos tanto como un ente aislado como en interacción con el resto de los componentes. Se refinó el diagrama de contexto y se empieza el diagramado mediante los objetos apoyándome en el diagrama de contexto estructurado que sirvió de puntapié inicial. Para ello se plantean los diagramas de casos de uso, figura 15.

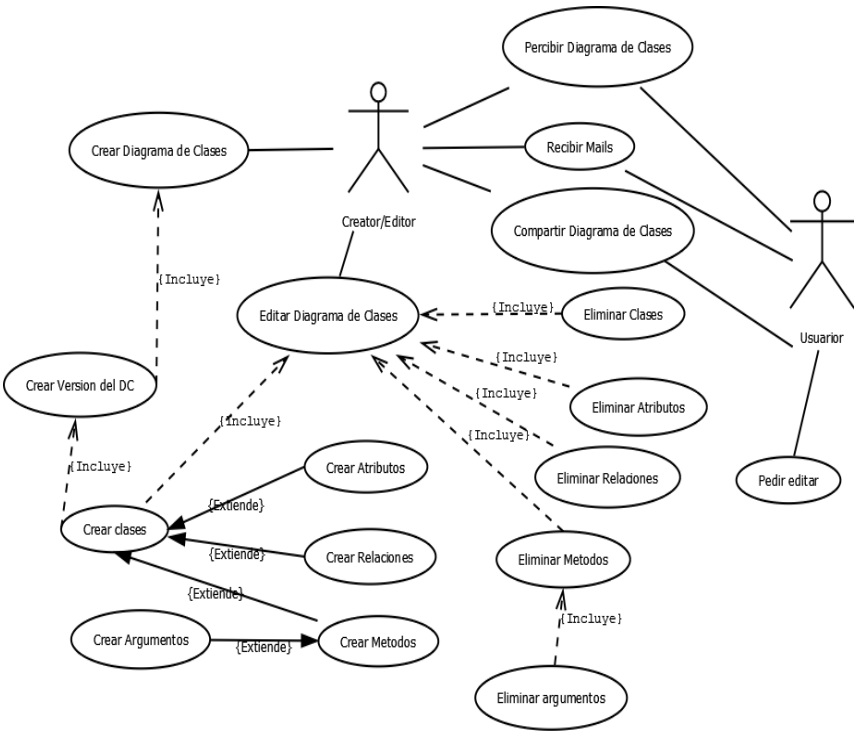


Figura 15: Diagrama de casos de uso.

3.7 PROCESO DE ANÁLISIS DE REQUISITOS

Aquí se asentarán las bases para el establecimiento conciso y preciso de un conjunto de requisitos que deben ser satisfechos por un producto software, indicando, siempre que sea apropiado, el procedimiento mediante el cual se puede determinar si se satisfacen los requisitos dados. Describe los requisitos funcionales, de rendimiento, y de interfaz del software y define los entornos de operación y de soporte. El documento es la salida con que culmina este proceso.

3.7.1 Requisitos del usuario

RU1. Se debe poder generar un diagrama de clases con los estándares establecidos por la IEEE y la UML, por medio de un simple ingreso de datos por pantalla, debe escribirse lo menos posible en dicha aplicación.

RU2. La parte estética del DC debe quedar totalmente desligada del usuario, ya que la visual pasara a 3er plano, muy por detrás de la estructuración y la transformación a audio.

RU3. Es ideal que se pueda realizar un trabajo colaborativo entre varios integrantes de un mismo grupo.

RU3. Los cambios deben poderse respaldar y se tienen que poder crear versiones simultaneas de un mismo DC para entablar una discusión entre participes y lograr ver el progreso o no de este diagramado.

3.7.2 Requisitos del sistema

RS1. El sistema debe ejecutarse desde cualquier dispositivo con conexión a internet.

RS2. El sistema debe poderse utilizar con o sin el uso del audio, debe funcionar aun sin el uso exclusivo de salidas de audio.

RS3. El sistema debe mandar informes vía mail cuando se realice algún cambio significativo.

RS3. El sistema debe tener algún tipo de validación para prohibir que cualquier persona acceda a DC que no debe.

3.7.3 Requisitos funcionales

RF1. El sistema permitirá envió de mails desde una dirección creada para Percepción UML.

RF2. El usuario podrá ver en todo momento como va quedando el DC.

RF3. Los DC se podrán descargar en formato de imagen, texto y audio.

RF3. Estos formatos se enviarán vía email. Principalmente los textos generados.

3.7.4 Requisitos no funcionales

RNF1. El software es íntegramente dependiente de la red de internet, en los momentos donde ésta no esté funcionando la aplicación no será utilizable.

3.7.5 Casos de uso

Un **caso de uso** es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Figura 16.

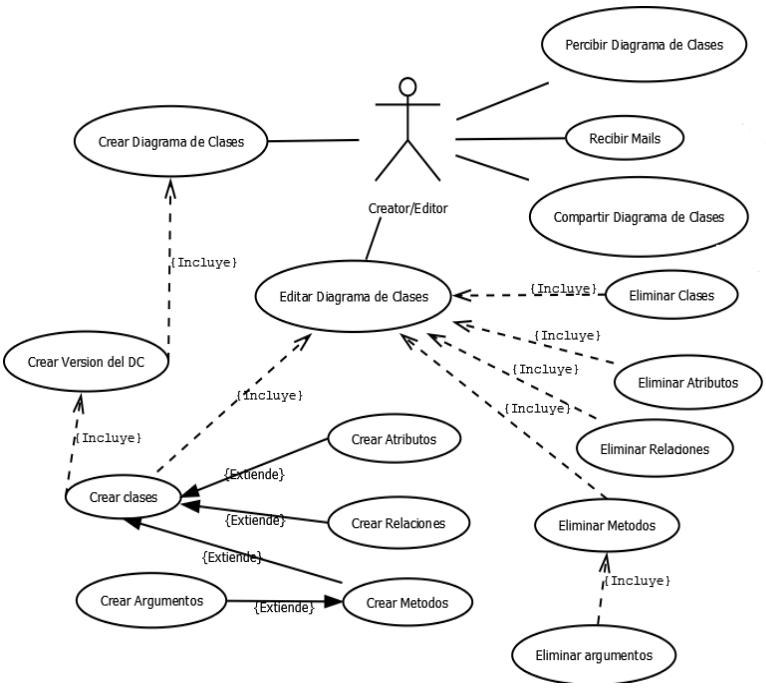


Figura 16: Casos de Usos.

A continuación, dejo la documentación detallada del diagrama de casos de uso (escenarios principales, figura 17).

Nombre del caso de uso: Recibir Mails	ID única: 1
Área: Receptores	
Actor(es): Receptores	
Descripción: Se envía el mail con el protocolo de creación, un audio en formato para descargar mp3, y un texto explicativo del DC	
Activar evento: El creador o editor quiere enviar el mail con los protocolos	
Tipo de señal (Externa o Temporal): Externa	
Pasos desempeñados (ruta principal)	**Información para los pasos**
- Crear un DC con todos los argumentos, atributos, métodos, clases y relaciones	- El Editor o creador debe conocer bien el uso de la interface para utilizar PERCEPCIÓN UML
- Llenar los campos de envió para poder transferir el DC.	- Solo el campo dirección es obligatorio.
Precondiciones: Haber creado una versión del DC	
Postcondiciones: El receptor recibe el mail con todos los adjuntos.	
Suposiciones: El receptor tiene mail.	
Reunir requerimientos: -	
Prioridad: Alta	
Riesgo: Medio	

Nombre del caso de uso: Percibir DC	ID única: 2
Área: Receptores	
Actor(es): Creador /Editor	
Descripción: Una vez creado el DC se lo puede ir visualizando o escuchando	
Activar evento: Querer "ver" la vista preliminar o final del DC	
Tipo de señal (Externa o Temporal): Externa	
Pasos desempeñados (ruta principal)	**Información para los pasos**
- Crear un DC con todos los argumentos, atributos, métodos, clases y relaciones	- El Editor o creador debe conocer bien el uso de la interface para utilizar PERCEPCIÓN UML
- Elegir si uno deseo ver o escuchar el DC	- Movimientos clásicos para reproducir cualquier audio
Precondiciones: Haber creado una versión del DC	
Postcondiciones: Se obtiene el audio o la imagen del DC	
Suposiciones: Se sabe usar la interface de PERCEPCIÓN UML	
Reunir requerimientos: -	
Prioridad: Media	
Riesgo: Alto	

Figura 17: Escenarios de casos de uso, principales.

3.8 PROCESO DE DISEÑO

Aquí se definen los pasos realizados para la creación de las dos maquetas o mejor dicho sus dos iteraciones. Donde muchas menciones fueron hechas en el punto correspondiente a la asignación del sistema.

3.8.1 Realizar el diseño preliminar

Para el diseño preliminar de la web, se utilizó la aplicación Eclipse, para la programación en JAVA, JSP y HTML, y las librerías de estilos provistas por Material Design. En la primera maqueta, desechable se construyó una aplicación que simplemente, dado un diagrama de clases estático, lo dibujaba y lo transformaba a un texto y a voz, esta maqueta fue diseñada únicamente para el entendimiento del problema y la incorporación de las tecnologías más indicadas para solucionar el problema.

3.8.2 Analizar el flujo de información

El flujo de información relacionado con el Sistema Percepción UML viene dada por los usuarios que ingresan los datos necesarios para un mínimo reconocimiento, luego generan datos que permiten crear el DC, estos datos claves son guardados en una base de datos, que luego es leída para generar el grafico, los textos y principalmente el audio del DC.

Para dar una muestra macro visual se realizó un Diagrama HIPO panorámico general (figura 18).

Me pareció pertinente recordar cuestiones básicas sobre este tipo de diagramas.

Definición:

El diagrama Hipo es aquel que indica cuáles son las entradas a un proceso, después la elaboración de un proceso y también las salidas de un proceso.

FINES:

Este método fue creado con el propósito de ayudar a los diseñadores a no perder la pista de alguna función dentro de un sistema grande.

DESARROLLO

Fueron desarrollados por IBM como esquemas de representación para un desarrollo jerárquico de arriba a abajo y como una ayuda de documentación para productos comercializados.

Un conjunto de diagramas HIPO contiene una tabla visual de contenido, un conjunto de diagramas generales y un conjunto de diagramas de detalles.

Los diagramas HIPO necesitan considerablemente cantidad de espacio gráfico, con el fin de ver todo el programa completo, son necesarias varias páginas, los diferentes niveles de diagramas ocupan también espacio, y en ocasiones es fácil el flujo del programa.

Entrada	Proceso	Salida
Login a la plataforma Crear DC o ingresar a DC Datos necesarios que definan al DC	Se genera la estructura clásica de un DC, ya sea métodos, atributos, relaciones, clases, etc.	Se crea una imagen, un audio, un texto coloquial y nuestro protocolo
HIPO para la interface grafica		

Entrada	Proceso	Salida
Conocer el protocolo Tipear el protocolo	Se genera la estructura clásica de un DC, ya sea métodos, atributos, relaciones, clases, etc.	Se genera una imagen que describe al DC
HIPO para usar nuestro protocolo		

Figura 18: Diagrama HIPO panorámico general

3.8.3. Diseñar la base de datos

En este apartado se lleva a cabo el diseño de cada dato que se va a guardar en la base de datos y sus respectivas relaciones.

Para eso se realiza un Diagrama de Entidad de Relación (DER) en Workbrench donde se colocó el nombre de las entidades, sus atributos y relaciones. Figura 19.

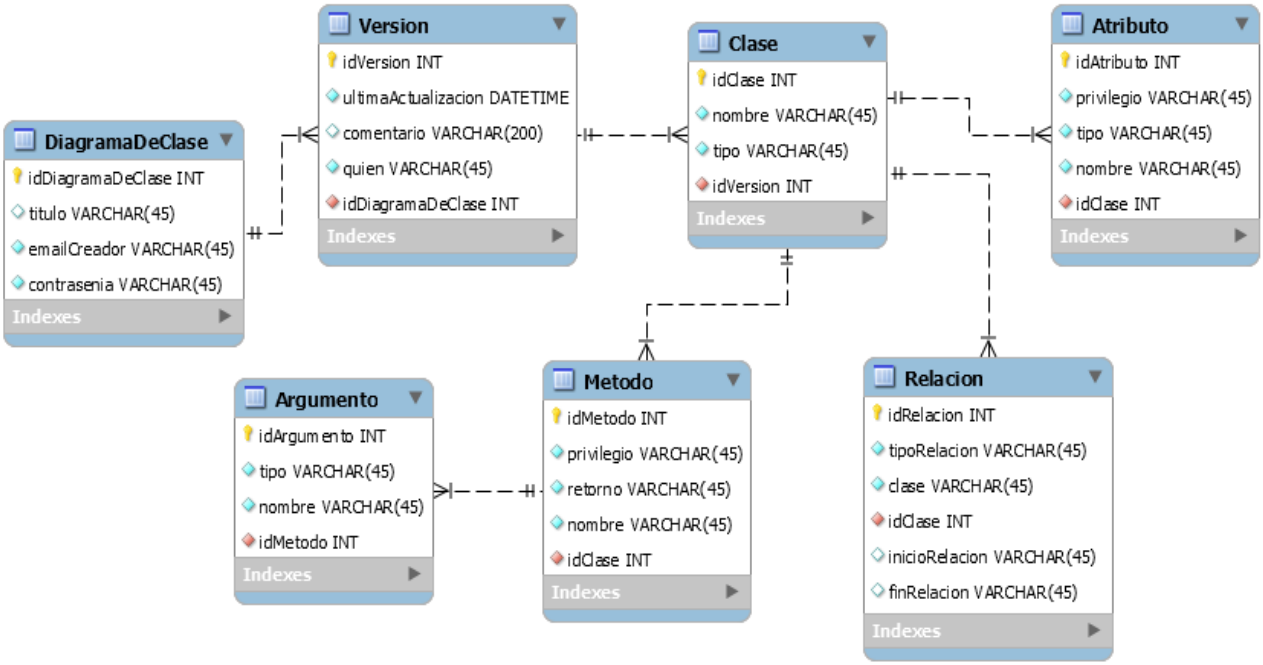


Figura 19: Diagrama de entidad relación, utilizado para guardar los datos generados por la interfaz de PERCEPCIÓN UML

3.8.3. Diseñar las interfaces

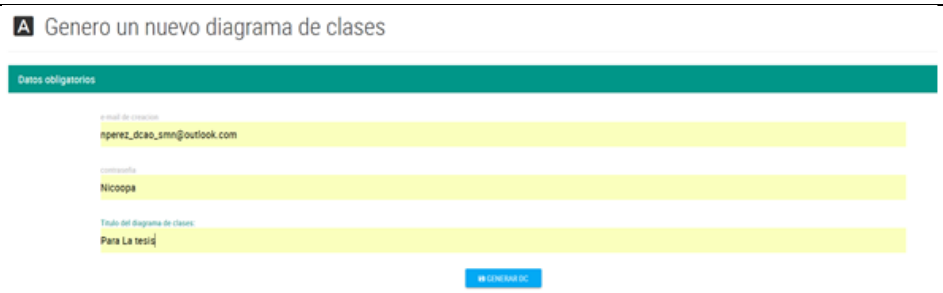
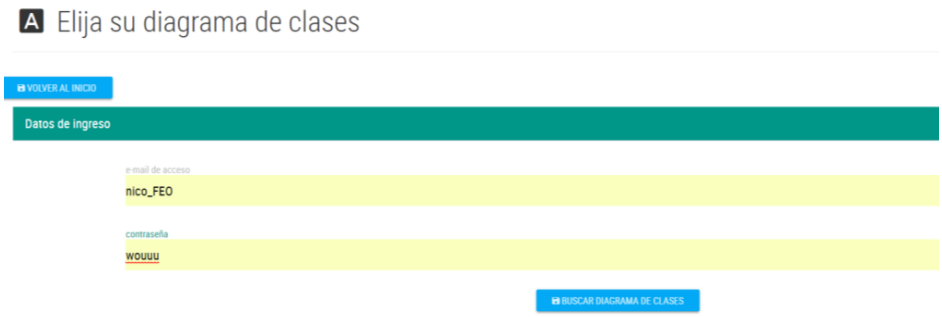
El Diseño de las interfaces se puede ver en las figuras mencionadas en PROCESO DE ASIGNACIÓN DEL SISTEMA. Figuras 6 a 13 sería una muestra de la interfaz web. Luego de la entrevista con su futuro usuario y especialistas en la OO, y de una devolución se realizarán los cambios según las sugerencias, si las hay, y se pasara a realizar las tareas de diseño detallado del sistema.

Como se indicó anteriormente luego de las devoluciones del usuario se realizarían cambios, tales como el cambio del texto generado, el agregado de una verificación de usuario, y la manera de seleccionar los datos mediante varios checklist desplegable con datos para tratar de disminuir al mínimo los campos ingresados por teclado.

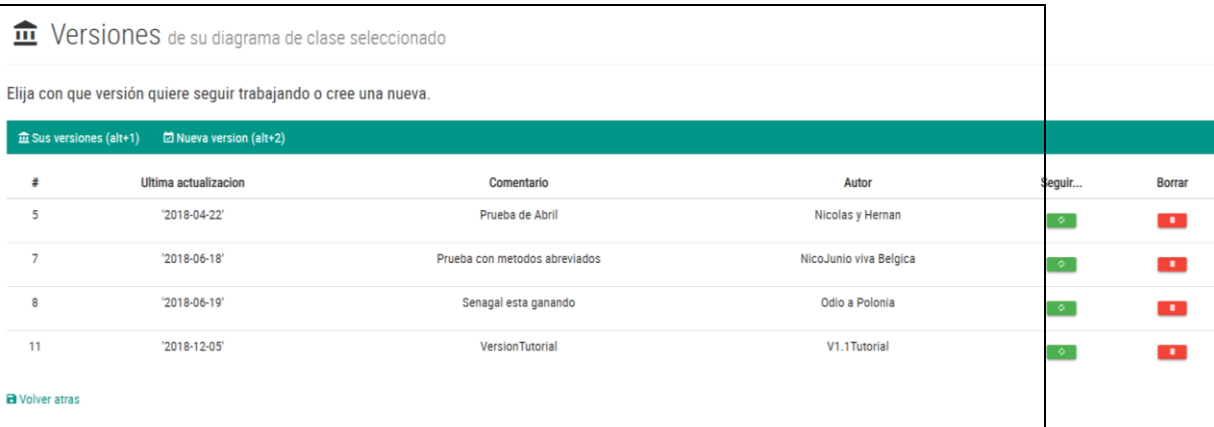
3.8.4 Diseño de interfaz gráfica

En base a todo lo planteado en el apartado 3.8.2, se generó la última interfaz gráfica. La misma con dos grandes aristas, modo visual con interacción por medio de campos de texto y opciones múltiples (con el plus de métodos abreviados para ingresar a cada opción), y un modo auditivo pensado únicamente para la interacción por medio del teclado, y las clásicas teclas de accesibilidad SHIFT, TAB, y ENTER.

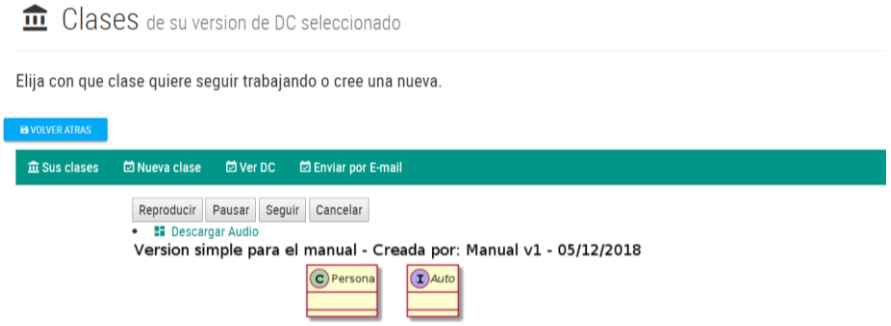
El recorrido por la interfaz final puede consultarse detalladamente en el manual de usuario adjunto en el anexo a este documento, o bien en esta breve tabla descriptiva de las opciones principales:

Opciones principales:	
Login:	
Selección de un DC:	

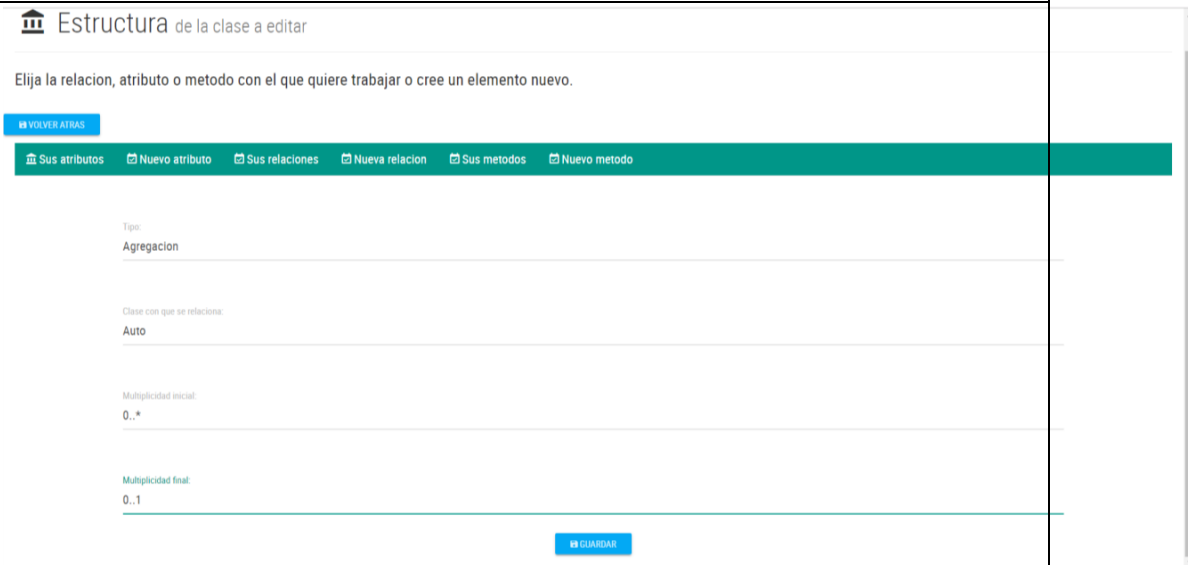
Versiones de un mismo DC:



Menú percibir un DC:



Modificación de estructura:



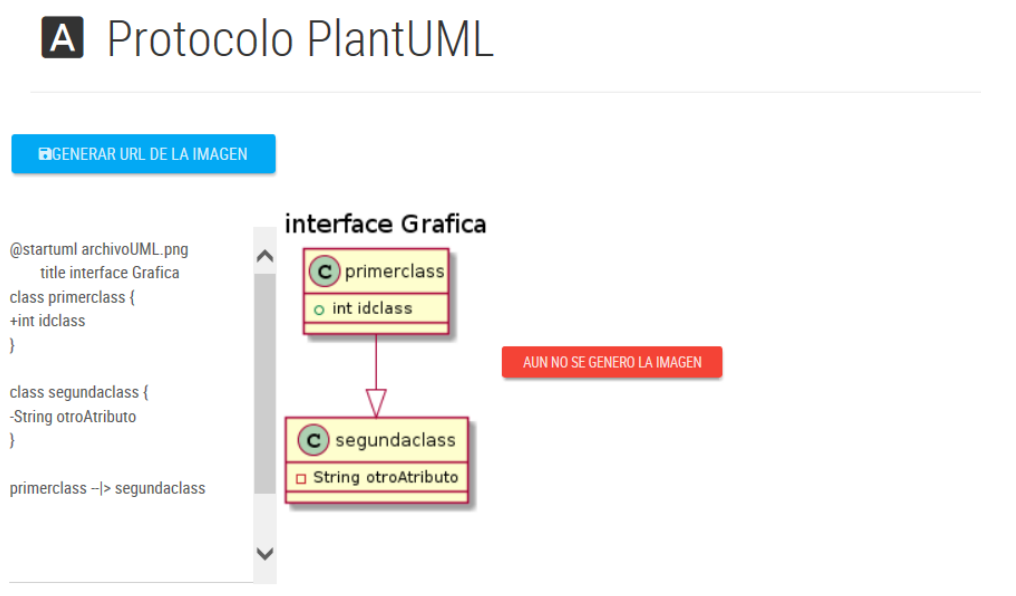
Introducir protocolo:	

Tabla 8: Navegación por la página PERCEPCIÓN UML

3.9 PROCESO DE IMPLEMENTACIÓN E INTEGRACIÓN

Este proceso transforma la representación del diseño detallado de un producto software a una realización en un lenguaje de programación apropiado. En este proceso se debe integrar el código y la base de datos.

3.9.1. Crear la documentación de operación

Todo el software, se lo puede encontrar documentado y versionado en:

Link: https://1drv.ms/f/s!AgB0dw0E7wKakIIU682sdWEyghwxHQ

3.9.2 Crear los datos de prueba

Este apartado fue esencial en el proyecto puesto que por lo general, creemos que llegada la fase de ejecución de pruebas, se nos podrán ocurrir datos lo suficientemente útiles para ejecutar nuestros casos de prueba sin mayor problema; sin embargo, lo “suficientemente útiles” se queda corto cuando se tiene la expectativa de detectar esos defectos que han sido inyectados desde etapas tempranas del desarrollo del software; de modo que, más bien debiéramos buscar preparar datos lo

"suficientemente correctos" para cada prueba, de manera que alcancemos una mayor eficiencia en nuestros ciclos de ejecución.

Entonces es el proceso de crear/preparar y utilizar adecuadamente datos de prueba "realistas" para propósitos distintos de los de producción; por ejemplo, para su uso por áreas de Desarrollo, Testing, Capacitación, etcétera (figura 20). Y como Datos de Prueba podemos considerar todas aquellas entradas con las que es alimentado el sistema para ser operado en períodos de evaluación, ya sea que se introduzcan como datos directamente alimentados desde la interfaz de usuario, desde archivos precargados en distintos formatos (xml, .jpeg, etc.), registros tomados desde las tablas de la base de datos, archivos de configuración, archivos de datos generados por el mismo sistema que servirán de entrada para algún otro proceso, así como también datos que ya deben existir en el sistema para que dichos casos de prueba puedan realmente ejecutarse.

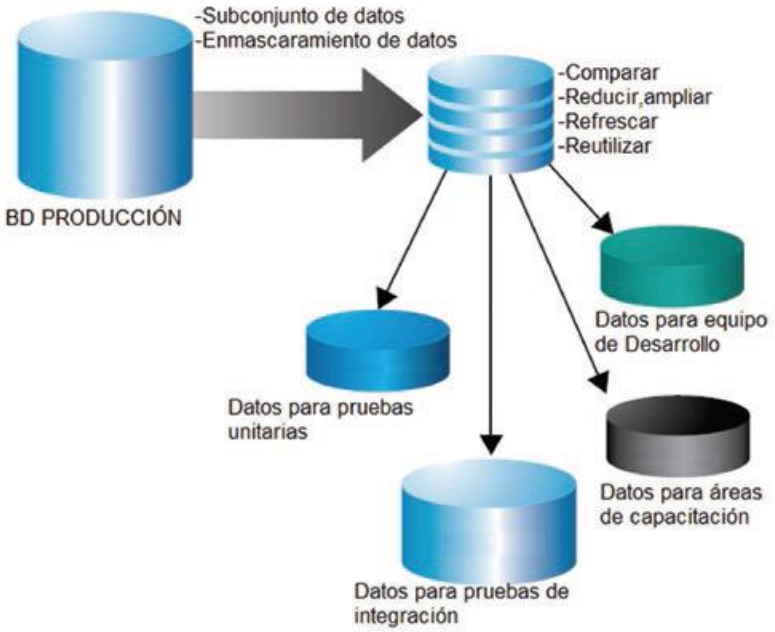


Figura 20: Creación de pruebas

A continuación, se dejan los casos de prueba ejecutados y las pruebas de stress que se realizaron y como estas repercutieron en los ajustes finales.

Casos de prueba más significativos (interfaz gráfica, detallados en la tabla 5 y el resultado en la figura 21). Mientras que se puede encontrar la prueba de stress en la figura 22.
En la prueba de estrés se generó un DC con múltiples clases, atributos, métodos y relaciones. Se genero un DC con 16 clases, 20 relaciones y más de 40 atributos para ver si el programa se

comportaba de la forma correcta. No se encontraron errores en la generación, el tiempo de la creación de la imagen era inferior a los dos segundos, el audio en mp3 quedo con una duración de poco menos de 11 minutos, mientras que el protocolo y el texto coloquial se adjuntan en este documento.

Protocolo generado con la prueba de estrés;

"Titulo Prueba con metodos abreviados - Creada por: Nico - 18/06/2018
Clase AeroPuerto InicioClase

-int idAeropuerto
-String nombre
-String direccion
#String telefono

+mostrarPrecios (String fecha , Int puntos) :String
FinClase
Clase Avion InicioClase

-int idAvion
+int cantidadDePasajeros
+float pesoMaximo

+repararComponente (Componente componente) :String
FinClase
Clase Componente InicioClase

-int idComponente
-String nombre

+calcularPrecio ( ) :Int
FinClase
Clase Precio InicioClase

-int idPrecio
-float descuento
-int tipo

FinClase
Clase Empleado InicioClase

-int legajo
-int anioDeIngreso

FinClase
Clase Pasajero InicioClase

-int numeroViajero
-float precioPagado

+hacerCheckIn ( ) :Void
+subirAAvion (Int numeroAvion) :Void
FinClase
Clase Persona InicioClase

-int dni
-String nombre

-String apellido
+String celular

+pedirAumento (Calendar fecha , ListaDePrecio precios) :float
FinClase
Interface PantallaPasajero InicioClase

-String fechaActualizacion

+mostrarArrivos ( ) :String
+mostrarTiempoDeDespegue ( ) :String
FinClase
ClaseAbstracta FuncionesPiloto InicioClase

+mostrarRuta ( ) :String
+mostrarTripulacion ( ) :String
FinClase
Clase ClaseStress1 InicioClase

FinClase
Interface ClaseStress2 InicioClase

FinClase
enum ClaseStress3 InicioClase

FinClase
Interface ClaseStress4 InicioClase

FinClase
Clase ClaseStress5 InicioClase

FinClase

AeroPuerto UnoAMuchos ComposicionDesde UnoAMuchos Avion
AeroPuerto UnoAMuchos AgregacionDesde UnoAMuchos Persona

Avion UnoAMuchos ComposicionDesde UnoAMuchos Componente

Componente UnoAMuchos Implementa CeroAUno Precio

Empleado HeredaA ClaseStress5

Pasajero Implementa PantallaPasajero

Persona HeredaDe Empleado
Persona HeredaDe Pasajero

PantallaPasajero CeroAUno ComposicionDesde UnoAMuchos ClaseStress1

FuncionesPiloto Implementa Empleado

ClaseStress1 HeredaA ClaseStress2
ClaseStress1 CeroAUno ComposicionDesde UnoAMuchos Precio
ClaseStress1 AgregacionDesde ClaseStress3

ClaseStress3 CeroAUno ComposicionDesde UnoAMuchos Persona
ClaseStress3 CeroAMuchos AgregacionDesde UnoAMuchos ClaseStress1
ClaseStress3 HeredaA Empleado

ClaseStress4 ComposicionDesde ClaseStress3

ClaseStress5 AgregacionDesde AeroPuerto"

Texto coloquial en la prueba de estrés:

"Diagrama de clases, creado por Nico el día 18/06/2018 ésta compuesto por:

Clase pública denominada AeroPuerto, donde sus atributos son:
Un atributo privado del tipo int denominado idAeropuerto.
Un atributo privado del tipo String denominado nombre.
Un atributo privado del tipo String denominado direccion.
Un atributo protegido del tipo String denominado telefono.
Un Método publico con nombre mostrarPrecios. Con un argumento del tipo String con nombre fecha. Con un argumento del tipo Int con nombre puntos. , este método retorna un String.

Clase pública denominada Avion, donde sus atributos son:
Un atributo privado del tipo int denominado idAvion.
Un atributo publico del tipo int denominado cantidadDePasajeros.
Un atributo publico del tipo float denominado pesoMaximo.
Un Método publico con nombre repararComponente. Con un argumento del tipo Componente con nombre componente. , este método retorna un String.

Clase pública denominada Componente, donde sus atributos son:
Un atributo privado del tipo int denominado idComponente.
Un atributo privado del tipo String denominado nombre.
Un Método publico con nombre calcularPrecio., este método retorna un Int.

Clase pública denominada Precio, donde sus atributos son:
Un atributo privado del tipo int denominado idPrecio.
Un atributo privado del tipo float denominado descuento.
Un atributo privado del tipo int denominado tipo.

Clase pública denominada Empleado, donde sus atributos son:
Un atributo privado del tipo int denominado legajo.
Un atributo privado del tipo int denominado anioDeIngreso.

Clase pública denominada Pasajero, donde sus atributos son:
Un atributo privado del tipo int denominado numeroViajero.
Un atributo privado del tipo float denominado precioPagado.
Un Método publico con nombre hacerCheckIn., este método retorna un Void.
Un Método publico con nombre subirAAvion. Con un argumento del tipo Int con nombre numeroAvion. , este método retorna un Void.

Clase pública denominada Persona, donde sus atributos son:
Un atributo privado del tipo int denominado dni.
Un atributo privado del tipo String denominado nombre.
Un atributo privado del tipo String denominado apellido.
Un atributo publico del tipo String denominado celular.
Un Método publico con nombre pedirAumento. Con un argumento del tipo Calendar con nombre fecha. Con un argumento del tipo ListaDePrecio con nombre precios. , este método retorna un float.

Interface denominada PantallaPasajero, donde sus atributos son:
Un atributo privado del tipo String denominado fechaActualizacion.
Un Método publico con nombre mostrarArrivos., este método retorna un String.
Un Método publico con nombre mostrarTiempoDeDespegue., este método retorna un String.

Clase abstracta denominada FuncionesPiloto, sin Atributos definidos.
Un Método publico con nombre mostrarRuta., este método retorna un String.
Un Método publico con nombre mostrarTripulacion., este método retorna un String.

Clase pública denominada ClaseStress1, sin Atributos definidos.

Interface denominada ClaseStress2, sin Atributos definidos.

Enum denominado ClaseStress3, sin Atributos definidos.

Interface denominada ClaseStress4, sin Atributos definidos.

Clase privada denominada ClaseStress5, sin Atributos definidos.

El Diagrama de clase posee las siguientes relaciones entre las clases:

La clase AeroPuerto con una relación de composición con la clase Avion.Además esta relación va desde 1 a muchos hasta 1 a muchos.

La clase AeroPuerto con una relación de agregación con la clase Persona.Además esta relación va desde 1 a muchos hasta 1 a muchos.
La clase Avion con una relación de composición con la clase Componente.Además esta relación va desde 1 a muchos hasta 1 a muchos.
La clase Componente con una relación de implementación con la clase Precio.Además esta relación va desde 1 a muchos hasta cero a uno.
La clase Precio no tiene relaciones a nombrar.
La clase Empleado con una relación de herencia con la clase ClaseStress5.
La clase Pasajero con una relación de implementación con la clase PantallaPasajero.
La clase Persona con una relación de herencia con la clase Empleado.
La clase Persona con una relación de herencia con la clase Pasajero.
La clase PantallaPasajero con una relación de composición con la clase ClaseStress1.
Además esta relación va desde cero a uno hasta 1 a muchos.
La clase FuncionesPiloto con una relación de implementación con la clase Empleado.
La clase ClaseStress1 con una relación de herencia con la clase ClaseStress2.
clase ClaseStress1 con una relación de composición con la clase Precio.Además esta relación va desde cero a uno hasta 1 a muchos.
La clase ClaseStress1 con una relación de agregación con la clase ClaseStress3.
La clase ClaseStress2 no tiene relaciones a nombrar.
La clase ClaseStress3 con una relación de composición con la clase Persona.Además esta relación va desde cero a uno hasta 1 a muchos.
La clase ClaseStress3 con una relación de agregación con la clase ClaseStress1.
Además esta relación va desde cero a muchos hasta 1 a muchos.
La clase ClaseStress3 con una relación de herencia con la clase Empleado.
La clase ClaseStress4 con una relación de composición con la clase ClaseStress3.
La clase ClaseStress5 con una relación de agregación con la clase AeroPuerto."

En última instancia de prueba de estrés se crea un diagrama de clases con múltiples clases, pero utilizando nuestro protocolo. La imagen generada fue exitosa y fue la siguiente (figura 21):

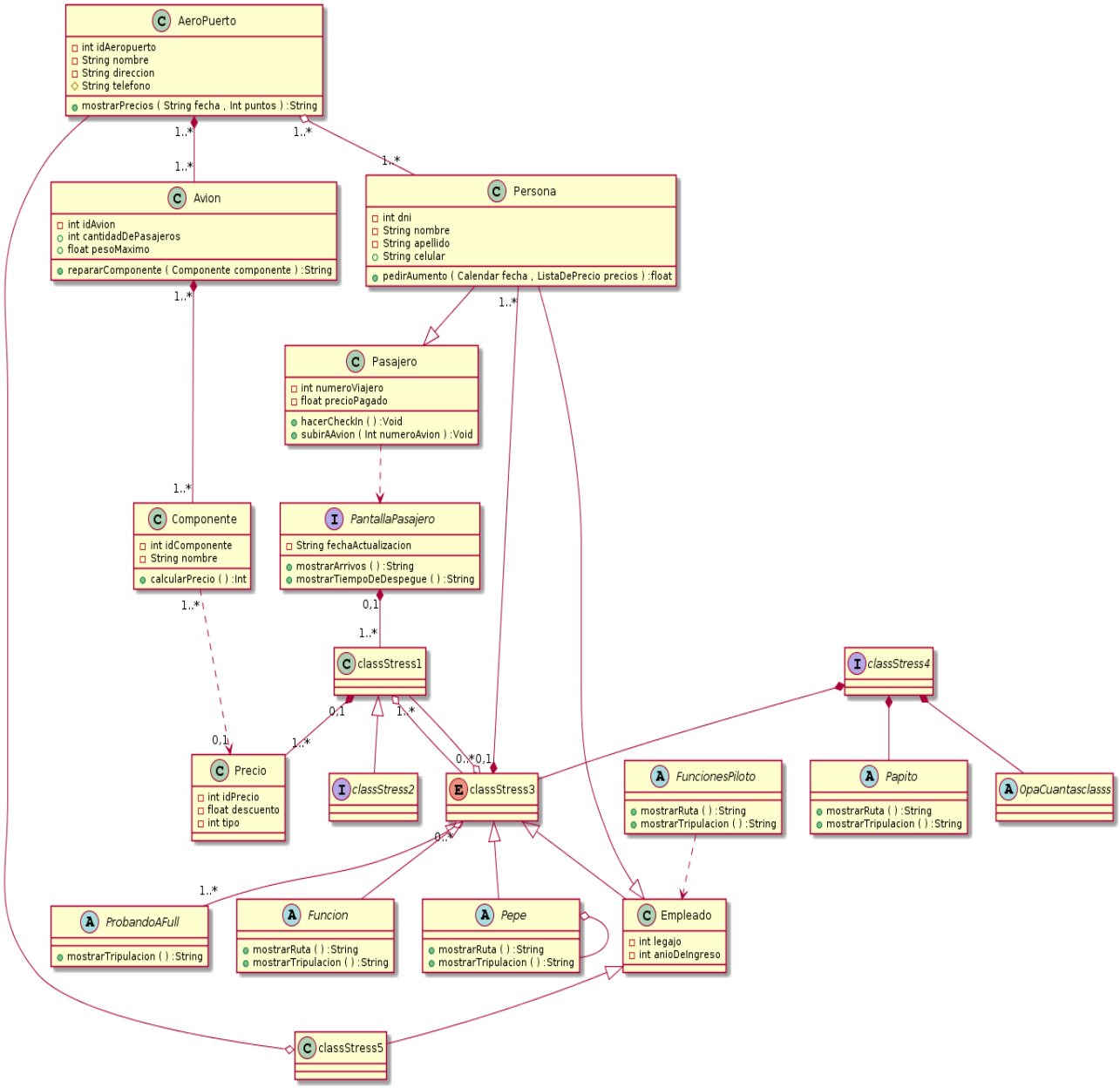


Figura 21: Prueba de stress exigiendo al máximo a la interfaz web y a los protocolos, audios y visualización. Se exigió al máximo el uso de las opciones disponibles en el programa, dejando de lado la lógica del DC.

Id	Caso de Prueba	Descripción	Área Funcional	Funcionalidad	Datos	Resueltado esperado	Requerimientos	Resultado	Estado	Observaciones
1	Creacion de usuario con pass	Se trata de crear un usuario	Ingreso al sistema	Crear el usuario en la BD	email, nombre diagrama	Se permite el ingreso al sistema	Tener email y conexión a internet	Muy satisfactorio	ok	Se creo perfectamente el DC para ese email
2	Creacion de usuario sin pass	Se trata de crear un usuario ingresando	Ingreso al sistema	Crear el usuario en la BD	email, nombre diagrama de clases	Se permite el ingreso al sistema	Tener email y conexión a internet	Muy satisfactorio	ok	Se creo perfectamente el DC para ese email, ademas se envia por mail el pass de ingreso.
3	Ingreso al sistema con DC ya creado anteriormente	Se pide trabajar con un DC ya creado	Ingreso al sistema	Se llama por medio del email y contraseña a un DC ya	email, pass	Se ven las versiones del DC seleccionado o se devuelve	Conocer el email y el pass que llevan al DC	Muy satisfactorio	ok	Si los datos son incorrectos se redirecciona a la solapa de selección
4	Ingreso al sistema con un usuario sin	Al ingresar a un DC que aun no se inicio no	Creacion de DC	Generar DC y versiones	email, pass	Lista de versiones vacias, es necesario	Conocer el email y el pass que llevan al DC	Muy satisfactorio	ok	La lista de versiones se vera vacia y solo tendra la opcion de crear una nueva
5	Crear nueva version del DC	Se crea una version del DC	Creacion de DC	Generar DC y versiones	Comentarios y nombre de la	Creacion de esta version	Ninguno	Muy satisfactorio	ok	Se pudo crear la version con o sin lo datos, se le asigna un id y la fecha de creacion
6	Borrar versión del DC	Se borra una version seleccionada	ABM de la BD	Borrar la version de la BD	Luego de visualizar las versiones	Se espera que deje de viualizarse esa version de la	Ninguno	Muy satisfactorio	ok	Se pudo borrar las versiones
7	Seguir trabajando con una version	Se ingresa a una version seleccionada	ABM de la BD	Se tiene acceso a los datos de la version	Ingreso de nuevas	Se visualizan todas las clases que forman a la	Ninguno	Muy satisfactorio	ok	En caso que ya existan clases se las puede PERCIBIR, en caso que
8	Volver a ver versiones del DC	Se quiere volver atrás	Retroceder	Se vuelve al menu anterior	clic en boton atrás	Volver a la interface anterior	Ninguno	Satisfactorio	Se puede mejorar en futuras actualizaciones	El boton de la aplicación funciona perfectamente, pero el boton atrás del browser puede fallar y perder consistencia, para evitar esto se vuelve al
9	Crear clase en una version	Se quiere agregar una clase a una determinada version	ABM de la BD	Se espera modificar la version con la nueva clase, es necesario que se previsualice el cambio	nombre(Opcional y deseable) y tipo de la clase.	Se agrega la clase a la version, la misma se puede percibir.	Generar la clase	MALO	Solcionado al 100%	El campo no obligatorio de nombre no trajo problemas en la BD pero si en el protocolo asi que **se cambia ese campo a obligatorio**
10	Percibir cambios en version	Se quiere ver y escuchar los cambios en las	PERCEPCION	Percibir los cambios realizados en la	clic en el boton percibir.	Se perciben los cambio sin problemas.	Audio activo para escuchar el diagrama de	Muy satisfactorio	ok	Se genera sin problemas el audio y la imagen ya sea con una version vacia o
11	Enviar por mail DC	Se quiere enviar por email el DC	Envio	Enviar por mail el DC	email del receptor, nombre y asunto (opcional)	Recibir mail con los protocolos y el audio	Conexión a internet	Regular	Solucionado al 100%	Si bien no se encontraron errores groseros se encontro un detalle no menor que era que los datos del receptor no eran obligatorios, **ahora ese dato y el nombre del emisor son obligatorios.**
12	Recibir mail	Se quiere recibir el mail con todos los datos adjuntos	Envio	Luego del envio se espera que el receptor reciba los datos del DC	email del receptor, nombre y asunto (opcional)	Recibir los txt con los textos de protocolos y la descarga del audio.	Cuenta de email	Bueno	ok	Se descubre que mandar el audio como adjunto tiene problemas con algunas casillas de email por cuestiones de seguridad. Asi que se envia un link de
13	Borrar Clase	Se quiere borrar una clase , sus	ABM de la BD	Borrar clase	clic en la clase a borrar	Debe desaparecer la clase del DC, se	Ninguno	Muy satisfactorio	ok	
14	Ver listas de la clase	Se quiere ver las listas de atributos, relaciones y	ABM de la BD	Ver estructura de la clase	Elegir que se quiere ver	Se deben ver las listas que forman a la clase	Ninguno	Muy satisfactorio	ok	En esta parte de la interface solo se proponen imágenes, sin texto y audio. Pero la pagina web
15	Agregar atributos a la clase	Se quiere agregar un atributo a alguna de las clases	ABM de la BD	Modificar Clase	nombre, tipo, privilegia del atributo	Debe quedar reflejado en la base de datos el cambio en la clase	Completar todos lo campos	Regular	ok - Mejorado	Gracias a esta prueba se encontro una error logico que si bien no arruinaba el sw no era tenido encuenta. **Luego de esta prueba se eliminan los espacios en el nombre de los atributos. Se lo cambia por un "_". Este error sirvio para reparar conflictos semejantes en otras funciones.**
16	Agregar metodo a la clase	Se quiere agregar un metodo a	ABM de la BD	Modificar Clase	nombre, tipo, privilegio	Debe quedar reflejado en la base de datos el	Completar todos lo campos	Muy satisfactorio	ok	
17	Agregar argumento a metodos	Se quiere agregar un argumento a	ABM de la BD	Modificar Clase	nombre y tipo	Debe quedar reflejado en la base de datos el	Completar todos lo campos	Muy satisfactorio	ok	
18	Agregar relaciones a las clases	Se quiere agregar una relacion a	ABM de la BD	Modificar Clase	inicio, fin, cardinalidad(opci	Debe quedar reflejado en la base de datos el	Completar todos lo campos	Muy satisfactorio	ok	
19	Ir atrás	Se quiere volver atrás en cualquier momento del sw	Consistencia	Volver atrás	clic en boton atrás	Poder volver atrás en todo momento	Ninguno	Regular	Solucionado al 100%	Se notaron imperfecciones en esta actividad, pero se lo reparo.

Tabla 9: Casos de prueba principales para la interfaz web.

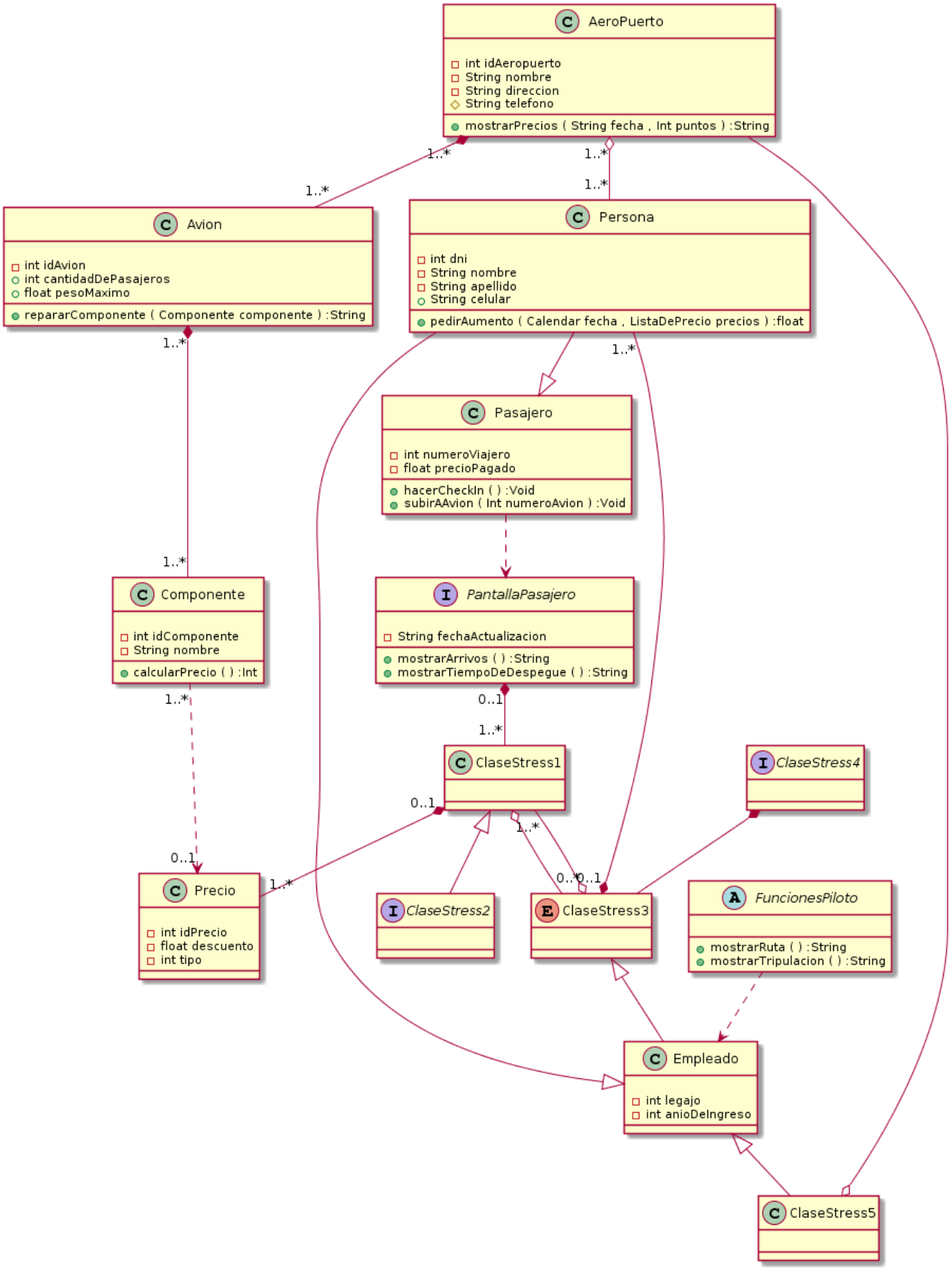


Tabla 10: Resultados de pruebas de interfaz web. Notar la flexibilidad de la aplicación, como las relaciones pueden ser en ambos sentidos, definiendo o no la cardinalidad de las mismas, como las clases pueden ser de varios tipos, tener o no estructura interna, y como en caso de tener un error puede ser modificado simplemente desde el protocolo o interfaz.

3.9.3 Crear el código fuente

Como se mencionó anteriormente el código fuente fue desarrollado en lenguaje Java, por medio del IDE Eclipse, con la tecnología Hibérnate para la conexión con la base de datos montada sobre MYSQL server, además se desarrolló la interface grafica por medio de HTML y JSP, se generaron las imágenes de los DC por medio de la integración JQuery documentada en www.plantuml.com.

Se programó por medio del paradigma de orientación a objetos utilizando el patrón MVC. Todo el código fuente es de acceso libre para futuras modificaciones o implementaciones. Dicho proyecto completo se lo puede encontrar en:

Link código fuente: https://1drv.ms/f/s!AgB0dw0E7wKakIIXBTZz5-GxjJ4a0Q
Link: GitHub: https://github.com/gruposeminario/PercepcionUML-UNLa
Link Base de datos: https://1drv.ms/f/s!AgB0dw0E7wKakIIgSi5yknFIUQv3Fw
Link hosting: http://node24730-uml-dc.jelastic.saveincloud.net/UML-DC/

3.9.4 Planificar la integración

Aquí se realizaron las ya conocidas pruebas de integración. Estas pruebas valoran si los componentes individuales trabajan en conjunto tal y como se espera de ellos. O lo que es lo mismo, se prueba el funcionamiento de los diferentes módulos del sistema una vez unidos o agrupados en elementos mayores, verificando el comportamiento de estos frente a las comunicaciones que se produzcan entre ellos. El objetivo es la localización de errores de interfases y comprobar el correcto funcionamiento conjunto de los componentes. Aún en los casos en los que los componentes básicos pasen sin error las pruebas de componentes, debe comprobarse su funcionalidad externa. Seguramente, con las pruebas de integración se localizarían todos los errores dados en las pruebas de componentes, pero sería una tarea más compleja, por ello, es aconsejable realizarlo en dos fases diferenciadas.

En las pruebas de integración debe tenerse en cuenta que existe la posibilidad de tener que reunir los resultados de diferentes desarrolladores y/o equipos de pruebas, pues cada componente puede tener un origen distinto. Y dado que no tiene por qué conocerse el funcionamiento intrínseco de cada módulo, es aconsejable que se utilicen técnicas de caja negra para el diseño de los casos de

prueba y que, además, sean equipos independientes del desarrollo quienes se encarguen de dichas pruebas.

En las pruebas de integración, se diferencian varios tipos de estrategias a seguir, como afirma Carlos Blanco Bueno (2011), Construcción y Pruebas de Software:

- **Top-Down:** En esta estrategia la integración se lleva a cabo sistemáticamente de arriba a abajo. Se empieza por los componentes que no son llamados desde otros componentes del software y paso a paso se integran aquellos componentes que únicamente son llamados desde la parte ya integrada.
- **Bottom-Up:** Se trata de la estrategia de pruebas inversa a Top-Down. En esta estrategia, se lleva a cabo una integración de abajo a arriba. Se empieza por aquellos componentes que no llaman a otras partes del programa y se continúa por los componentes que sólo llaman a la parte ya integrada. Se suele aplicar en desarrollos nuevos, pruebas de equipos grandes y distribuidos.
- **End-to-End:** Es una estrategia de pruebas de integración orientada al proceso de negocio en el cual se integran en cada caso los componentes necesarios por parte de un proceso completo de negocio, o lo que es lo mismo, al emplear la estrategia se valida si el flujo de una aplicación está funcionando tal y como fue diseñado. En la estrategia End-to-End se pueden emplear controladores para sustituir los componentes de rango superior y stubs para los subordinados.
- **Funciones:** consiste en diseñar la integración orientada a una función del sistema, de manera que se integra cada uno de los componentes necesitados por la función correspondiente del sistema. En verdad, este tipo de estrategia por funciones podría decirse que es un subconjunto de la estrategia End-to-End centrada en unas funciones en concreto, con la diferencia de que no tienen por qué ser funciones que validen un comportamiento dado de principio a fin.
- **Ad-Hoc:** en esta estrategia se enfoca el software como módulos independientes y una vez que se finaliza su implementación y validación exitosa por parte de desarrollo, se integran.

La estrategia Top-Down fue desarrollada desde las raíces del código fuente en cada una de las pruebas de desarrollo o test, al igual que la estrategia Botton-Up. Cada vez que se superaba una prueba Top-Down el componente se integraba al código fuente.

Distinto ha sido el uso de end-to-end, donde se lo ha utilizado sólo en salvadas veces por medio de la técnica de sustitución por controladores, por ejemplo, en la etapa de prueba de tecnologías, donde se quería probar el diagramado por medio de PlantUML.
Las funciones del sistema fueron testeadas por ejemplo en los casos de pruebas mencionados en el apartado anterior, mientras que Ad-Hoc no se ha desarrollado en esta aplicación, aunque suele estar implícita en todo el ciclo de vida del proyecto por medio de la integración de pequeñas funcionalidades.

En la integración de esta aplicación planteo la mejora continua, donde el objetivo es mejorar los procesos y procedimientos de dicha metodología. Las actividades que lo componen son (Figura 22):

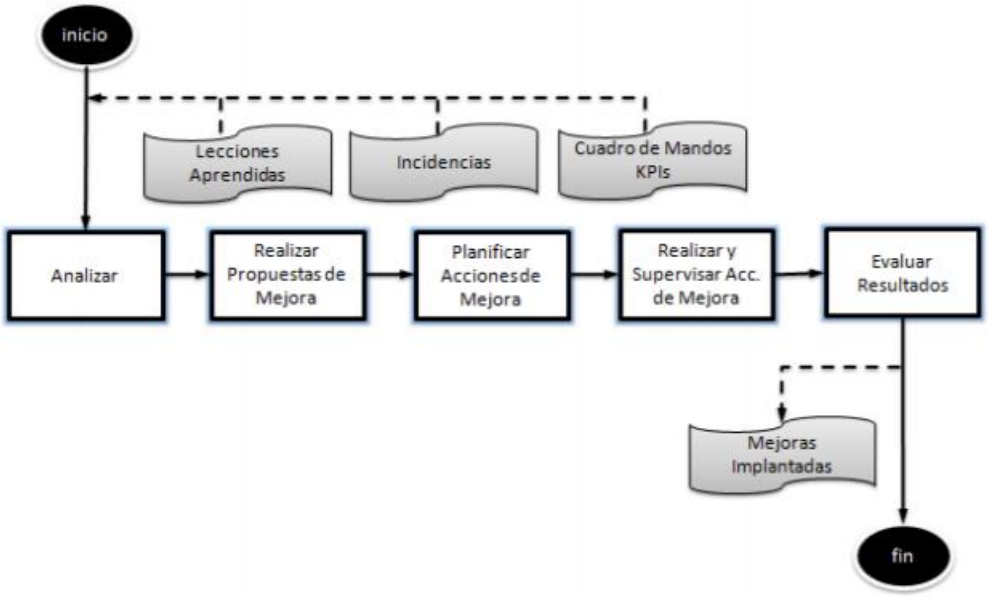


Figura 22: Mejora continua y sus etapas.

También se realizaron las pruebas de stress con casos límites para testear la integración final del SW, se obtuvieron resultados muy buenos, visibilizados en la figura 21, página 76.

3.10 PROCESO DE VERIFICACIÓN Y VALIDACIÓN

La verificación y validación es una actividad que juega un papel importante en la consecución de procesos y productos de calidad. Por ello y dado los problemas que históricamente han surgido en

los desarrollos software, su estudio y mejora suponen un avance importe para la futura evolución de las tecnologías de la información.

En el presente documento se establece la evolución de dicha actividad y su actual impacto en los distintos tipos de organizaciones que las implementan, con el fin de poder extraer algunos de los conceptos más importantes en su implantación y desarrollo, y así elaborar una guía de buenas prácticas que permita seleccionar las mejores opciones para realizar las pruebas de software en función de las características de una organización.

3.10.1 Planificar la verificación y validación

Hoy en día existen dos paradigmas principales que permiten la aplicación de las actividades de V&V en un proyecto, el análisis estático y el dinámico. **El análisis estático** necesita de la ayuda de una herramienta que analice la representación estática del sistema (diagramas de flujo, casos de uso, código fuente, otros) con el fin de descubrir errores.

Este tipo de análisis implica la no ejecución de modelos, ni del código fuente para determinar las posibles fallas. Se trata por lo tanto de verificar los "productos" obtenidos durante el desarrollo del software. Para llevarlos a cabo es habitual el uso de técnicas de inspección, éstas no resultan normalmente un reemplazo de las pruebas de software, sino que las complementan, enfocándose en aquellas diferencias que no son posibles detectar usando únicamente pruebas.

Por el contrario, **el análisis dinámico** implica normalmente la ejecución del código fuente del software con datos de prueba, con el fin de evaluar en tiempo real los diferentes escenarios soportados por el programa.

Las técnicas de inspección comprenden el análisis automático del código fuente y la verificación formal. Por el contrario, las técnicas estáticas sólo pueden comprobar la correspondencia entre un programa y su especificación (verificación), no pudiendo demostrar que el software es operacionalmente útil.

En nuestro proyecto hemos planificado la V&V en 2 segmentos, técnicas estáticas, y técnicas dinámicas, donde:

Las técnicas estáticas son las primeras comprobaciones que se aplican al software. Son técnicas cuyo objetivo es el de mejorar la calidad de los productos del software, ayudando a los ingenieros a reconocer y arreglar sus propios errores en etapas tempranas del proceso de desarrollo sin ejecutar el código. Para ello, buscan defectos sin ejecutar el código, llevándose a cabo una vez se encuentra escrito el código. Estas técnicas tienen que ver con análisis y el control de las representaciones del sistema, es decir con los diferentes modelos construidos durante el proceso de desarrollo de software, tales como los documentos de requerimientos, los diagramas de análisis y de diseño y el código fuente. En esta categoría entran, por ejemplo, las revisiones, las inspecciones de programas, la verificación formal (basada en el uso de notaciones formales con base matemática) y las herramientas de análisis estático (muchas de ellas provistas hoy en día como parte del compilador, como por ejemplo la detección de variables no utilizadas, código inalcanzable, etc.).

Las técnicas dinámicas, también conocidas como pruebas, se basan en ejercitar una implementación. Por lo tanto, sólo pueden ser aplicadas si existe una versión operativa o ejecutable del producto. A pesar de que las técnicas de verificación estáticas son usadas cada vez más, la prueba de los programas sigue siendo la técnica predominante en la validación y verificación. La prueba involucra ejecutar el programa proveyéndole entradas con el fin de detectar la mayor cantidad de defectos posibles. Son diseñadas con el propósito de revelar la presencia de defectos, por ello se dice que un caso de prueba es exitoso cuando los logra poner en evidencia. Además, son consideradas como la única técnica de validación para los requerimientos no funcionales, debido a que el software debe de ser ejecutado para ver su comportamiento, proporcionando confianza en el software.

3.10.2 Planificar las pruebas

Se ejecutarán las pruebas de verificación y validación. Detallando lo ocurrido en la revisión formal del código. Esta tarea será plasmada con las siguientes especificaciones de casos de prueba detallados en la tabla 6.

3.10.3 Desarrollar las especificaciones de las pruebas

En el link siguiente puede descargarse el plan de prueba a ejecutarse:

Link: https://1drv.ms/f/s!AgB0dw0E7wKakII2u_FS52ynW6P4Iw

En la tabla 11a y 11b (página 85 y 86 respectivamente) puede apreciarse los casos de prueba diseñados y ejecutados a la fecha. En esta tabla puede distinguirse la identificación de la prueba, su descripción, el área funcional a probar, la característica de la prueba, los datos que necesita y el resultado esperado.

Además, en la tabla 11.a y 11.b se aprecian los resultados obtenidos de cada caso de prueba, sus dependencias, los requerimientos de ambiente, y los procedimientos especiales requeridos. Las observaciones y el estado final.

Id	Caso de Prueba	Descripción	Área Funcional	Funcionalidad	Datos	Resueltado esperado	Requerimientos	Resultado	Estado	Observaciones
1	Creacion de usuario con pass	Se trata de crear un usuario ingresando todos los datos solicitados	Ingreso al sistema	Crear el usuario en la BD	email, nombre diagrama de clases, pass	Se permite el ingreso al sistema	Tener email y conexión a internet	Muy satisfactorio	ok	Se creo perfectamente el DC para ese email
2	Creacion de usuario sin pass	Se trata de crear un usuario ingresando algunos datos solicitados, todos menos el pass	Ingreso al sistema	Crear el usuario en la BD	email, nombre diagrama de clases	Se permite el ingreso al sistema	Tener email y conexión a internet	Muy satisfactorio	ok	Se creo perfectamente el DC para ese email, ademas se envia por mail el pass de ingreso.
3	Ingreso al sistema con DC ya creado anteriormente	Se pide trabajar con un DC ya creado	Ingreso al sistema	Se llama por medio del email y contraseña a un DC ya creado	email, pass	Se ven las versiones del DC seleccionado o se devuelve mensaje de error si los datos son incorrectos	Conocer el email y el pass que llevan al DC	Muy satisfactorio	ok	Si los datos son incorrectos se redirecciona a la solapa de selección
4	Ingreso al sistema con un usuario sin versiones disponibles	Al ingresar a un DC que aun no se inicio no se ven versiones diponibles	Creacion de DC	Generar DC y versiones	email, pass	Lista de versiones vacias, es necesario crear version nueva	Conocer el email y el pass que llevan al DC	Muy satisfactorio	ok	La lista de versiones se vera vacia y solo tendra la opcion de crear una nueva version del DC
5	Crear nueva version del DC	Se crea una version del DC	Creacion de DC	Generar DC y versiones	Comentarios y nombre de la version, datos no obligatorios	Creacion de esta version	Ninguno	Muy satisfactorio	ok	Se pudo crear la version con o sin lo datos, se le asigna un id y la fecha de creacion
6	Borrar versión del DC	Se borra una version seleccionada	ABM de la BD	Borrar la version de la BD	Luego de visualizar las versiones se elige borrar una.	Se espera que deje de viualizarse esa version de la lista	Ninguno	Muy satisfactorio	ok	Se pudo borrar las versiones
7	Seguir trabajando con una version	Se ingresa a una version seleccionada	ABM de la BD	Se tiene acceso a los datos de la version	Ingreso de nuevas clases	Se visualizan todas las clases que forman a la version.	Ninguno	Muy satisfactorio	ok	En caso que ya existan clases se las puede PERCIBIR, en caso que no existan se PERCIBE un DC vacio
8	Volver a ver versiones del DC	Se quiere volver atrás	Retroceder	Se vuelve al menu anterior	clic en boton atrás	Volver a la interface anterior	Ninguno	Satisfactorio	Se puede mejorar en futuras actualizaciones	El boton de la aplicación funciona perfectamente, pero el boton atrás del browser puede fallar y perder consistencia, para evitar esto se vuelve al menu de inicio para no perder datos
9	Crear clase en una version	Se quiere agregar una clase a una determinada version	ABM de la BD	Se espera modificar la version con la nueva clase, es necesario que se previsualice el cambio	nombre(Opcional y deseable) y tipo de la clase.	Se agrega la clase a la version, la misma se puede percibir.	Generar la clase	MALO	Solcionado al 100%	El campo no obligatorio de nombre no trajo problemas en la BD pero si en el protocolo asi que se cambia ese campo a obligatorio

Tabla 11.a: Casos de prueba y su impacto en el SW.

Id	Caso de Prueba	Descripción	Área Funcional	Funcionalidad	Datos	Resueltado esperado	Requerimientos	Resultado	Estado	Observaciones
10	Percibir cambios en version	Se quiere ver y escuchar los cambios en las versiones	PERCEPCION	Percibir los cambios realizados en la versión.	clic en el boton percibir.	Se perciben los cambio sin problemas.	Audio activo para escuchar el diagrama de clases.	Muy satisfactorio	ok	Se genera sin problemas el audio y la imagen ya sea con una version vacia o completa total o parcialmente.
11	Enviar por mail DC	Se quiere enviar por email el DC	Envio	Enviar por mail el DC	email del receptor, nombre y asunto (opcional)	Recibir mail con los protocolos y el audio	Conexión a internet	Regular	Solucionado al 100%	Si bien no se encontraron errores groseros se encontro un detalle no menor que era que los datos del receptor no eran obligatorios, ahora ese dato y el nombre del emisor son obligatorios.
12	Recibir mail	Se quiere recibir el mail con todos los datos adjuntos	Envio	Luego del envio se espera que el receptor reciba los datos del DC	email del receptor, nombre y asunto (opcional)	Recibir los txt con los textos de protocolos y la descarga del audio.	Cuenta de email	Bueno	ok	Se descubre que mandar el audio como adjunto tiene problemas con algunas casillas de email por cuestiones de seguridad. Asi que se envia un link de descarga del audio y no el adjunto
13	Borrar Clase	Se quiere borrar una clase , sus metodos, atributos y relaciones	ABM de la BD	Borrar clase	clic en la clase a borrar	Debe desaparecer la clase del DC, se edita la imgen y el audio	Ninguno	Muy satisfactorio	ok	
14	Ver listas de la clase	Se quiere ver las listas de atributos, relaciones y metodos dentro de la clase a modificar	ABM de la BD	Ver estructura de la clase	Elegir que se quiere ver	Se deben ver las listas que forman a la clase	Ninguno	Muy satisfactorio	ok	En esta parte de la interface solo se proponen imágenes, sin texto y audio. Pero la pagina web es totalmente accesible.
15	Agregar atributos a la clase	Se quiere agregar un atributo a alguna de las clases	ABM de la BD	Modificar Clase	nombre, tipo, privilegia del atributo	Debe quedar reflejado en la base de datos el cambio en la clase	Completar todos lo campos	Regular	ok - Mejorado	Gracias a esta prueba se encontro una error logico que si bien no arruinaba el sw no era tenido encuenta. Luego de esta prueba se eliminan los espacios en el nombre de los atributos. Se lo cambia por un "_". Este error sirvio para reparar conflictos semejantes en otras funciones.
16	Agregar metodo a la clase	Se quiere agregar un metodo a alguna de las clases	ABM de la BD	Modificar Clase	nombre, tipo, privilegio	Debe quedar reflejado en la base de datos el cambio en la clase	Completar todos lo campos	Muy satisfactorio	ok	
17	Agregar argumento a metodos	Se quiere agregar un argumento a alguno de los metodos	ABM de la BD	Modificar Clase	nombre y tipo	Debe quedar reflejado en la base de datos el cambio en la clase	Completar todos lo campos	Muy satisfactorio	ok	
18	Agregar relaciones a las clases	Se quiere agregar una relacion a alguna de las clases	ABM de la BD	Modificar Clase	inicio, fin, cardinalidad(opcional)	Debe quedar reflejado en la base de datos el cambio en la clase	Completar todos lo campos	Muy satisfactorio	ok	
19	Ir atrás	Se quiere volver atrás en cualquier momento del sw	Consistencia	Volver atrás	clic en boton atrás	Poder volver atrás en todo momento	Ninguno	Regular	Solucionado al 100%	Se notaron imperfecciones en esta actividad, pero se lo reparo.
20	Reproducir audio	Una vez generado el DC se quiere escuchar la "vista previa"	Percepción	Escuchar DC	Clic en botones de interacción de audio	Se debe poder escuchar, pausar, reanudar el audio que	Audio activo.	Bueno	ok	Se noto que responsivevoice inicia el audio desde el principio cuando se cambia de ventana de Browser, no es un error, simplemente

Tabla 11.b: Casos de prueba y su impacto en el SW.

Según el color se puede apreciar el nivel de importancia del caso de prueba: ROJO (Importante), AMARILLO (Regular), VERDE (Leve).

3.10.4 Pruebas de rendimiento

Para realizar estas pruebas verifique el tiempo promedio de descarga de los audios generador por Responsivevoice. Se hicieron las pruebas con diagramas de clases con desde 1 a 10 clases, con 3 conexiones distintas. Claramente el texto y el audio generado con 10 clases era mucho mayor que el generado con una sola clase, por eso surgió la necesidad de saber cuánto se podría demorar en generar la descarga de dicho audio.

A modo de estandarización se optó arbitrariamente por ponerle dos atributos y dos métodos con dos argumentos a cada una de las clases.

Los resultados obtenidos en segundos para la generación del audio de descarga son los siguientes (figura 23):

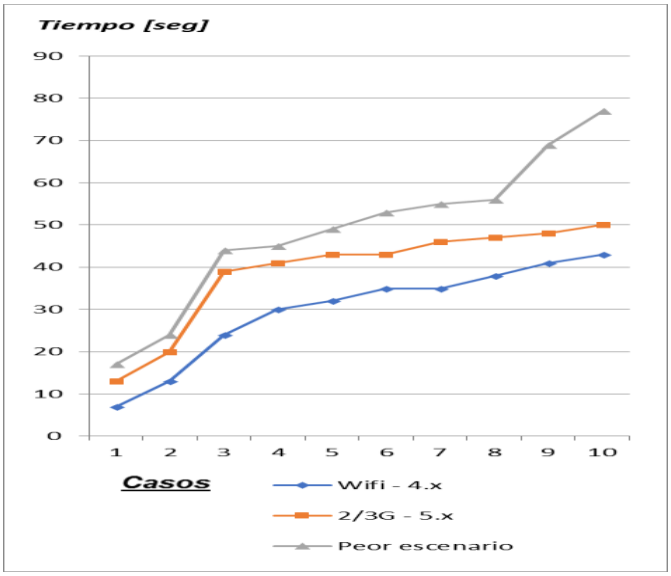


Figura 23: Prueba de rendimiento, para distintas conexiones.

Se aprecia que con una muy mala conexión de internet y un diagrama de clases grande (10 clases) el tiempo de espera de la generación del audio con esta tecnología podría superar al minuto. Pero en promedio y con conexiones menos disminuidas con menos de medio minuto se genera el audio a la perfección para poder manipularlo desde la web.

3.10.5 Validación con el usuario

Este tipo de productos sólo puede validarse con una relación estrecha con los usuarios finales, Hernán ha tenido un feedback positivo en este aspecto ya hace más de dos años, alimentándome a mí como a todos los integrantes de este proyecto con todo lo aprendido. Además, por mi parte, hemos tenido 3 encuentros con Cristian, en primera instancia simplemente le mostré el primer prototipo funcional, mandándole al mail los textos generados y haciéndole escuchar el DC. De este encuentro surgió la necesidad de los botones de reproducción en la página web, además nos observó que el audio no sería lo más importante del SW ya que ellos están acostumbrados a sus audios personalizados, por lo cual usarían mucho más los textos.

Luego de este primer encuentro se agregaron estas cuestiones y se trabajó más con la estructuración de los textos. En el segundo encuentro, se aceptó la redacción modificada de los textos, pero se decidió hacer unos cambios en el protocolo, evitando el uso de los corchetes, ya que, si bien a él en particular no le resultaba complejo utilizarlos, nos planteó que el uso de esos caracteres puede ser engorroso. Además, se aceptaron las sugerencias para la navegación, simplificando un poco las pantallas.

En el tercer encuentro, se aceptó el protocolo y el usuario volvió a escuchar los textos con su audio, no se realizaron observaciones.

Con esto tres encuentros se dio por terminada la validación con el usuario final, si bien somos conscientes que el uso operativo, repetitivo y sobre todo el uso por muchos usuarios nos seguirá dando posibles cambios futuros.

3.11 PROCESO DE GESTIÓN DE LA CONFIGURACIÓN

El objetivo de la gestión de la configuración es mantener la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas de información, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan. Así, entre los elementos de configuración software, se encuentran no únicamente ejecutables y código fuente, sino también los modelos de datos,

modelos de procesos, especificaciones de requisitos, pruebas, etc. La gestión de configuración se realiza durante todas las actividades asociadas al desarrollo del sistema, y continúa registrando los cambios hasta que éste deja de utilizarse. La gestión de configuración facilita el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio, tanto evolutivo como correctivo. Asimismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización.

3.11.1 Realizar el control de la configuración

Cualquier interesado involucrado en el proyecto puede solicitar cambios, ya sea Cristian o cualquiera de los usuarios que quieran sacarle provecho a este aplicativo. Aunque los cambios pueden iniciarse verbalmente, siempre deben registrarse por escrito e ingresarse al sistema de gestión de cambios y/o al sistema de gestión de la configuración, se tendrá acceso a estos pedidos desde la propia aplicación, para ver cómo acceder es necesario ver el manual del usuario.

Siempre que se requiera, el proceso Realizar el Control Integrado de Cambios incluirá un comité de control de cambios (CCB) que será responsable de aprobar o rechazar las solicitudes de cambio, este comité puede estar compuesto por el Creador (en este caso el autor de esta tesis), el Cliente y los miembros más involucrados en el proyecto.

Por ahora sólo se realizaron los siguientes cambios:

-En cada menú hay una ayuda auditiva.

-Se cambio el tipo de letra en el modo Escribir Diagrama de Clases, letra más pequeña.

-Se incorporó un mensaje de contacto para recibir comentario y/o críticas.

-El link de descarga se envía en un txt para evitar la protección de algunos de los servicios de mail que bloquean él envió, como Hotmail.

3.11.2 Realizar la información del estado de la configuración

Los cambios en la configuración y el versionado junto con las descargas respectivas de la documentación y software están en:

Link: https://1drv.ms/f/s!AgB0dw0E7wKakII1tcBvvU6Bk3tcLA

3.12 PROCESO DE DESARROLLO DE DOCUMENTACIÓN

Aquí tendremos disponible todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del proyecto, pasando por modelaciones (UML), diagramas, pruebas, manuales de usuario, manuales técnicos, etc.; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.

3.12.1 Planificar la documentación

Se realizarán los siguientes documentos:

- Diagrama de sistemas
- Diagrama de contexto
- DER
- Casos de Uso
- Especificación de casos de uso
- Especificación de casos de prueba
-

3.12.2 Implementar la documentación. Producir y distribuir la documentación

Toda la documentación pertinente a este software puede descargarse desde el enlace que se dejara a continuación, además se puede acceder a los últimos documentos desde los enlaces de descarga disponibles en la web de Percepción UML.

Link: https://1drv.ms/f/s!AgB0dw0E7wKakIIU682sdWEyghwxHQ

3.13 PROCESO DE FORMACIÓN

En esta etapa se lleva a cabo la planificación, desarrollo, validación e implementación de los programas de formación de desarrolladores, personal de soporte técnico y clientes y la elaboración de los materiales de formación adecuados.

Aquí, desde la presentación de este trabajo final de licenciatura en adelante se tratará de formar a todos los interesaros en utilizar este SW, así como también a lo que deseen mejorarlo o implementarlo en nuevas tecnologías con los mismos criterios y fines en común.

Como ya se mencionó en otros apartados, este SW va dirigido principalmente a dos grandes usuarios:

- Docentes: Los cuales utilicen la aplicación para generar DC en formato de texto y audio para poder compartirlos con los alumnos que lo necesiten.
- Estudiantes: Que por algún tipo de imposibilidad visual necesiten audio o texto para poder leer un DC.

Luego, desde ya, surgirán destinatarios mixtos, por ejemplo, docentes que requieran audios y textos, estudiantes que quieran generar los protocolos para compartir con docentes u otros compañeros. Además de en algún ámbito laboral que pueda necesitarse. Y esperamos que en muchos otros entornos a futuro.

Entonces, inicialmente se necesita formar a los docentes para la utilización de la web visual, para que puedan enviar estos protocolos de forma automática. Y en segunda instancia se necesita que Cristian o cualquier otro usuario aprenda los protocolos para diagramar sus DC de forma visual para los docentes.

4. CONCLUSIONES

En este Capítulo se presentan las conclusiones del creador y se destacan las futuras líneas de trabajo que se consideran pertinentes para la completitud de este proyecto.

4.1 CONCLUSIONES DEL CREADOR

Recordemos brevemente lo que se pretendía con este proyecto. Surgió la necesidad de brindarle a alumnos con discapacidades visuales la oportunidad de poder estudiar y trabajar en sistemas a la par de alguien que no posea disminuciones visuales. Entonces, se planteo el problema que en la Licenciatura en sistemas prácticamente todo está basado en diagramas que modelan lo que luego se transformara en codificación para solucionar problemas puntuales.

¿Cómo trabajar estos diagramas si son totalmente basados en la estética visual? Aquí aparecemos nosotros, todos los que trabajamos en PERCEPCION UML y los proyectos ligados. Por medio de este trabajo en equipo pretendemos trabajar estos diagramas visuales en protocolos de texto que puedan generalizar cada elemento de las imágenes para que puedan ser comprendidas en su totalidad por los usuarios ciegos; además este conjunto de herramientas y futuras herramientas quiere generar estándares de texto, protocolos, que por medio de un simple tipeo se pueda transformar en imágenes de diagrama ya estandarizados en sistemas.

En este trabajo se ha planteado una pequeña solución bidireccional, el nuevo SW permite compartir un DC a una persona no vidente por medio de un archivo de texto con un protocolo de fácil entendimiento y de una muy simple lectura por medio de los reproductores de texto a voz utilizados por los usuarios finales.

También permite que los usuarios con capacidades visuales disminuidas puedan crear una imagen del DC utilizando una sintaxis muy simple y semejante al español, para poder darle a cualquier programador una imagen con los formatos establecidos por UML.

Se consiguió una herramienta bastante ágil a la hora de manipular desde un navegador, permite mantener un trabajo colaborativo y se logró accesibilidad a la hora del ingreso de un protocolo que genera una imagen para enviar por e-mail.

Si bien esta nueva tecnología agrupa muchas otras ya existentes, ha logrado dar un importante puntapié inicial para un proyecto mucho más grande que facilitará mucho el hospedaje en la carrera de Sistemas a personas con capacidades visuales disminuidas. Y no sólo en la carrera, sino también en la vida laboral.

Se lograron los primeros requisitos planteados por este proyecto, e incluso algunos más, estaremos abiertos a nuevas modificaciones y mejoras que planteen todos los usuarios y esperamos que sea el primero de muchos otros proyectos.

Como bien se detalla en este documento, el SW tiene dos grandes direcciones, la primera, orientada a docentes y estudiantes, los cuales por medio de una interfaz sencilla, con el uso del mouse, opciones múltiples y escribiendo muy poco pueden generar un DC desentendiéndose por completo del diseño, enfocándose únicamente en la estructura interna de dicho DC, generarán una imagen con los parámetros establecidos por UML, además se generará de forma automática un texto en un español coloquial de muy fácil entendimiento para estudiantes no muy avanzados, brindará también un audio que describe detalladamente la imagen del DC y por último generará un protocolo simple para crear el camino inverso; estos 4 formatos para la transferencia del DC podrán ser enviados por mail a cualquier usuario, en particular Cristian o cualquier persona con restricciones en su visión.

Luego, la segunda arista, trató de recrear el camino inverso, por medio de la escritura de un protocolo muy simple, utilizando únicamente el teclado y atajos de comandos, en muy pocos pasos se puede crear una imagen del DC – UML para enviar por mail. Esta segunda arista es la que esperamos sacarle más provecho y mejorar con el correr del tiempo en futuros proyectos.

Las primeras pruebas con docentes, en la arista número uno han sido muy prometedoras, donde se notó que el SW podrá ser de gran utilidad para estandarizar la forma de transmisión de estos diagramas.

Mientras que las pruebas de la 2da arista también han sido buenas, con la salvedad que sólo ha sido usada en forma vertiginosa por docentes sin problemas en la visión, con Cristian solo se han realizado algunas pruebas también prometedoras, pero esperamos realizar muchas más, una vez que la aplicación este disponible las 24hs en un servidor dedicado a este servicio.

¿Esto ha terminado entonces? Pues claro que no, es solo el inicio de muchos otros proyectos. Se pretende generar de forma similar muchos otros diagramas de vital importancia para la carrera de Sistemas, se trabajará en futuros trabajos finales en la creación de protocolos, técnicas, textos y

audios para la creación de DER, DFD, DCU y una web integral que permita acceder a todas estas funcionalidades.

Los DER y DCU son temas que ya fueron expuestos y serán atacados en los próximos trabajos finales de Licenciatura. Esperemos el mismo destino también para la creación de DFD.

También uno de los principales problemas o ítems a diseñar que creemos totalmente necesarios son la creación de índices. ¿Qué es esto?, nosotros, los que nos apoyamos en la visión para la lectura, ¿Qué hacemos cuando necesitamos buscar algo?, básicamente hacemos una lectura global o casi un "pantallazo" para tratar de encontrar lo que estamos buscando, algo así como cuando buscamos una palabra en el diccionario, la vamos encontrando por aproximación.

¿Pero cómo puede hacer algo así alguien que no lee por medio de sus ojos?, índices o atajos auditivos que permitan encontrar lo que se busca sin la necesidad de tener que escuchar todo el audio completo. Esta será la otra línea de investigación que esperamos tener en las próximas presentaciones.

5. REFERENCIAS

Bisquerra Alzina, R. (1989); CEAC*; Métodos de investigación educativa*.

Booch, Grady (2003); Rumbaugh, James; Jacobson, Ivar; Saez Martinez, Jose; Trad.; Garcia Molina, Jesus J; Rev. Tec. *Lenguaje unificado de modelado*. Addison Wesley.

Böehm, B. *Software Engineering***.** IEEE Trans. Computers C-25,12. Dec. 1976.

Böehm, B. W. *A Spiral Model of Software Development and Enhancement***.** ACM Software Engineering Notes 11, 4. 1986.

Böehm, B. W. *A Spiral Model of Software Development and Enhancement***.** Computer, May 1988, pp. 61-72.

Böehm. B. "Software Engineering-R&D Trends and Delense Needs". In Research Dircclions in Software Technolog (p. Wegner, ed.), MIT press. 1979, pp. 44-86.

Budde, R., K. Kuhlenkamp, L. Mathiassen, and H. Zullighoven. *Approaches to Prototyping***.** Springer-Verlag, New York 1984.

Claudio Segovia (2013). *Accesibilidad e internet*. Guatemala, Universidad Galileo. Recuperado de: https://es.slideshare.net/lili369/notas-sobre-la-obra-accesibilidad-e-internet-de-claudio-segovia

Daniel Torres Burriel (2012). Siete años de experiencia de usuario. Recuperado de http://www.torresburriel.com/weblog/2012/05/16/libro-siete-anos-de-experiencia-de-usuario/

Deitel, Harvey. Deitel Paul(2004): *Como programar en JAVA*. Quinta Edición, Pearson educación

Humphrey, W. S. *A discipline for Software Engineering***.** Addison-Wesley. Readings, Massachusetts, EE. UU. 1995.

IEEE Std. 1074-1989. *IEEE Standard Software Life Cycle Processes*. 1989.

Licesio J. Rodríguez-Aragón (2013*). Internet y Teleinformática*. Recuperado de: https://previa.uclm.es/profesorado/licesio/Docencia/IB/IBTema4.pdf

Lic. Gabriela A. Toledo (2011, p.10.). *Manual de prácticas de accesibilidad digital*.

Luis Joyanes Aguilar, Ignacio Zahonero Martìnez, Programación en Java 2: Algoritmos, Estructuras de Datos y Programación Orientada a Objetos, Madrid, McGraw Hill Interamericana.

McCracken, D.D., and Jackson, M.A. *Life-Cycle Concept Considered Harmful***.** ACM SW Eng. Nates, Apr. 1982, pp. 29-32.

Opera (2006). Opera Web Browser [en línea] Opera Software. [Consulta: 23-10-2006].

Pressman, R. S. *Ingeniería del Software: Un enfoque práctico***.** 3 ed. McGraw-Hill, Madrid. 1993.

Primera biblioteca digital para ciegos (2016). Argentina, Buenos Aires. Recuperado de: http://www.tiflolibros.com.ar/

Sidar (2003). Herramienta Hera Versión 1.0 [en línea]. Fundación Sidar – Acceso Universal [Consulta: 24- 10-2006].

Sidar (2005) Herramienta Hera Versión 2.0 beta [en línea]. Fundación Sidar – Acceso Universal [Consulta: 23-10- 2006].

Anexo:
Manual del usuario
PERCEPCIÓN UML

Estudiante

A.P.U. PEREZ, Nicolás Ignacio

Director

Mg. AMATRIAIN, Hernán

TRABAJO INTEGRADOR PRESENTADO PARA OBTENER EL TÍTULO
DE
LICENCIADO EN SISTEMAS

DEPARTAMENTO
DE DESARROLLO PRODUCTIVO Y TECNOLOGICO
UNIVERSIDAD NACIONAL DE LANUS

MARZO,2019

1- Modo online - Visual

a- Inicio y primeros pasos.

Iniciar la actividad en Percepción UML es muy simple, tan solo hay que ingresar a la pagina oficial:

Link: http://node24730-uml-dc.jelastic.saveincloud.net/UML-DC/

Cabe destacar que el servidor puede ir migrando por lo cual el link puede llegar a cambiar, para evitar que los usuarios no estén al tanto de estos cambios, todos tienen acceso a las redes sociales, foros de preguntas y soporte online en las siguientes plataformas de interacción social:

Facebook: https://www.facebook.com/groups/546266565848437

Google: https://plus.google.com/102793582451064465757

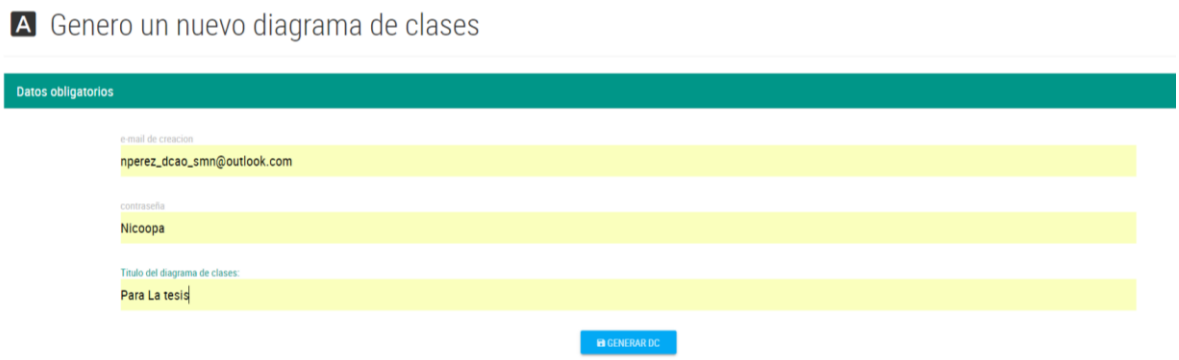
Instagram: https://www.instagram.com/percepcionuml/

En dichas redes sociales se publicarán soluciones a problemas, ejemplos y en caso de ser necesario los nuevos links para acceder a la aplicación.

b- Crear Login y DC

Si vas a crear un Diagrama de Clases desde cero, debes indicar con que mail lo vas a crear, generar una contraseña y darle un nombre al diagrama. Para conseguir esto hay que ir a la segunda opción de la pantalla inicial o en su defecto usar el método abreviado ALT + 2.

Una vez clickeado en CREAR UN NUEVO DIAGRAMA, tendremos que llenar el formulario:

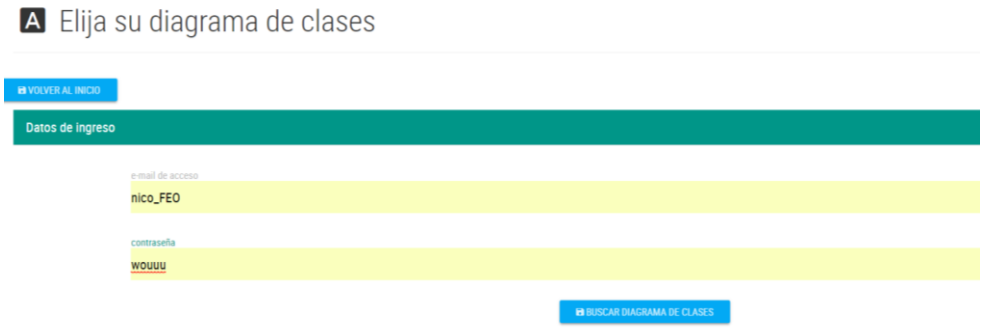


El único campo no obligatorio es el de contraseña, en caso de no poder una contraseña el sistema le asignara una que se enviara a su email para que no la olvide.

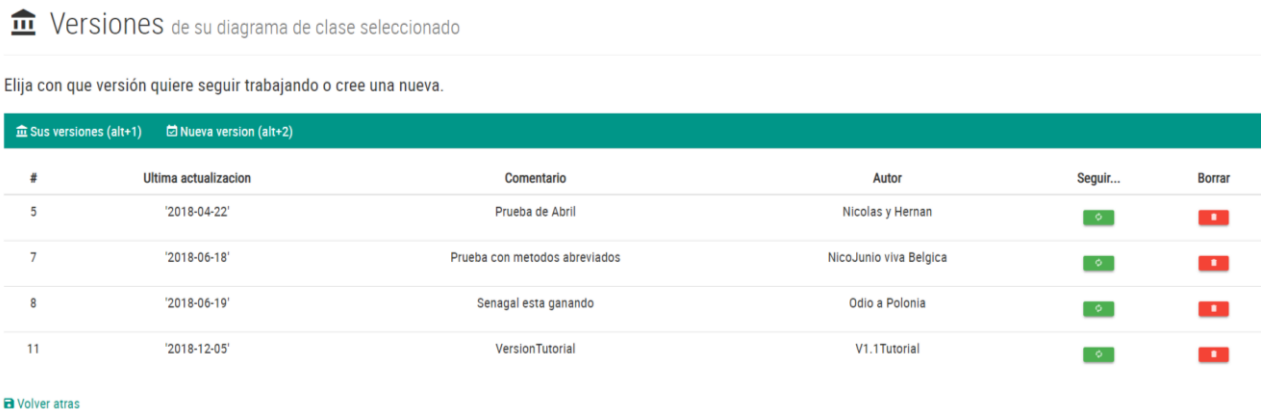
c- Crear DC con un Login ya existente

Una vez que ya se realizo el paso b, ya podemos trabajar con uno o varios diagramas de clases. Para ello vamos a la primera opción, ABRIR DC EXISTENTE, o en su defecto apretar ALT + 1.

Esto nos pedirá ingresar el mail de ingreso al DC y la contraseña.



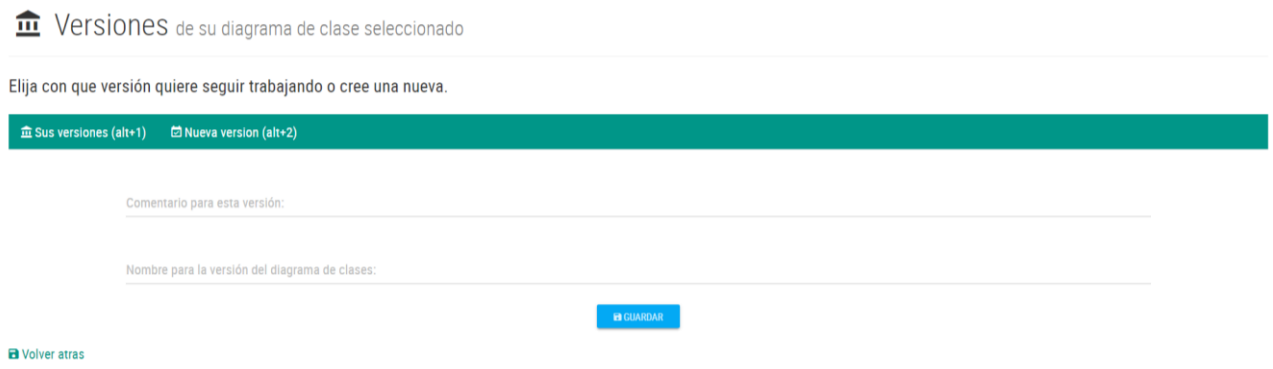
Si el Login es correcto aparecerá una pantalla con todos los DC que se crearon con esa cuenta:



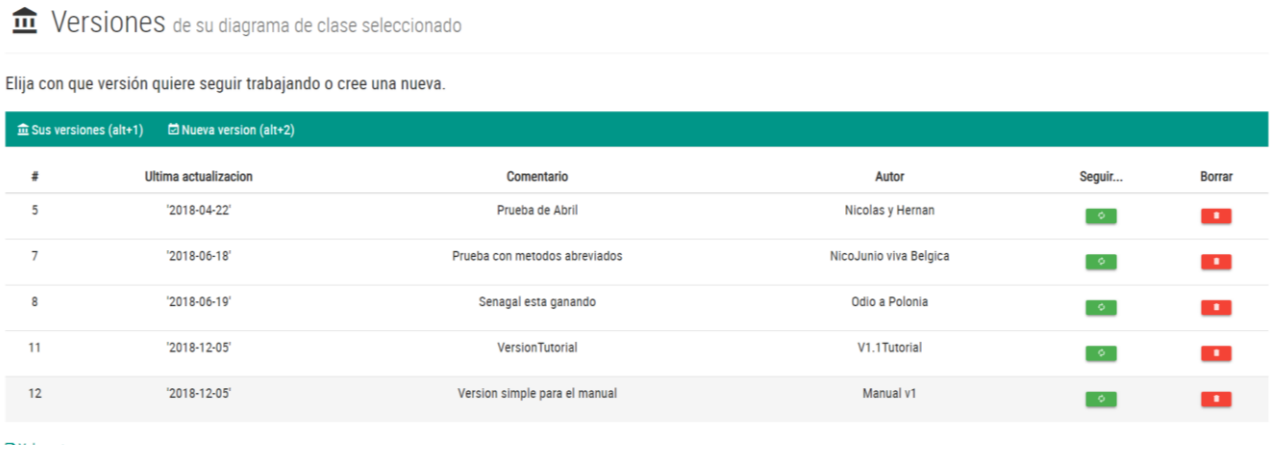
Es normal que la primera vez no tengamos nada en esta pantalla, por eso es necesario pasar al siguiente ítem de este manual.

d- Primera versión de un DC

Una vez que estamos en la pantalla versiones, puede que no tengamos ninguna y queramos generarla. Para esto ir a la solapa NUEVA, o en su defecto ALT +2.



Esto nos pedirá un comentario descriptivo para el DC y un nombre único para este DC. Una vez llenos estos campos se presiona Generar.



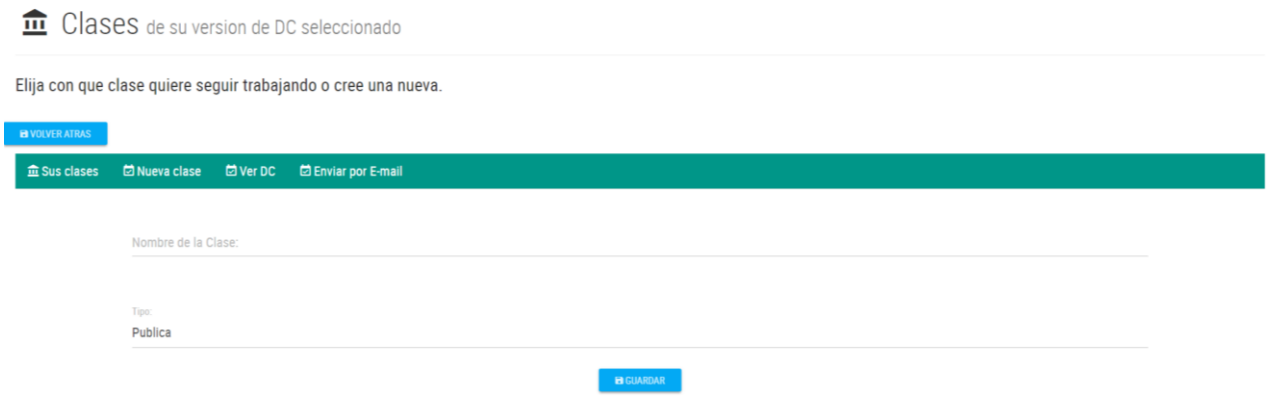
Ya tenemos disponible nuestra primera versión del DC.

e- Trabajar sobre una versión de un DC.

Una vez que ya tenemos por lo menos una versión del DC podemos ir trabajando con esta versión. Presionar SEGUIR para trabajar con esta versión, en su defecto si se quiere eliminar por completo esta versión apretar BORRAR.

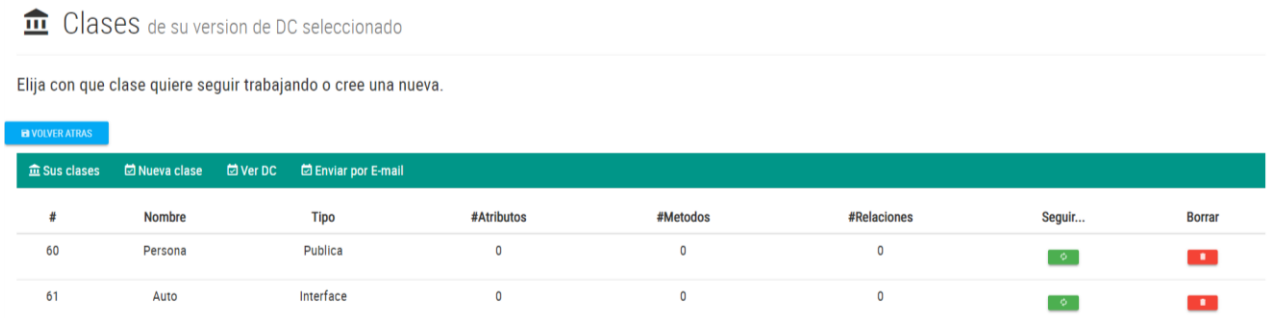
Al ir a SEGUIR, aparece una pantalla con todas las clases de este diagrama.
Es normal que inicialmente no tengamos ninguna clase. Las opciones de este menú son VER CLASES, NUEVA CLASE, VER DC, ENVIAR.

Como no tenemos ninguna clase, crearemos algunas. Para esto vamos a la solapa NUEVA CLASE.

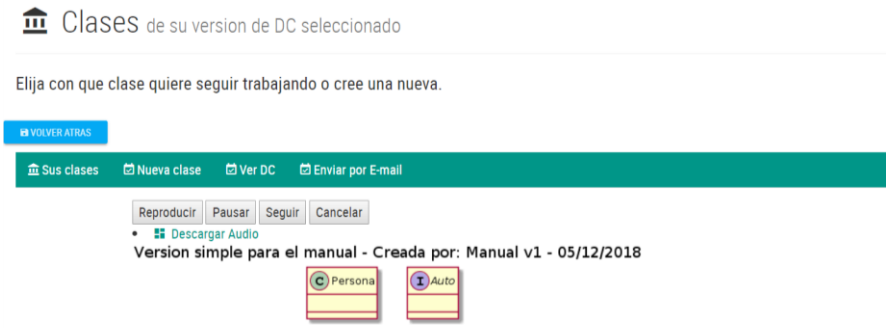


Esto nos pedira, el nombre de la clase y con un menu desplegable el tipo de clase.
Crearemos a modo de práctica, la clase Publica Persona, y la clase Interface Auto.

Al terminar se verá en el menú nuestras dos clases, con contadores que indican cuantos atributos, métodos y relaciones tiene cada una de estas clases:



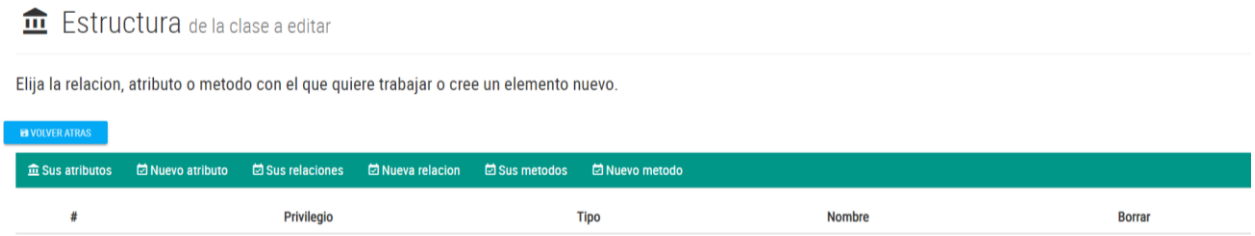
Ya podemos “PERCIBIR” este diagrama de clase, para esto vamos a la solapa VER DC.



No solo vemos el DC que fuimos generando, si no también podremos escucharlo, clickeando en los botones de reproducción. También se puede descargar un mp3 con el audio del DC.

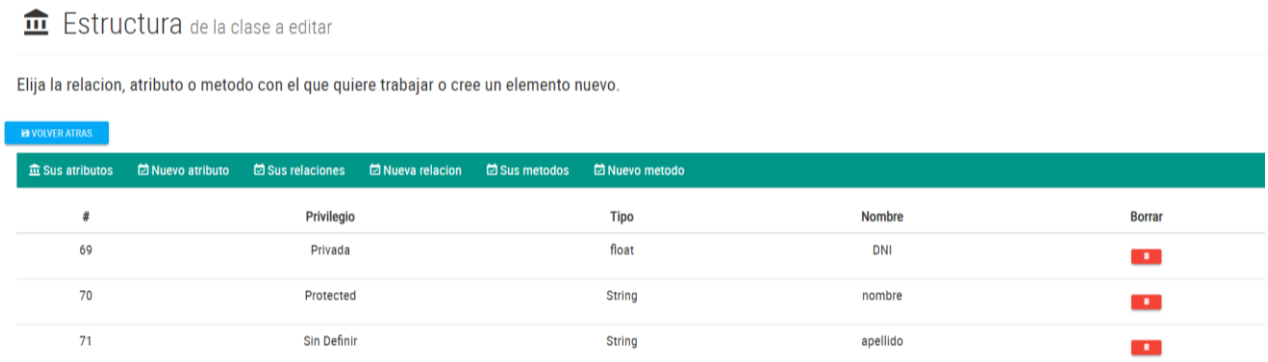
f- Trabajar sobre una clase.

Si volvemos a SUS CLASES, y trabajamos con una de las ya creada, por ejemplo, PERSONA, por medio del botón SEGUIR, se despliega un menú así:



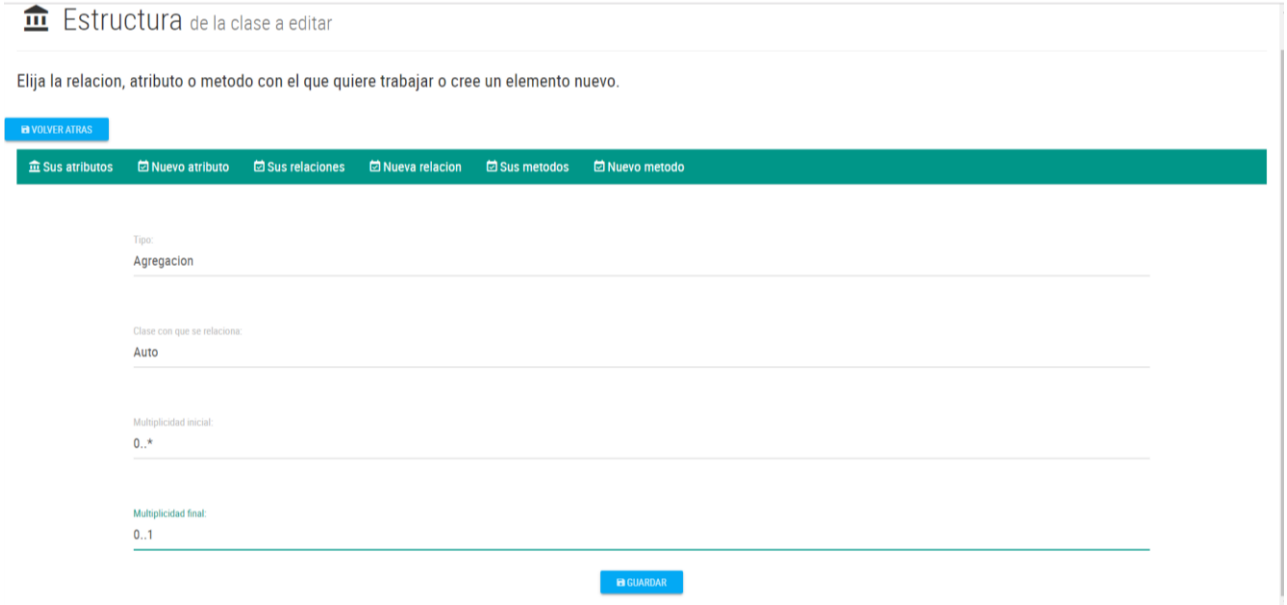
Aquí se verán los atributos, métodos y relaciones de la clase. Además podremos crear nuevos desde esta misma pantalla, por ejemplo, creemos 3 atributos nuevos. DNI, NOMBRE y APELLIDO. Para esto vamos a NUEVO ATRIBUTO.

Al generarlos, nuestra pantalla quedara algo así:

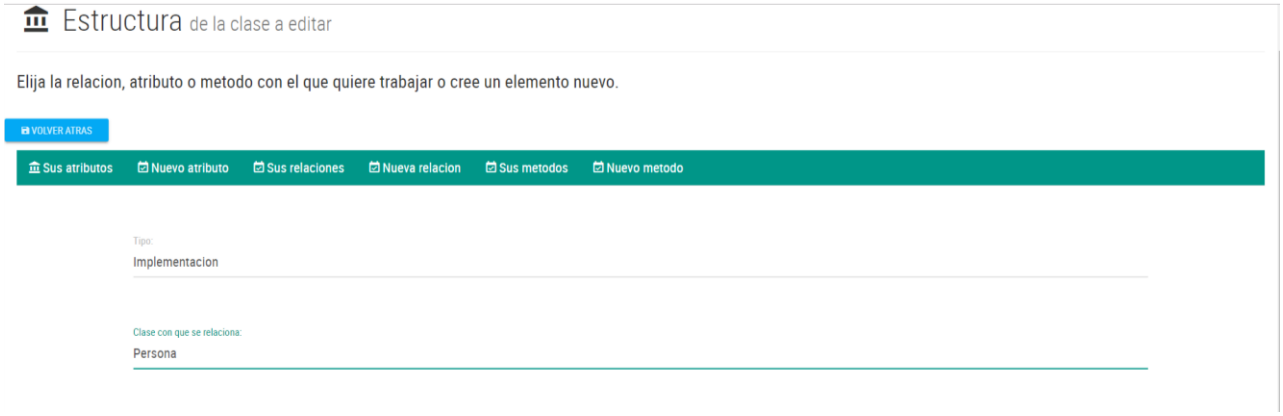


Ya tenemos atributos, podemos crear relaciones ahora, por ejemplo, entre Persona y Auto, para esto vamos a NUEVA RELACION.

Elegimos el tipo de relación, con que clase es la relación, y la multiplicidad de la relación, de ser necesario. Por ejemplo:

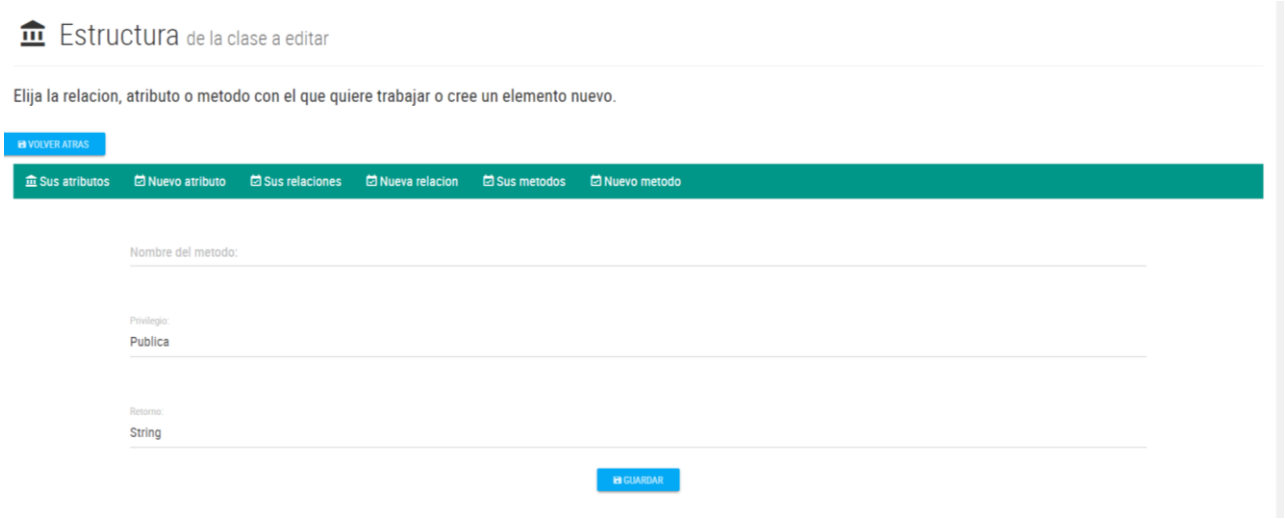


Al apretar generar, ya podemos ver la relación yendo a SUS RELACIONES. Crearemos otra más, entre PERSONA Y PERSONA.

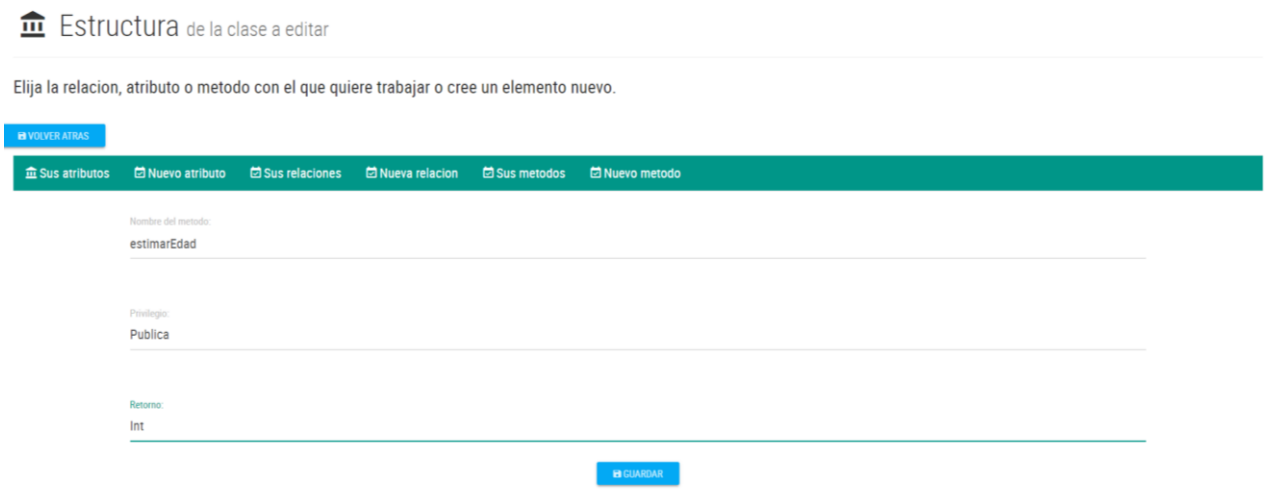


La visualización de ambas sería algo así:

Por último, nos faltaría poder agregar métodos, para esto vamos a NUEVO MÉTODO. Tendremos que elegir el nombre del método, el privilegio y el retorno de este.

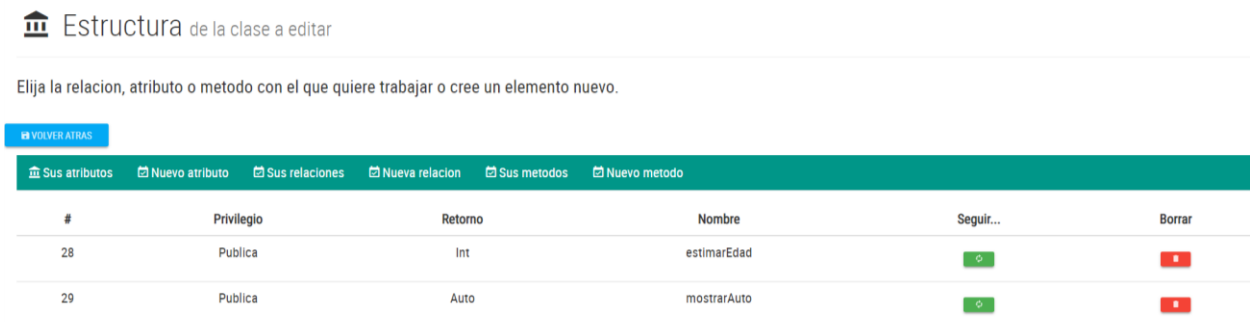


Crearemos dos métodos, estimaEdad, que devuelve un entero, del tipo público.

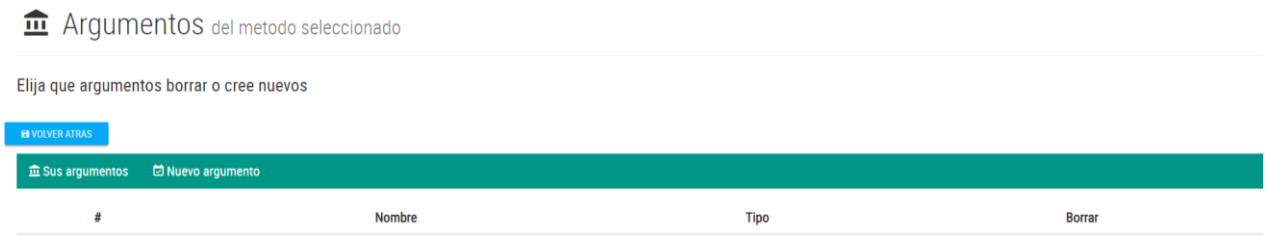


Ahora creemos uno que tenga argumentos, por ejemplo, un método público, mostrarAuto, que retorna un void, pero tiene como argumento un Auto.

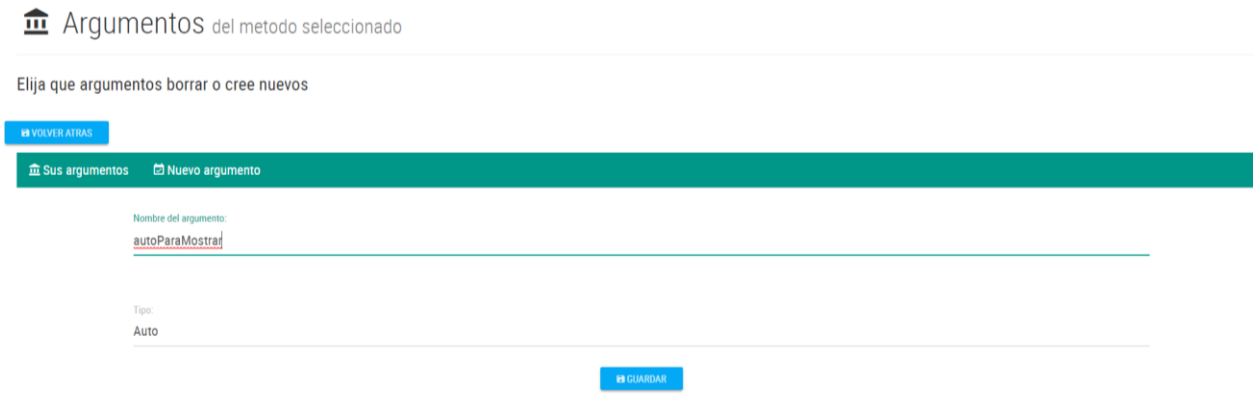
Esta claro, que hicimos lo mismo en los dos últimos ejemplos, pero ¿como hacemos para agregar argumento?, vamos a SUS METODOS, veremos algo así:



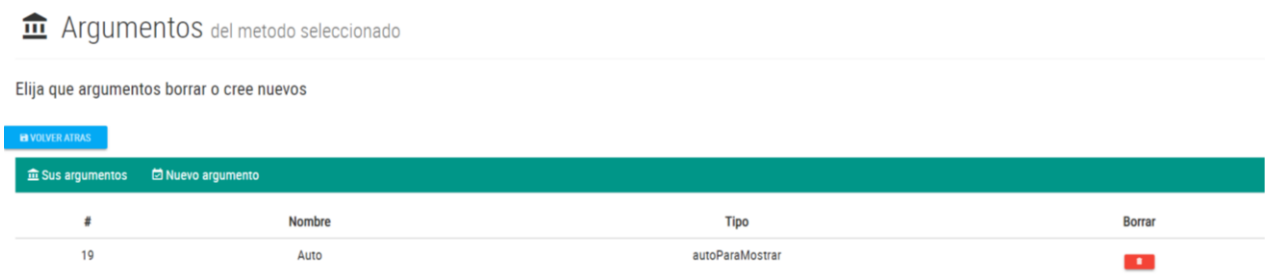
Aquí tenemos compactos todos los métodos, si queremos agregarle argumentos a alguno de estos métodos, vamos a SEGUIR, por ejemplo, con mostrarAuto:



En esa pantalla se verán los argumentos, que naturalmente no hay ninguno, para crear uno vamos a NUEVO ARGUMENTO. Y completamos a gusto los valores.



Al ir a SUS ARGUMENTOS, ahora veremos algo así:



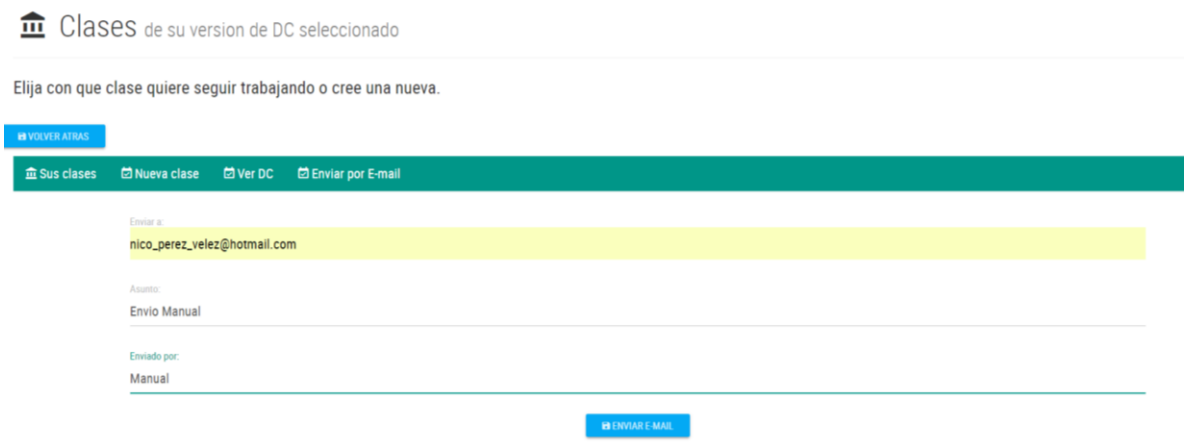
Ya vimos casi todas las opciones de PERCEPCION UML, vamos a VOLVER, dos veces, a ver como quedo nuestra clase.

Los contadores reflejan todo lo que fuimos agregando a nuestra clase.
Con la clase Auto podemos realizar los mismos pasos, pero antes de seguir, pueden ir a Ver DC.

Aquí se aprecia el primer gran logro de PERCEPCION UML, vemos la imagen, podremos escuchar el DC y enviarlo por mail.

g- Enviar DC

Ya lo vimos y lo escuchamos, vamos a compartirlo. Para esto vamos a ENVIAR POR E-MAIL. Completamos el asunto, el remitente y destino, para poder enviarlo.



Si todo salió bien se volverá a la pantalla de inicio sin ningún mensaje de error.

h- Recepción de email.

Luego de la opción g, alguien va a recibir, 3 archivos, un archivo con un protocolo para generar la imagen, un texto explicativo del DC y un audio, el mismo audio que podíamos escuchar desde la pagina web.

En nuestro caso se recibirá el siguiente texto explicativo:

…"Diagrama de clases, creado por Manual v1 el dia 05/12/2018 esta compuesto por:

Clase publica denominada Persona, donde sus atributos son:
Un atributo privado del tipo float denominado DNI.
Un atributo protegido del tipo String denominado nombre.
Un atributo sin definir del tipo String denominado apellido.
Un Metodo publico con nombre estimarEdad, este metodo retorna un Int.
Un Metodo publico con nombre mostrarAuto. Con un argumento del tipo Auto con nombre autoParaMostrar, este metodo retorna un Auto.

Interface denominada Auto, sin Atributos definidos.

El Diagrama de clase posee las siguientes relaciones entre las clases:

La clase Persona con una relacion de agregacion con la clase Auto.Ademas esta relacion va desde cero a muchos hasta cero a uno.
La clase Persona con una relacion de implementacion con la clase Persona.
La clase Auto no tiene relaciones a nombrar."…

2- Modo online – Teclado

a- Inicio de página web.

Iniciar la actividad en Percepción UML es muy simple, tan solo hay que ingresar a la pagina oficial:

Link: http://node24730-uml-dc.jelastic.saveincloud.net/UML-DC/

Cabe destacar que el servidor puede ir migrando por lo cual el link puede llegar a cambiar, para evitar que los usuarios no estén al tanto de estos cambios, todos tienen acceso a las redes sociales, foros de preguntas y soporte online en las siguientes plataformas de interacción social:

Facebook: https://www.facebook.com/groups/546266565848437

Google: https://plus.google.com/102793582451064465757

Instagram: https://www.instagram.com/percepcionuml/

En dichas redes sociales se publicarán soluciones a problemas, ejemplos y en caso de ser necesario los nuevos links para acceder a la aplicación.

b- Dibujar por medio del protocolo

En la pantalla inicial de Percepcion UML, ir a la 3er opción, por medio de ALT + 3.

c- Escribir protocolo

En siguiente pantalla nos permite escribir un protocolo para dibujar un DC. Apretando TAB una vez ya podemos escribir en la pantalla.

d- Estructura del protocolo

El protocolo ser escrito así:

```
Titulo este es el tirulo del DC
Clase NombreClase1InicioClase
-tipo1 atributo1
+tipo2 atributo2
#tipo3 atributo3
tipo4 atributo4
-metodo1(tipo1 argumento1, tipo2 argumento2): retorno1
+metodo2(tipo3 argumento3, tipo4 argumento2): retorno2
FinClase
Clase NombreClase2InicioClase
-tipo nombre
+método(): retorno
FinClase
```

NombreClase1 ComposicionDesde NombreClase2
NombreClase1 ComposicionHacia NombreClase2
NombreClase1 AgregacionDesde NombreClase2
NombreClase1 AgregacionHacia NombreClase2
NombreClase1 UnoAMuchos ComposicionDesde CeroAMuchos NombreClase2
NombreClase1 CeroAUno ComposicionHacia NombreClase2
NombreClase1 HeredaDe NombreClase2
NombreClase1 HeredaA NombreClase2

Todas las combinaciones de comandos y ejemplos pueden descargarse de:

Link: https://1drv.ms/f/s!AgB0dw0E7wKakLFSIKSP6eW_2eNnAQ

e- Envio de nuestro protocolo

Una vez escrito el protocolo, apretar TAB y enter para pasar al menú siguiente.

Podemos seguir modificando el texto de quererlo así, o ir enviando cada cambio. Para enviarlos presionar TAB una vez para GENERAR LA IMAGEN, luego enter y TAB para llegar a la opción enviar por mail, presionar ENTER y nos llevara al formulario de envió por mail.

Completar los datos del destinatario, esto enviara por mail una imagen que ilustrara nuestro protocolo.