



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

SurveyingPointCode

Automatización del proceso de
delineación a partir de datos de
un Levantamiento Topográfico.



Presentado por
José Eduardo Risco Sánchez-Cortés
en Universidad de Burgos — 5 de junio de 2019
Tutores: Dr. César Ignacio García-Osorio
y Dr. Carlos López Nozal

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	10
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	17
B.3. Catalogo de requisitos	18
B.4. Especificación de requisitos	19
Apéndice C Especificación de diseño	35
C.1. Introducción	35
C.2. Diseño de datos	35
C.3. Diseño arquitectónico	37
C.4. Diseño procedimental	39
Apéndice D Documentación técnica de programación	43
D.1. Introducción	43
D.2. Estructura de directorios	43
D.3. Manual del programador	44
D.4. Compilación, instalación y ejecución del proyecto	47
D.5. Pruebas del sistema	47

Apéndice E Documentación de usuario	55
E.1. Introducción	55
E.2. Requisitos de usuarios	55
E.3. Instalación	56
E.4. Manual del usuario	56
E.5. Preguntas Frecuentes	71
Bibliografía	73

Índice de figuras

A.1. Diagrama del ciclo iterativo <i>Scrum</i>	1
A.2. Gestión de tareas con <i>ZenHub</i> en una fase del proyecto.	2
A.3. Logo de <i>SurveyingPointCode</i>	3
A.4. Logo de <i>Dibujo con líneas y splines</i>	4
A.5. Compatibilidad entre licencias	14
A.6. Mensaje de aviso de <i>Cookies</i>	15
B.1. Diagrama de general de casos de uso.	20
B.2. Diagrama de caso de uso CU-01	21
B.3. Diagrama de caso de uso CU-02.	22
B.4. Diagrama de caso de uso CU-03.	23
B.5. Diagrama de caso de uso CU-04	24
B.6. Diagrama de caso de uso CU-05.	26
B.7. Diagrama de caso de uso CU-06.	27
B.8. Diagrama de caso de uso CU-07	28
B.9. Diagrama de caso de uso CU-08.	29
B.10. Diagrama de caso de uso CU-09.	30
B.11. Diagrama de caso de uso CU-10.	31
B.12. Diagrama de caso de uso CU-11.	32
B.13. Diagrama de caso de uso CU-12	33
B.14. Diagrama de caso de uso CU-13	34
C.1. Diagrama relacional principal.	36
C.2. Diagrama relacional elementos del dibujo.	37
C.3. Esquema de un patrón MTV.	37
C.4. Diseño arquitectónico con ORM y BBDD.	38
C.5. Arquitectura de <i>Docker</i>	38
C.6. Arquitectura final de la aplicación con <i>Docker</i>	39
C.7. Diagrama de flujo de la aplicación	40
C.8. Diagrama de secuencia general	41

E.1. Mensaje de información <i>cookies</i>	56
E.2. Pantalla de bienvenida.	57
E.3. Registro de usuario.	58
E.4. Registro con éxito.	58
E.5. <i>Login</i> de usuario.	59
E.6. Carga de archivos.	60
E.7. Error formato de archivo topográfico, <i>parser</i>	62
E.8. Error archivo topográfico vacío.	62
E.9. Error archivo topográfico, dibujo de cuadrados.	62
E.10. Error archivo topográfico, dibujo de rectángulos.	63
E.11. Error formato de archivo configuración de la conversión, <i>parser</i> . . .	64
E.12. Error archivo configuración de la conversión vacío.	64
E.13. Error archivo configuración de la conversión, códigos repetidos. . .	65
E.14. Errores corregibles a través de la interfaz.	65
E.15. Error archivo de símbolos, no contiene bloques.	66
E.16. Elección de colores.	67
E.17. Elección de símbolos.	67
E.18. Elección de versión de CAD.	68
E.19. Conversión con éxito.	68
E.20. Errores corregibles a través de la interfaz.	69
E.21. Descarga de archivos.	69
E.22. <i>Logout</i>	70
E.23. <i>Cierre de sesión por inactividad</i>	70

Índice de tablas

A.1. Costes de personal.	10
A.2. Costes de <i>hardware y software</i>	11
A.3. Costes varios.	11
A.4. Costes totales.	11
A.5. Tipos de suscripciones y cuotas	12
A.6. Dependencias del proyecto.	13
B.1. CU-01 Registro de usuarios.	21
B.2. CU-02 <i>Login</i> de usuarios.	22
B.3. CU-03 Carga de archivo de campo.	23
B.4. CU-04 Carga de archivo de configuración.	25
B.5. CU-05 Carga de archivo de símbolos.	26
B.6. CU-06 Asociar capas y colores.	27
B.7. CU-07 Asociar símbolos.	28
B.8. CU-08 Elegir versión de CAD.	29
B.9. CU-09 Dar nombre al archivo DXF generado.	30
B.10. CU-10 Conversión a DXF.	31
B.11. CU-11 Descargar archivos.	32
B.12. CU-12 <i>Logout</i>	33
B.13. CU-13 Limpiar archivos.	34
D.1. Caso de prueba CP-01.	48
D.2. Caso de prueba CP-02.	48
D.3. Caso de prueba CP-03.	48
D.4. Caso de prueba CP-04.	49
D.5. Caso de prueba CP-05.	49
D.6. Caso de prueba CP-06.	49
D.7. Caso de prueba CP-07.	50
D.8. Caso de prueba CP-08.	50
D.9. Caso de prueba CP-09.	50

D.10.Caso de prueba CP-10.	51
D.11.Caso de prueba CP-11.	51
D.12.Caso de prueba CP-12.	51
D.13.Caso de prueba CP-13.	52
D.14.Caso de prueba CP-14.	52
D.15.Caso de prueba CP-15.	53
D.16.Caso de prueba CP-16.	53
D.17.Caso de prueba CP-17.	53
D.18.Caso de prueba CP-18.	54
D.19.Caso de prueba CP-19.	54
D.20.Caso de prueba CP-20.	54

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se va a detallar como se ha llevado a cabo la planificación del proyecto. En la planificación es donde se estudia el coste tanto en tiempo, como en recursos y también, aunque sea un proyecto educacional, el coste económico y sus posibles beneficios.

Con los resultados obtenidos realizaremos la planificación temporal del proyecto y el estudio de viabilidad del mismo.

A.2. Planificación temporal

Como ya se ha mencionado en el inicio del proyecto, se ha optado por seguir la metodología ágil *Scrum*. Se ha intentado seguirla fielmente, con la salvedad de que normalmente en este tipo de proyectos participan varias personas y en este ha participado solo una persona, con la supervisión del tutor.

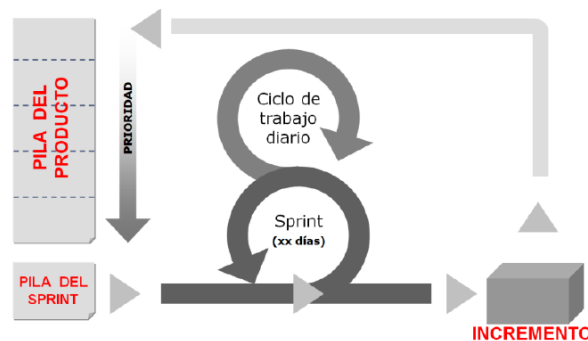


Figura A.1: Diagrama del ciclo iterativo *Scrum*.

Scrum se basa en aplicar una estrategia de desarrollo incremental, que intenta mantener un ritmo de avance constante. Se planifican una serie de *sprints* donde se debían desarrollar una serie de tareas completamente funcionales. La duración de estos *sprints* era de una semana y al final de cada *sprint* se realizaban reuniones para comprobar si se habían cumplido los objetivos y planificar el siguiente *sprint*.

Los requisitos del sistema se registraron en dos formatos:

- **Pila del producto:** como una lista ordenada de todo aquello que el propietario de producto cree que necesita el producto. La pila del producto nunca se da por completada; está en continuo crecimiento y evolución.
- **Pila del *sprint*:** como una lista de las tareas necesarias para construir las historias de usuario que se van a realizar en un *sprint*. Refleja los requisitos vistos desde el punto de vista del equipo de desarrollo.

Para la gestión del proyecto se utilizó *ZenHub*, en ella podíamos hacer el seguimiento de lo que está en revisión, lo que debe probarse, lo que se está haciendo o lo ya cerrado.

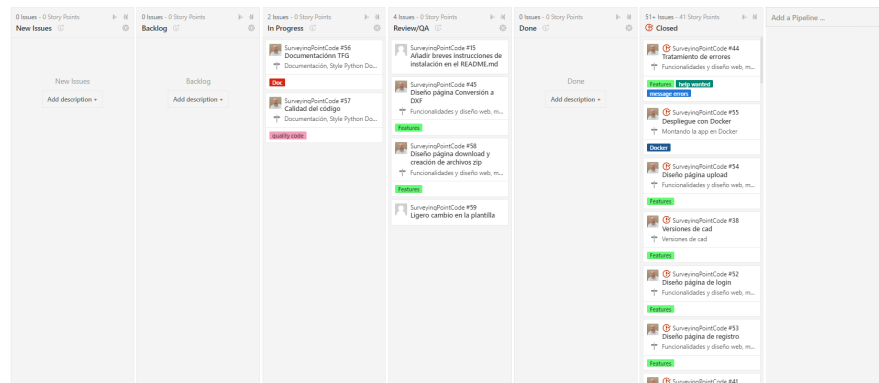


Figura A.2: Gestión de tareas con *ZenHub* en una fase del proyecto.

A continuación se describen los diferentes *sprints* que se han realizado:

***Sprint 1* (24/01/2019 - 30/01/2019)**

En este primer *sprint* y como primera toma de contacto con el proyecto, los objetivos planteados fueron los siguientes:

- Preparación del entorno de trabajo. El proyecto se iba a desarrollar en Python, se instaló la version *Python 3.7* y como entorno de la aplicación *PyCharm 2019.1.1 Community Edition*.

- Repaso de conocimientos en *GIT* y creación del repositorio en *GitHub*. Se refrescaron los conocimientos en *GIT*, en algunos tutoriales online como *git - la guía sencilla* [8] y *Tutorial de Git. Manual básico con ejemplos* [12]
- Formación en *PEPs*. Para conocer bien las guías de estilo y la convenciones de *Python*, se consultaron las guías PEP 8 y PEP 257, mencionadas en la memoria.
- Formación en *L^AT_EX*. Se consulto como guía principal, el libro Edición de Textos Científicos *L^AT_EX* [1]. Y como herramientas para realizar la memoria se instalaron las aplicaciones: *Termaker 5.0.3* y *MikTex 2.9*.
- Diseño del logotipo de la aplicación.



Figura A.3: Logo de *SurveyingPointCode*.

Sprint 2 (31/01/2019 - 06/02/2019)

Los objetivos planteados fueron los siguientes:

- Formalización de la entrada de datos: Se procede a formalizar la gramática que debe tener el archivo de entrada y se determina que: el archivo de entrada será un archivo de texto, compuesto por una o múltiples líneas. Estas líneas serán los puntos medidos en campo y cada línea tendrá la siguiente estructura:
 número de punto, coordenada x , coordenada y , coordenada z , código
 - El numero de punto debe ser de tipo *integer*
 - Las coordenadas x , y , z de tipo *float* o *integer*
 - El código de tipo string, pudiendo estar formado por letras, números y los signos '-' y '+'
- Elección de una herramienta de análisis sintáctico: Se elige una herramienta de análisis sintáctico para poder validar los archivos de entrada. Las opciones eran:

- *Ply*
- *ANTLR*
- *Flex Bison*

Nos decantamos por *Ply* ya que está implementada completamente en *Python* y encaja perfectamente con la filosofía de realizar la mayor parte del proyecto con este lenguaje.

- Creación de un prototipo: Se creó un prototipo que permitía indicarnos si el archivo de entrada era correcto o no. Se comprobó su funcionamiento con dos archivos, uno con la gramática correcta y otro gramática incorrecta, y el resultado fue satisfactorio en ambos casos.

Sprint 3 (07/02/2019 - 13/02/2019)

Como objetivo se planteó seguir con el desarrollo de este primer prototipo, ampliando sus funcionalidades. Se definieron las siguientes tareas:

- Organizar elementos en listas de capas: Se leerán las líneas del fichero de entrada (puntos medidos en campo) y se organizarán en una estructura de datos en forma de lista, donde cada lista contendrá los elementos (puntos) correspondientes a una misma capa. Se obtuvo un resultado correcto.
- Identificar y organizar los diferentes tipos de líneas: Se pretende identificar todos los puntos que forman una línea, y almacenar todas las líneas existentes en estructuras de datos. Se obtuvo un resultado correcto.
- Comenzando con biblioteca *ezdxf*: Se comienza a estudiar la biblioteca *ezdxf* y a realizar pequeñas pruebas, incluyendo estas en el prototipo. Dibujar las líneas y *splines* de un fichero de entrada.

El resultado fue correcto, lo podemos ver en la siguiente imagen.

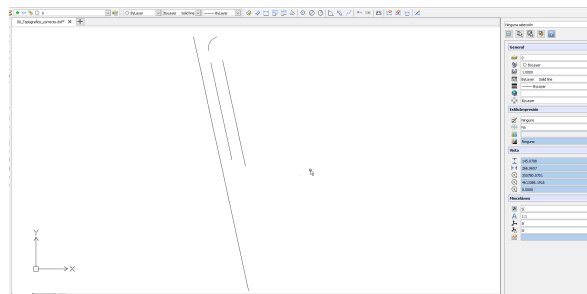


Figura A.4: Logo de *Dibujo con líneas y splines*.

Sprint 4 (14/02/2019 - 20/02/2019)

Una vez acabado el primer prototipo funcional que podía validar un archivo, diferenciar las diferentes líneas y curvas, y generar un archivo DXF, el objetivo de este *sprint* fue comenzar a crear un prototipo basado en una aplicación Web, utilizando el *micro framework* **Flask**. Para ello se definieron las siguientes tareas:

- **Formación en Flask:** Se consultó la documentación de **Flask** mencionada en la memoria.
- **Prototipo con Flask:** Se creó un prototipo que permitía subir un archivo desde el equipo del usuario al servidor. Se obtuvo un resultado correcto.
- **Procesado del archivo subido y descarga del archivo DXF generado:** El prototipo debería validar el archivo subido por el usuario, lo convertiría a DXF, y permitiría descargarlo en su equipo. Se obtuvo un resultado correcto.

Aquí empezó a trabajar con un entorno virtual creado en *Anaconda*¹ por sugerencia del tutor.

Sprint 5 (21/02/2019 - 27/02/2019)

Ya teníamos la aplicación Web funcionando, el siguiente paso sería la validación de usuarios, el uso de una base de datos y a ser posible comenzar a investigar con *Docker* y alojar la base de datos en un contenedor. Para ello se definieron las siguientes tareas:

- **Comenzando con Docker:** Se consultó la documentación de *emphDocker* mencionada en la memoria (ver apartado 5.3 Formación), y se configuró un contenedor *PostGIS* para alojar la base de datos. También se configuró otro contenedor *PgAdmin4* para poder administrar y ver las modificaciones en la base de datos más fácilmente. También serviría para probar como conectar dos contenedores entre sí.
- **Validación de usuarios:** Se implementó la aplicación para que un usuario pudiera logearse y registrarse. Se diseñó el modelo de la base de datos y la base de datos se crearía automáticamente al arrancar la aplicación. Se utilizaron las librerías **SQLAlchemy** y **Flask-Login**, mencionadas en la memoria, de las cuales se estudió su documentación. Se obtuvo un resultado correcto.

¹ *Anaconda*: <https://www.anaconda.com/>

Sprint 6 (08/03/2019 - 13/03/2019)

Este *sprint* se basó en el aprendizaje de *HTML*, formularios y *Bootstrap*. Incluyendo el tratamiento de mensajes `flash` en `Flask`.

También se estudió como incorporar un visor en el navegador, para poder visualizar el plano. Se estudiaron varias opciones:

- Convertir el archivo a formato SVG: Se convirtió el archivo DXF a SVG, con un conversor online², pero el resultado obtenido no era lo esperado. Se probó con otros conversores siendo el resultado igual de malo, por lo que se descartó esta opción.
- Usar el visor *Three-Dxf*: *Three-Dxf*³ en un visor en *JavaScript*, para archivos DXF, que se podría incorporar a cualquier aplicación Web. En su documentación indicaba que soportaba los elementos que usábamos en nuestros DXF, como los tipos de líneas, tipos de símbolos, etc. Pero comprobándolo resultó que no era así, daba errores. Conseguía visualizar archivos muy simples, pero no los que necesitábamos en este proyecto.

Al final se descartó la idea de que la aplicación tuviera un visor incorporado.

Sprint 7 (14/03/2019 - 20/03/2019)

Este *sprint* se basó en implementar funciones que crearan los elementos del archivo DXF. Para ello se definieron las siguientes tareas:

- Creación de capas: Creación de capas en el DXF, a partir de los códigos del archivo de campo. Hay tres capas que se deberían crear siempre:
 - *Point*: Capa que deberá contener todos los puntos.
 - *Altitude*: Capa que deberá contener la altitud de todos los puntos. en forma de texto.
 - *Label*: Capa que deberá contener el código de todos los puntos, en forma de texto.

Se obtuvo un resultado correcto.

- Inserción de puntos en el modelo: Se insertan todos los puntos del archivo de campo en el modelo del dibujo. Se dibujan también, los textos correspondientes al código y a la elevación del punto, cada uno en su correspondiente capa, todo ello interpretando la codificación. Se obtuvo un resultado correcto.

²convertio: <https://convertio.co/es/dxf-svg/>

³Three-Dxf: <https://github.com/gdsestimating/three-dxf>

- Creación de circunferencias: Se crearían círculos, a partir de un punto y un radio, y se guardaría en su correspondiente capa, todo ello interpretando la codificación. Se obtuvo un resultado correcto.
- Creación de líneas: Se crearían líneas, a partir de una serie de puntos, y se guardarían en su correspondiente capa, todo ello interpretando la codificación. Se obtuvo un resultado correcto.
- Creación de cuadrados: Se crearían cuadrados, a partir de dos puntos, y se guardarían en su correspondiente capa, todo ello interpretando la codificación. El cuadrado siempre se dibujaría a la derecha de la alineación definida por estos dos puntos. Se obtuvo un resultado correcto.
- Creación de rectángulos: Se crearían rectángulos, a partir de tres puntos, y se guardarían en su correspondiente capa, todo ello interpretando la codificación. Se obtuvo un resultado correcto.
- Uso de puntos no medidos en campo: Se utilizarían puntos no medidos en campo para la generación de líneas y se guardarían en su correspondiente capa, todo ello interpretando la codificación. Se obtuvo un resultado correcto.

Una vez dibujados los elementos, se mejoró estéticamente el dibujo, modificando el tamaño de los textos y su posición respecto a los puntos.

Sprint 8 (21/03/2019 - 27/03/2019)

En este *sprint* se comenzó con la realización de test unitarios, estudiando las diferentes opciones. También como se podría leer un archivo DXF proporcionado por el usuario, que contuviera símbolos en forma de bloques y una vez, incorporarlos al modelo de dibujo que queríamos generar. Para ello se definieron las siguientes tareas:

- Estudio de una biblioteca de Python para la realización de los test unitarios: Se consideraron las bibliotecas: `unittest` [9] y `pytest` [11]. La opción elegida fue `unittest` esta biblioteca ya se había usado durante la realización del grado.
- Comienzo con los test unitarios.
- Crear una función para cargar un archivo DXF.
- Crear una función extraer los bloques del archivo DXF.
- Incorporar los bloques al modelo del dibujo.

- Asociar cada bloque con cada punto correspondiente, interpretando la codificación.

Se obtuvo un resultado correcto.

Sprint 8 (28/03/2019 - 03/04/2019)

En este *sprint* se comenzó a trabajar con la posibles versiones de CAD que podría generar la aplicación. También se comenzó a trabajar con el archivo de configuración de usuario, definiendo como sería su gramática e implementando un *parser* para poder validarlo. Para ello se definieron las siguientes tareas:

- Versiones de CAD válidas: Se comprobaría la generación del archivo DXF en diferentes versiones y su validez.
- Definición de gramática para el archivo de configuración de usuario.
- Carga y validación del archivo de usuario.

En este *sprint* se detecta un error en el uso de dos *parsers* que se detalla en la memoria, *Conflicto entre parsers*.

Sprint 9 (04/04/2019 - 10/04/2019)

En este *sprint* se comenzó a trabajar con las sesiones de **Flask** multiusuario, se continuó con la definición, e implementación de funciones para la gestión del archivo de configuración de usuario y se soluciona el problema de del conflicto entre los dos *parsers*. Para ello se definieron las siguientes tareas:

- Gestión del archivo de configuración de usuario.
- Problema con el uso de dos *parsers*.
- Carga y validación del archivo de usuario.
- Comienzo con las sesiones en **Flask**.

Sprint 10 (11/04/2019 - 24/04/2019)

Este *sprint* se consideró más largo, había una semana de vacaciones en medio. Básicamente se basó en completar las funcionalidades Web, el diseño de la Web, y el tratamiento de los mensajes de aviso. Para ello se definieron las siguientes tareas:

- Añadir funcionalidades a la Web: Añadir o quitar botones, menús, etc.

- Mejorar el diseño de la Web: Página de inicio, barra de navegación superior, página de login, etc.
- Tratamiento de errores: Ver e implementar las distintas formas, en que se presentaran los errores o mensajes de aviso al usuario.

Se solucionaron errores como, el surgido al añadir la letra «ñ» y la tildes, a la gramática del archivo de configuración (se puede ver la solución en la memoria *Codificación UTF-8*). También un error no detectado anteriormente en la función `calculate_azimut_distance`, con un error de división por 0. Además se comprobó que la paleta de colores de los programas CAD, no era la habitual (se puede ver la solución en la memoria *Paleta de colores en CAD*)

Sprint 11 (25/04/2019 - 01/05/2019)

En este *sprint* y como parte final de la aplicación, el objetivo era facilitar su posterior despliegue en explotación mediante la utilización de contenedores *Docker*. En un principio integrar la aplicación *Flask* y por último ejecutar todos los contenedores a la vez, con el uso de la herramienta *Docker Compose*. Se obtuvo un resultado correcto.

Sprint 12 (02/05/2019 - 15/05/2019)

En este *sprint* se redacta la documentación del proyecto.

Sprint 13 (16/05/2019 - 22/05/2019)

En este *sprint* se continua con la redacción de la documentación del proyecto. Además se añaden las siguientes funcionalidades:

- Mensaje de *Cookies*.
- Tiempo de caducidad de la sesión de usuario
- Mejora de la barra de navegación, incluyendo más funcionalidades.

Se consiguieron los objetivos.

Sprint 14 (23/05/2019 - 29/05/2019)

En este *sprint* se continuará con la redacción de la documentación del proyecto. Además se añadirán las siguientes funcionalidades:

- Ampliación de la base de datos para recoger estadísticas sobre el uso que hace de la aplicación cada usuario.

- Mejora estética en la paleta de colores desplegable.
- Revisión de las licencias.
- Realización de un manual en vídeo.

Se consiguieron los objetivos.

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se analizan los costes del proyecto y también los posibles beneficios del proyecto en caso de comercializarse.

Costes

Costes de personal:

El desarrollo del proyecto, tanto el tiempo empleado en la formación, como la implementación de la aplicación y redactar la documentación generada, ha sido llevado a cabo por una sola persona durante 476 horas. Considerando 40 horas semanales, hemos trabajado 2.98 meses, por lo que redondeamos a 3 meses de trabajo a tiempo completo. Si consideramos que el trabajo lo ha realizado un programador junior, el salario será el siguiente:

Concepto	Coste
Salario mensual neto	1.088,29 €
Retención IRPF (15 %)	296.,27 €
Seguridad Social (29,9 %)	590,56 €
Salario mensual bruto	1.975,12 €
Total 3 meses	5.925,36 €

Tabla A.1: Costes de personal.

La retribución [5] a la Seguridad Social se ha calculado como un 23,60 % por contingencias comunes, más un 5,50 % por desempleo de tipo general, más un 0,20 % para el Fondo de Garantía Salarial y más un 0,60 % de formación profesional. En total un 29,9 % que se aplica al salario bruto.

El porcentaje de IRPF [14] se ha establecido en el 15 % ya que es el considerado como rendimientos de trabajo para la elaboración de obras científicas.

Costes de *hardware* y *software*:

A continuación se presentan los costes por el *hardware* y *software*, utilizado, en el desarrollo del proyecto. Considerando una amortización a 5 años y se ha usado 3 meses.

Concepto	Coste	Coste amortizado
Ordenador personal	1.200 €	60 €
Windows 10 Home	145 €	7,25 €
Total	1.345 €	67,25 €

Tabla A.2: Costes de *hardware* y *software*.

Costes de varios:

Se considera un coste adicional de asesoría técnica, proporcionada en este caso por el tutor en cada *sprint*. Consultando varias fuentes [13] [7], el precio/hora de consultoría o asesoría técnica ronda los 60 €/hora. En este proyecto se han realizado 14 *sprints*, por lo que el coste de asesoramiento técnico asciende a 840 €.

Además, se han considerado también los costes de alquiler de una oficina y el servicio de conexión a Internet.

Concepto	Coste
Asesoría técnica	840 €
Alquiler de oficina	250 €
Internet	120 €
Total	1.210 €

Tabla A.3: Costes varios.

Costes totales:

El coste total del proyecto es:

Concepto	Coste
Personal	5.925,36 €
<i>Hardware & Software</i>	67,25 €
Varios	1.210 €
Total	7.202,61 €

Tabla A.4: Costes totales.

Beneficios

En un principio se desplegará o distribuirá la aplicación de forma gratuita. Se evaluará su grado de aceptación en el mercado y si es viable, se estudiará como distribuir la aplicación comercialmente. Un modelo de negocio podría ser, establecer diferentes cuotas por tramos, en función del tamaño de los archivos o del número de puntos a convertir.

Por ejemplo:

Número de puntos	Cuota
10.000	10 €/mes
30.000	20 €/mes
Sin Límite	40 €/mes

Tabla A.5: Tipos de suscripciones y cuotas

Añadiendo nuevas funcionalidades, se estudiarán otros parámetros en el modelo de negocio como por ejemplo, si el usuario desea:

- Imprimir el plano.
- Almacenar en la aplicación una base de datos con símbolos.
- Unir varios archivos.
- Modificar la escala y el formato del plano.
- ...

En este caso la idea sería ofrecer paquetes con diferentes funcionalidades a diferentes precios, por ejemplo:

- Paquete básico: Conversión de ficheros.
- Paquete intermedio: Conversión de ficheros, cambios de escalas y formatos.
- Paquete avanzado: Conversión de ficheros, cambios de escalas y formatos, impresión de archivos.
- Paquete completo: Todas las opciones.

Podría existir una funcionalidad gratuita, que permitiera visualizar el archivo, sin permitir nada más, para atraer posibles clientes.

Viabilidad legal

Licencias de *software*

En este apartado se mostraran las licencias de las herramientas y bibliotecas de *software* utilizadas en el desarrollo del proyecto. Por otra parte, asignaremos una licencia a este proyecto.

En la tabla siguiente se pueden ver las licencias de las dependencias utilizadas.

Dependencia	Versión	Descripción	Licencia
Flask	1.0.2	<i>Micro framework</i> Python crear aplicaciones Web.	BSD
Ply	3.11	Herramienta de análisis sintáctico en Python.	BSD
ezdxf	0.8.9	Paquete de Python para crear y modificar dibujos DXF.	MIT
SQLAlchemy	1.2.18	Conjunto de bibliotecas SQL de Python y ORM.	MIT
flask-login	0.4.1	Biblioteca para gestionar sesiones de usuarios en Flask .	MIT
WTForms	2.2.1	Biblioteca para generar y validar formularios Web en Python	BSD
bs custom file input	1.3.2	<i>Plugin</i> de Bootstrap para entrada de archivos.	MIT
Bootstrap	4	Bootstrap es una biblioteca multiplataforma para el diseño de sitios Web.	MIT
Bootstrap Colorpicker	3.1.1	<i>Plugin</i> de selector de color modular para Bootstrap 4.	MIT
TinyColor	1.4.1	Es un <i>micro framework</i> que permite la manipulación y conversión de color en JavaScript	MIT
jquery.cookie	1.4.1	Es un <i>plugin</i> de <i>jQuery</i> para gestionar las <i>cookies</i> .	MIT

Tabla A.6: Dependencias del proyecto.

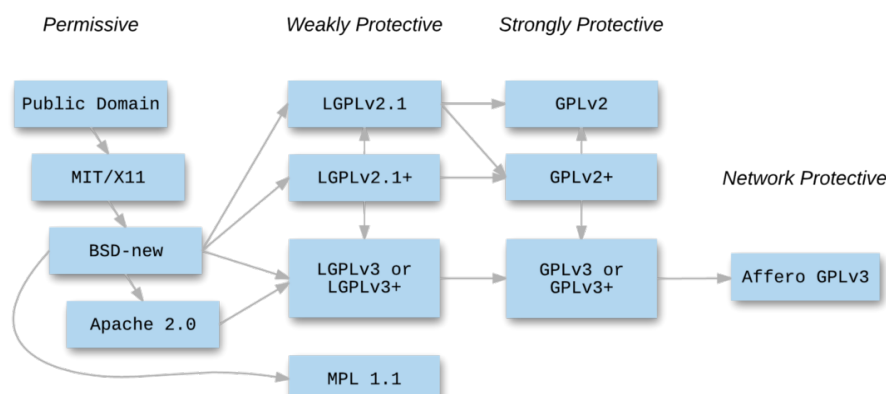


Figura A.5: Compatibilidad entre licencias [10].

Como se puede observar en la figura anterior, podemos aplicar cualquier licencia a nuestro proyecto respecto a las dependencias que tenemos.

Se ha optado por:

GNU General Public License v3.0

«Exige la publicación del código fuente y que todos los trabajos derivados del original conserven la misma licencia GPL, no permite enlaces con módulos privativos (de código cerrado) y requiere que todos los cambios realizados a la versión original sean reflejados en el código fuente con sus respectivos autores. Los derechos de autor deben conservarse tanto en el código fuente como en los binarios.» [2]

Política de *Cookies*

El desarrollo de la aplicación ha implicado el uso de sesiones y por consiguiente, el uso de *Cookies*.

Las obligaciones legales de las *cookies* se encuentran en: el artículo 22.2 LSSI (Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico), este sentido el Grupo de Trabajo del Artículo 29 en su Dictamen 4/2012, indica que, quedan exceptuadas las que tienen por finalidad:

- *Cookies* de entrada del usuario.
- *Cookies* de autenticación o identificación de usuario (únicamente de sesión)
- *Cookies* de seguridad del usuario
- *Cookies* de sesión de reproductor multimedia

- *Cookies* de sesión para equilibrar la carga
- *Cookies* de personalización de la interfaz de usuario
- *Cookies* de complemento (*plugin*) para intercambiar contenidos sociales

En este caso, no es necesario informar, las que usa la aplicación *Surveying-PointCode* son de tipo:

«*Cookies* de autenticación o identificación de usuario (únicamente de sesión).»

Aun así se ha implementado un mensaje informando de que se está utilizando *cookies* y cual es su uso.

El mensaje es el siguiente:

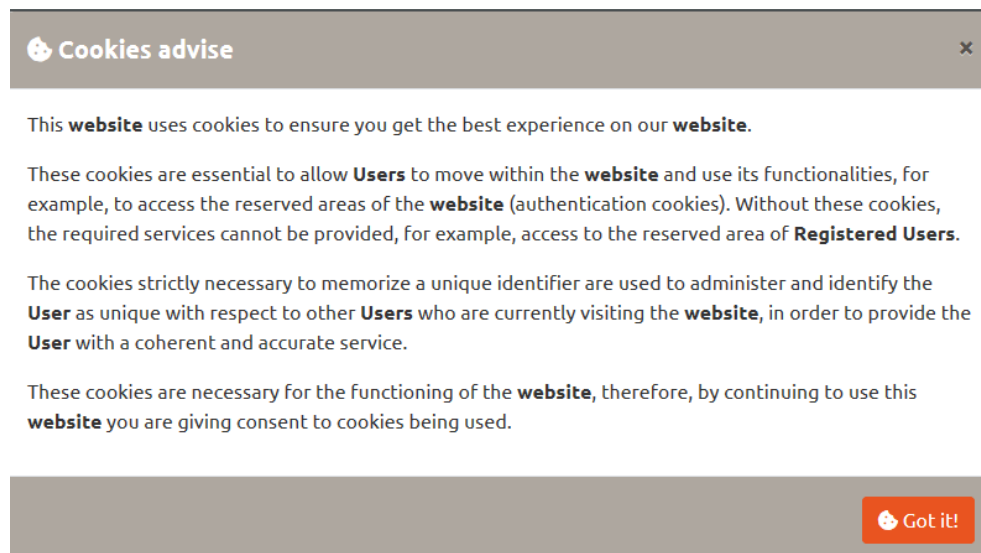


Figura A.6: Mensaje de aviso de *Cookies*.

Uso de la dirección de e-mail

En la aplicación se ha utilizado y registrado el dato del e-mail, para poder registrar y permitir el acceso del usuario. Este dato se considera de carácter personal, ya que hay ocasiones en que este se puede rastrear hasta conocer la identidad de la persona. Todo esto está regulado en la nueva Ley Orgánica de Protección de Datos de Carácter Personal 2018 [6], donde se indica que el usuario debe aceptar el uso de su e-mail, debiendo ser el consentimiento explícito, específico y verificable y ese consentimiento ha de ser revocable en cualquier momento.

En este proyecto, al ser educativo no se han implementado esas opciones. Con este apartado se desea dejar claro que se está en conocimiento de la ley y de como aplicarla.

Licencia en la documentación

Con respecto a la documentación se ha optado por una licencia de tipo *Creative Commons* y se ha elegido la siguiente:

CC BY-NC-SA 4.0 [\[3\]](#)

Esta licencia permite a otros entremezclar, ajustar y construir a partir de su obra con fines no comerciales, siempre y cuando le reconozcan la autoría y sus nuevas creaciones estén bajo una licencia con los mismos términos.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se describen los servicios que ha de ofrecer la aplicación *SurveyingPointCode* y las restricciones asociadas a su funcionamiento. La función principal de la especificación de requisitos es servir como medio de comunicación entre clientes, usuarios, y desarrolladores.

Se recogerán todos los requisitos funcionales y no funcionales.

B.2. Objetivos generales

En este apartado se detallarán los distintos objetivos generales que se persiguen con este proyecto:

- Definir una codificación de puntos en un levantamiento topográfico que permita la automatización del proceso e delineación y mejore la toma de datos en campo.
- Desarrollar una aplicación Web, que permita la conversión de un archivo de campo, con datos topográficos a un archivo DXF, interpretando la codificación y generando todos los elementos del dibujo de forma automática.
- Trabajar con archivos personalizados del usuario como: configuración de la conversión y archivo DXF de símbolos, para la obtención del DXF final.
- El usuario debe estar registrado, para acceder a la aplicación.

B.3. Catalogo de requisitos

A continuación, definirán tanto los requisitos funcionales como los no funcionales necesarios para cumplir con los objetivos generales.

Requisitos funcionales

- **RF-1 Registro de usuario:** el usuario debe poder registrarse, mediante un nombre, un e-mail y una contraseña.
- **RF-2 *Login* de usuario:** el usuario debe poder *logearse*, mediante un e-mail y una contraseña.
- **RF-3 Carga de archivos:** el usuario debe ser capaz de gestionar diferentes tipos de archivos: de campo, de configuración de la conversión y de símbolos.
 - **RF-3.1 Carga de archivo de campo:** el usuario debe poder cargar este archivo con los datos de un levantamiento topográfico, indicando su validez y en caso contrario, indicando cual es su error. Este archivo es obligatorio.
 - **RF-3.2 Carga de archivo de configuración:** el usuario debe poder cargar este archivo con los datos de una configuración personalizada para la conversión, indicando su validez y en caso contrario, indicando cual es su error. Este archivo es opcional.
 - **RF-3.3 Carga de archivo DXF con simbología:** el usuario debe poder cargar este archivo DXF con símbolos personalizados, indicando su validez y en caso contrario, indicando cual es su error. Este archivo es opcional.
- **RF-4 Conversión:** el usuario debe ser capaz de generar un archivo DXF, partiendo del archivo de campo, pudiendo configurar esa conversión utilizando los archivos opcionales. Debe también existir la posibilidad de configurar desde cero o modificar una configuración de la conversión existente a través de la interfaz gráfica. El usuario debe poder elegir la versión de CAD para el DXF a generar, y ponerle un nombre personalizado al archivo generado.
 - **RF-4.1 Asociar capas y colores:** el usuario debe poder asociar capas y colores, a los códigos, a través del archivo configuración de la conversión o de la interfaz.
 - **RF-4.2 Asociar símbolos:** el usuario debe poder asociar símbolos a los códigos. Previamente debe haber cargado un archivo válido con símbolos.

- **RF-4.3 Elección de versión de CAD:** el usuario debe poder elegir la versión de CAD para generar el DXF, a través de un desplegable con las versiones disponibles.
- **RF-4.4 Elección de nombre del archivo DXF generado:** el usuario debe poder dar un nombre personalizado al archivo DXF generado.
- **RF-5 Descarga de archivos :** el usuario debe poder descargar los archivos generados a su equipo de forma individual o varios archivos en formato comprimido.
- **RF-6 Logout del usuario:** el usuario debe poder cerrar la sesión.
- **RF-7 Limpieza de archivos almacenados en el servidor:** la aplicación debe eliminar los archivos utilizados durante la sesión, al cerrarse la sesión.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la aplicación debe ser intuitiva, con una curva baja de aprendizaje y mensajes de errores claros para el usuario.
- **RNF-2 Soporte:** la aplicación debe funcionar en los navegadores Web más usuales, como: Google Chrome, Mozilla Firefox y Microsoft Edge.
- **RNF-3 Rendimiento:** la aplicación debe tener unos tiempos de conversión del archivo mínimos, por ejemplo, con archivos de campo de tamaño grande, 5.000 puntos, debe realizar la conversión en no más de 5 segundos.
- **RNF-4 Escalabilidad:** la aplicación debe ser desarrollada de manera que permita la escalabilidad de la misma de forma sencilla.
- **RNF-5 Seguridad:** la aplicación debe gestionar de forma adecuada todos los datos sensibles, como claves, tokens, etc.

B.4. Especificación de requisitos

A continuación se mostrarán los diagramas de casos de uso y el detalle de cada uno de ellos.

Actores

En este sistema, existe un actor que interactuará con el sistema, y el propio sistema.

Casos de uso

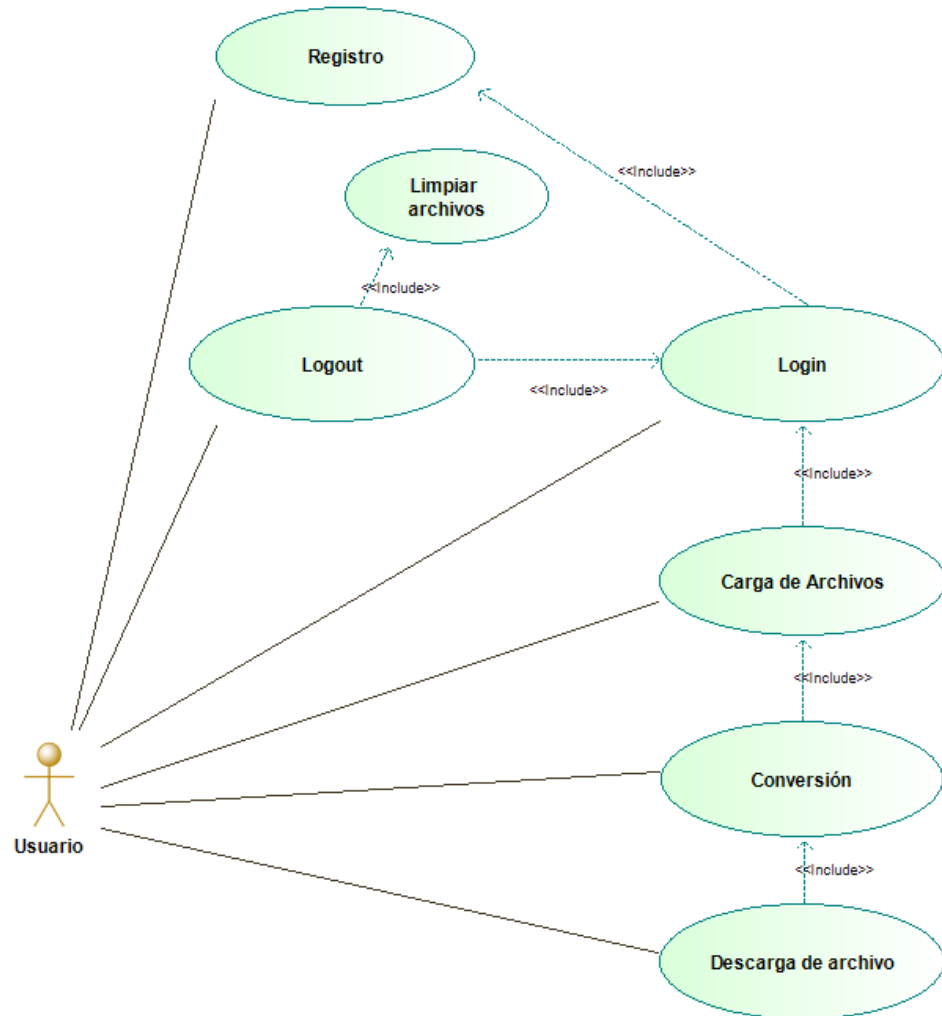


Figura B.1: Diagrama de general de casos de uso.

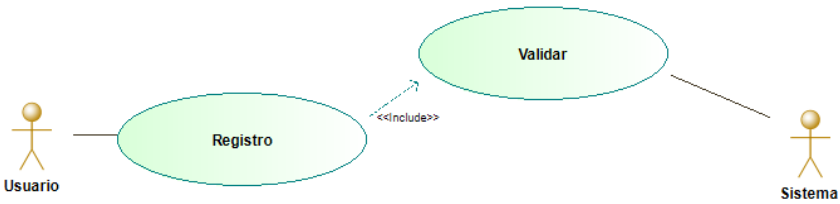


Figura B.2: Diagrama de caso de uso CU-01

CU-01	Registro del usuario
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-1
Descripción	Permite al usuario registrarse.
Precondición	El usuario no debe estar <i>logeado</i> .
Acciones	<div>1. El usuario entra en la aplicación y accede a la página de registro.</div> <div>2. Introduce su nombre, e-mail, contraseña y confirma esta.</div> <div>3. El usuario confirma el registro con el botón <i>Register</i></div>
Postcondición	<div>■ La aplicación va a la pantalla de <i>Login</i></div> <div>■ mensaje: <i>You are now a registered user!</i>.</div>
Excepciones	<div>■ mensaje: <i>Please use a different username.</i></div> <div>■ mensaje: <i>Please use a different email address.</i></div> <div>■ mensaje: <i>Field must be equal to password.</i></div>
Importancia	Alta

Tabla B.1: CU-01 Registro de usuarios.

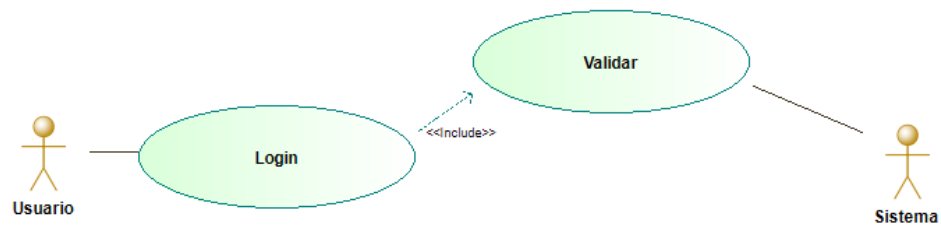


Figura B.3: Diagrama de caso de uso CU-02.

CU-02	<i>Login</i> de usuario
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-2
Descripción	Permite al usuario <i>logearse</i> .
Precondición	El usuario debe estar registrado.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la aplicación y accede a la página de <i>login</i>. 2. Introduce su e-mail y contraseña. 3. El usuario confirma el <i>login</i> con el botón <i>Submit</i>
Postcondición	La aplicación va a la pantalla de <i>Upload File</i>
Excepciones	<ul style="list-style-type: none"> ▪ mensaje: <i>Invalid email address</i>. ▪ mensaje: <i>Invalid username or password!</i>
Importancia	Alta

Tabla B.2: CU-02 *Login* de usuarios.

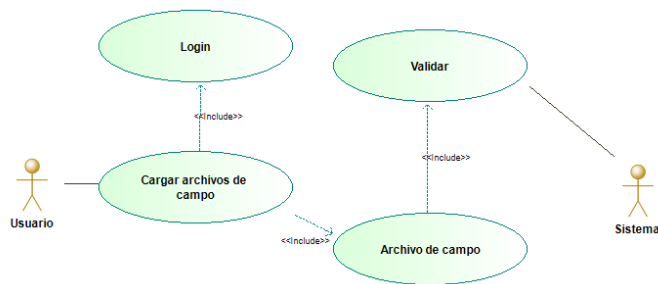


Figura B.4: Diagrama de caso de uso CU-03.

CU-03	Carga de archivo de campo
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-3,RF-3.1, RF-2
Descripción	Permite al usuario cargar un archivo de campo.
Precondición	El usuario debe estar logeado.
Acciones	El archivo debe tener extensión <code>txt</code> o <code>csv</code> . <div>1. El usuario selecciona el archivo de campo, bien a través del botón <i>Browse</i>, o arrastrando el archivo hasta la caja <i>Topographical File data</i>. 2. El nombre del archivo aparece en la caja <i>Topographical File data</i>. 3. El usuario confirma la carga con el botón <i>Upload</i></div>
Postcondición	La aplicación cambia a la pantalla de <i>Upload File</i>
Excepciones	<div>■ mensaje: <i>Topographic data file has the following errors...</i> ■ mensaje: <i>Topographic data file is empty</i>. ■ mensaje: <i>The number of points with TC code in the topographic data file is not multiple of 2</i>. ■ mensaje: <i>The number of points with TR code in the topographic data file is not multiple of 3</i>.</div>
Importancia	Alta

Tabla B.3: CU-03 Carga de archivo de campo.

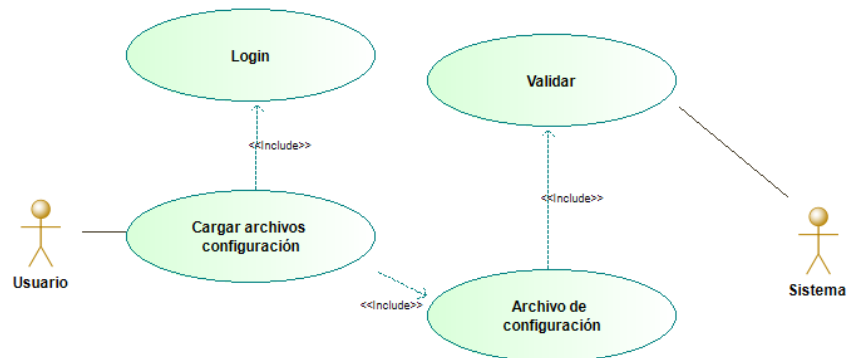


Figura B.5: Diagrama de caso de uso CU-04

CU-04	Carga de archivo de configuración
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-3, RF-3.1, RF-3.2, RF-2
Descripción	Permite al usuario cargar un archivo de configuración de la conversión.
Precondición	El usuario debe estar logeado. El archivo debe tener extensión <code>txt</code> o <code>csv</code> . El archivo de campo debe estar seleccionado
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el archivo de configuración, bien a través del botón <i>Browse</i>, o arrastrando el archivo hasta la caja <i>User Configuration File</i>. 2. El nombre del archivo aparece en la caja <i>User Configuration File</i>. 3. El usuario confirma la carga con el botón <i>Upload</i>

CU-04	Carga de archivo de configuración
Postcondición	
<ol style="list-style-type: none"> 1. Postcondición 1, no se muestran errores: La aplicación cambia a la pantalla de <i>Upload File</i>. Aparece cargada la configuración en la tabla de la interfaz. 2. Postcondición 2, e muestran errores: La aplicación cambia a la pantalla de <i>Upload File</i>. Aparece cargada la configuración en la tabla de la interfaz. Pueden mostrarse estos 2 errores: <ul style="list-style-type: none"> ▪ mensaje: <i>User configuration has different colors on the same lines.</i> ▪ mensaje: <i>Some color of the user configuration is not a CAD color.</i> <p>Estos errores se deben subsanar por parte del usuario en esta pantalla sin necesidad de volver a cargar el archivo.</p> 	
Excepciones	
<ul style="list-style-type: none"> ▪ mensaje: <i>User configuration file file has the following errors...</i> ▪ mensaje: <i>User configuration file is empty.</i> ▪ mensaje: <i>User configuration file has duplicate items on different lines.</i> 	
Importancia	Alta

Tabla B.4: CU-04 Carga de archivo de configuración.

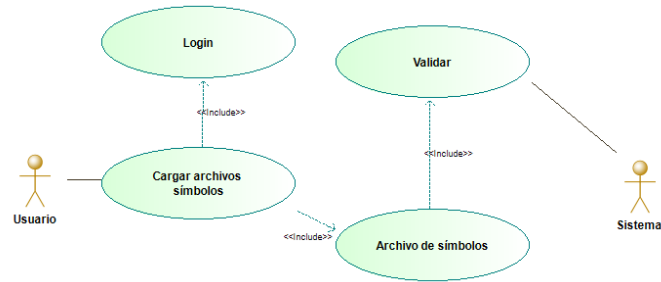


Figura B.6: Diagrama de caso de uso CU-05.

CU-05	Carga de archivo de símbolos
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-3, RF-3.1, RF-3.3, RF-2
Descripción	Permite al usuario cargar un archivo DXF con símbolos.
Precondición	El usuario debe estar logeado. El archivo debe tener extensión dxf . El archivo de campo debe estar seleccionado
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el archivo de símbolos, bien a través del botón <i>Browse</i>, o arrastrando el archivo hasta la caja <i>CAD Symbols File</i>. 2. El nombre del archivo aparece en la caja <i>CAD Symbols File</i>. 3. El usuario confirma la carga con el botón <i>Upload</i>
Postcondición	La aplicación cambia a la pantalla de <i>Upload File</i>
Excepciones	<ul style="list-style-type: none"> ■ mensaje: <i>CAD symbols file does not contain blocks</i>.
Importancia	Alta

Tabla B.5: CU-05 Carga de archivo de símbolos.

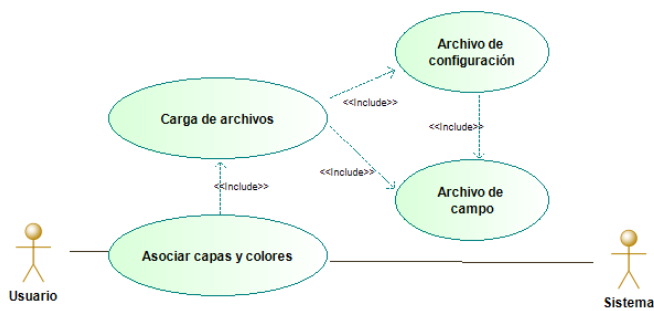


Figura B.7: Diagrama de caso de uso CU-06.

CU-06	Asociar capas y colores
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-4, RF-4.1, RF-3, RF-3.1
Descripción	Permite al usuario asociar capas y colores, a los códigos extraídos del archivo de campo
Precondición	El archivo de campo debe estar cargado.
Acciones	<div>1. El usuario introduce nombres de capas y selecciona colores, en la tabla, a través de la interfaz. En el caso de haber cargado un archivo de configuración esta aparecerá cargada en la tabla y se podrá modificar por el usuario.</div>
Postcondición	Los nombres de capas y colores seleccionados deben permanecer en la interfaz.
Excepciones	-
Importancia	Alta

Tabla B.6: CU-06 Asociar capas y colores.

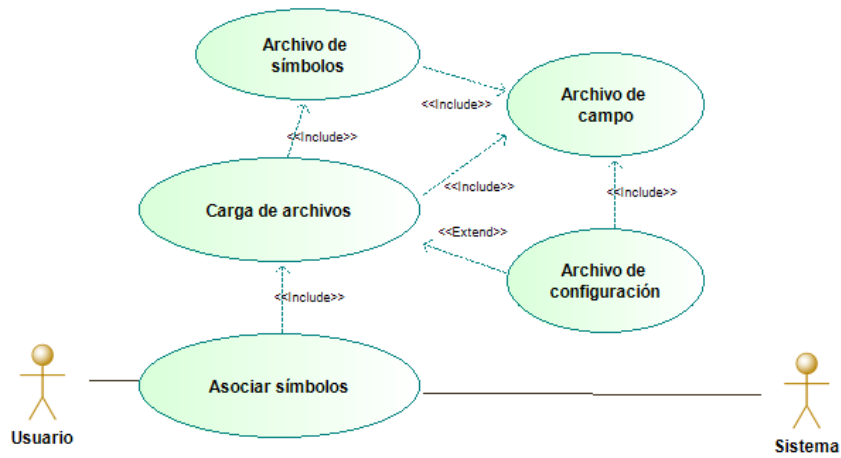


Figura B.8: Diagrama de caso de uso CU-07

CU-07	Asociar símbolos
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-4, RF-4.2, RF-3, RF-3.1
Descripción	Permite al usuario asociar símbolos, a los códigos extraídos del archivo de campo
Precondición	El archivo de campo debe estar cargado. El archivo de símbolos debe estar cargado.
Acciones	<ol style="list-style-type: none"> 1. El usuario elige el símbolo a asociar a cada capa mediante un botón desplegable que contienen todos los símbolos disponibles.
Postcondición	Los símbolos elegidos deben permanecer en la interfaz.
Excepciones	-
Importancia	Alta

Tabla B.7: CU-07 Asociar símbolos.

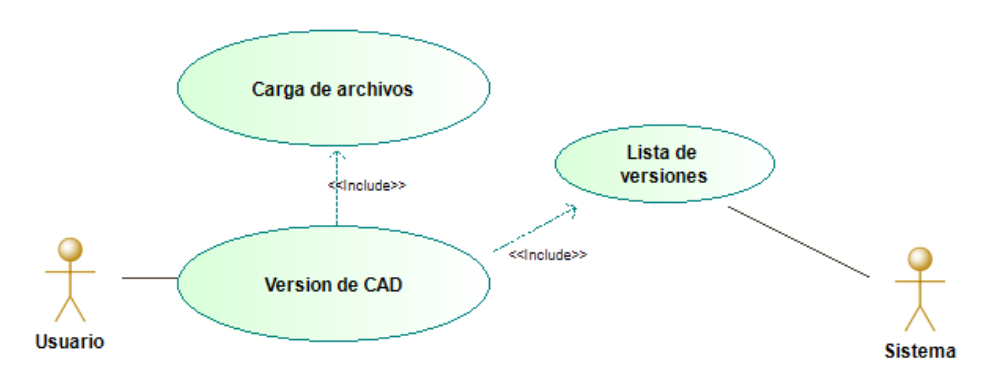


Figura B.9: Diagrama de caso de uso CU-08.

CU-08	Elegir versión de CAD
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-4, RF-4.3, RF-3, RF-3.1
Descripción	Permite al usuario elegir la versión de CAD para el archivo DXF generado
Precondición	El archivo de campo debe estar cargado. Deben ser accesibles las versiones de CAD disponibles.
Acciones	<div>1. El usuario elige la versión de CAD mediante un botón desplegable que contienen todos los símbolos disponibles.</div>
Postcondición	La versión elegida debe permanecer en la interfaz.
Excepciones	-
Importancia	Alta

Tabla B.8: CU-08 Elegir versión de CAD.

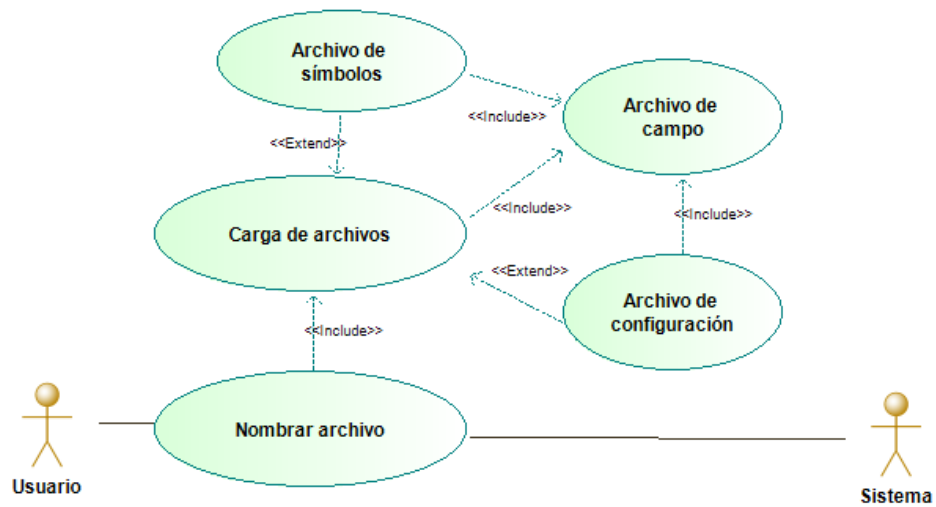


Figura B.10: Diagrama de caso de uso CU-09.

CU-09	Dar nombre al archivo DXF generado
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-4, RF-4.4, RF-3, RF-3.1
Descripción	Permite al usuario dar un nombre personalizado al archivo DXF generado
Precondición	El archivo de campo debe estar cargado.
Acciones	<ol style="list-style-type: none"> 1. El usuario escribe un nombre en la caja llamada <i>Filename</i> 2. El nombre del archivo aparece en la caja.
Postcondición	El nombre del archivo debe permanecer en la interfaz.
Excepciones	-
Importancia	Media

Tabla B.9: CU-09 Dar nombre al archivo DXF generado.

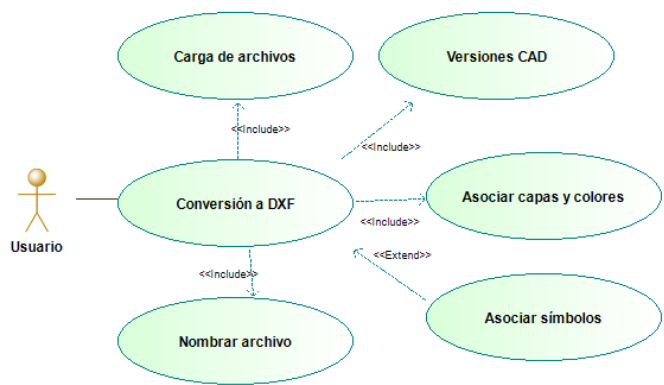


Figura B.11: Diagrama de caso de uso CU-10.

CU-10	Conversión a DXF
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-4, RF-4.1,RF-4.2,RF-4.3, RF-4.4, RF-3, RF-3.1 RF-3.2, RF-3.3
Descripción	Permite al usuario generar un archivo DXF
Precondición	El archivo de campo debe estar cargado.
Acciones	<div>1. El usuario confirma la conversión con el botón <i>Convert</i></div>
Postcondición	La aplicación cambia a la pantalla de <i>Download File</i> mensaje: <i>File successfully converted!</i>
Excepciones	<div><div>■ mensaje: <i>User configuration file has duplicate items on different lines.</i></div><div>■ mensaje: <i>User configuration has different colors on the same lines.</i></div></div>
Importancia	Alta

Tabla B.10: CU-10 Conversión a DXF.

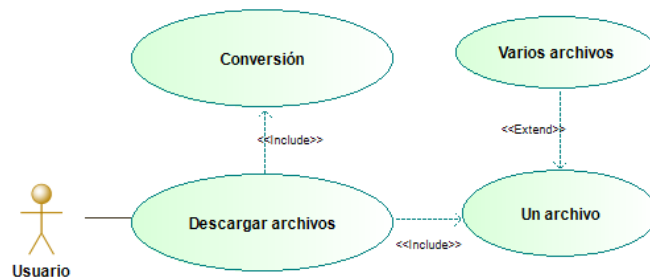


Figura B.12: Diagrama de caso de uso CU-11.

CU-11	Descargar archivos
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-5, RF-4, RF-4.1, RF-4.2, RF-4.3, RF-4.4
Descripción	Permite al usuario descargar a su equipo los archivos generados.
Precondición	Deben existir archivos convertidos a DXF.
Acciones	<ol style="list-style-type: none"> El usuario puede elegir descargar un archivo individualmente. <ul style="list-style-type: none"> El usuario pulsa sobre el archivo a descargar. El archivo se descarga correctamente con formato dxf. El usuario puede elegir descargar varios archivos a la vez. <ul style="list-style-type: none"> El usuario pulsa sobre el botón <i>Download all</i>. El archivo se descarga correctamente con formato comprimido zip.
Postcondición	La aplicación permanece la pantalla de <i>Download File</i>
Excepciones	-
Importancia	Alta

Tabla B.11: CU-11 Descargar archivos.

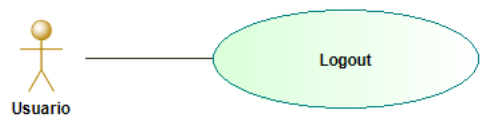


Figura B.13: Diagrama de caso de uso CU-12

CU-12	<i>Logout</i>
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-6, RF-2
Descripción	Permite al usuario cerrar la sesión.
Precondición	El usuario debe estar <i>logeado</i> .
Acciones	<div>1. El usuario pulsa el botón <i>Sign out</i>.</div> <div>2. La aplicación va a la pantalla de bienvenida.</div>
Postcondición	La aplicación cierra la sesión.
Excepciones	-
Importancia	Alta

Tabla B.12: CU-12 *Logout*.



Figura B.14: Diagrama de caso de uso CU-13

CU-12	Limpiar archivos
Versión	1.0
Autor	José Eduardo Risco Sánchez-Cortés
Requisitos asociados	RF-7, RF-6, RF-2
Descripción	Permite al sistema eliminar los archivos usados durante la sesión.
Precondición	El usuario debe estar <i>logeado</i> . El usuario debe haber subido archivos o realizado conversiones.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa el botón <i>Sign out</i>. 2. La aplicación va a la pantalla de bienvenida.
Postcondición	La aplicación elimina todos los archivos utilizados en la sesión.
Excepciones	-
Importancia	Media

Tabla B.13: CU-13 Limpiar archivos.

Especificación de diseño

C.1. Introducción

En este apartado se describe cómo se ha diseñado la aplicación *SurveyingPointCode* para cumplir con los objetivos y las especificaciones definidas anteriormente.

Se definen los datos a utilizar por la aplicación así como el diseño procedimental y su arquitectura.

C.2. Diseño de datos

Las entidades de esta aplicación están enfocadas a la generación de un dibujo final. A continuación se describe cuales son y como se relacionan entre ellas.

- **Dibujo:** Almacena todos los elementos que componen el dibujo, tanto geométricos, como no geométricos. Está compuesto por: un nombre, capas, puntos y textos. De forma opcional puede contener: líneas, *splines*, círculos y símbolos.
- **Capa:** Todos los elementos del dibujo se guardan en capas. Está compuesto por: un nombre y un color.
- **Punto:** Elemento básico del dibujo. Está compuesto por: un número, una coordenada X , una coordenada Y , una coordenada Z , un código y una capa.
- **Línea:** Formada por una lista de puntos. Está compuesto por: una lista de puntos y una capa.

- **spline:** Formada por una lista de puntos. Está compuesto por: una lista de puntos y una capa.
- **Círculo:** Está compuesto por: un punto, un radio y una capa.
- **Bloque:** Elemento interno de CAD que almacena un símbolo. Está compuesto por: un nombre, un punto y una capa.
- **Texto:** Texto asociado a un punto, en este caso puede ser, el número del punto, su altitud o su código. Está compuesto por: un punto, el tamaño del texto, el desplazamiento con respecto al punto, el texto y una capa.

Aunque no se ha implementado como una base de datos, la relación entre los elementos dentro de la biblioteca **ezdxf**, se comporta de la misma forma.

A continuación se puede ver el diagrama relacional. Se ha representado en dos diagramas para facilitar su comprensión (en un mismo diagrama saldrían líneas superpuestas y sería muy confuso de interpretar).

En la Figura C.1, se ven las relaciones de todos los elementos con el elemento principal, el dibujo.

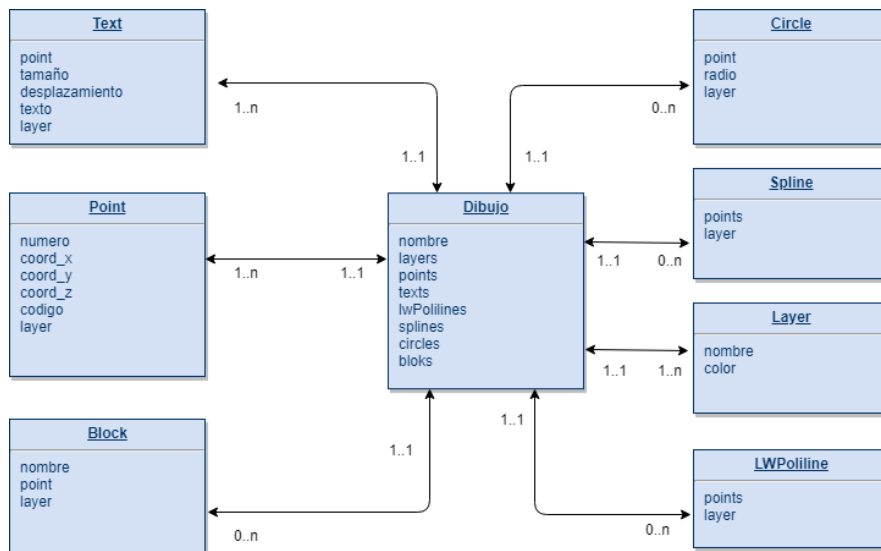


Figura C.1: Diagrama relacional principal.

En la Figura C.2, se ven las relaciones entre sí, del resto de elementos.

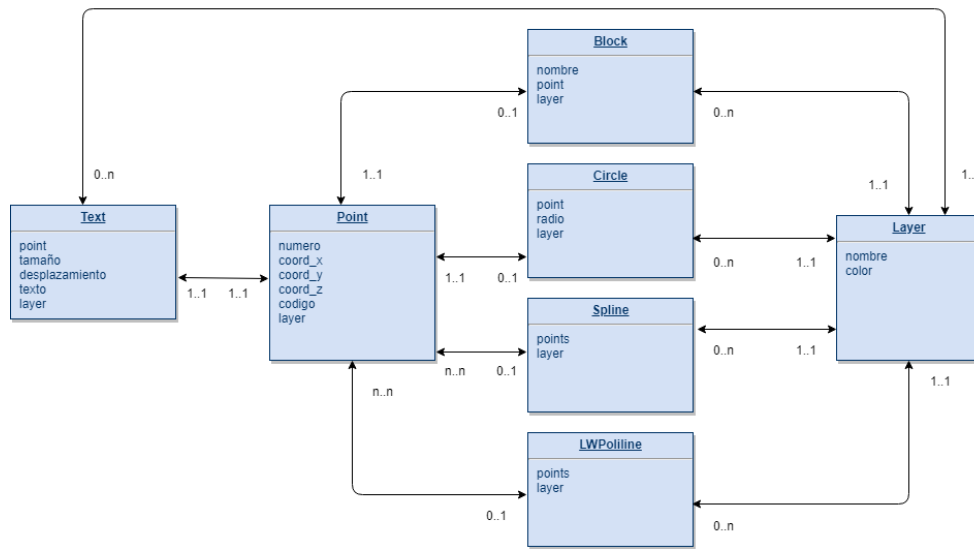


Figura C.2: Diagrama relacional elementos del dibujo.

C.3. Diseño arquitectónico

Como se ha detallado en la memoria (ver apartado 4.2. Patrones de diseño), para el diseño de la aplicación se ha utilizado el patrón arquitectónico **MTV**, que es similar al MCV, pero con alguna diferencia. **T** significa *Template*, plantilla, que son los datos que se presentan al usuario y **V** significa vista, la capa de la lógica de negocios. **MTV** es la arquitectura usada por **Flask**.

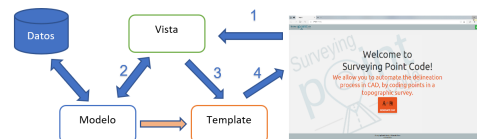


Figura C.3: Esquema de un patrón MTV.

Como sistema de base de datos se ha utilizado **PostGIS**. Para tener más independencia de la base de datos se ha optado por usar un **ORM**, *SQLAlchemy*. El uso de un ORM, simplificará el desarrollo y si en un futuro se decide cambiar de base de datos, será mas sencilla la adaptación de la aplicación.

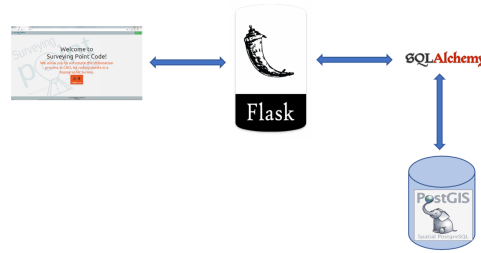


Figura C.4: Diseño arquitectónico con ORM y BBDD.

La aplicación se ha terminado de construir utilizando el sistema de contenedores *Docker*. Con el uso de esta infraestructura se mejora la capacidad de ejecutar varios procesos y aplicaciones por separado y, al mismo tiempo, también conserva la seguridad que tendría con sistemas separados. Los contenedores *Docker* son una especie de máquinas virtuales, pero más ligeras, que permiten encapsular cualquier arquitectura, convirtiéndola en un contenedor portable y autosuficiente.

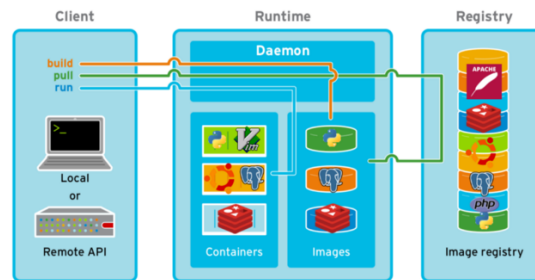


Figura C.5: Arquitectura de *Docker* [4].

como se puede ver en la imagen anterior, *Docker* está compuesto por tres partes:

- *Docker client*: herramienta en línea de comandos, responsable para comunicarnos con el servidor *Docker*. La comunicación se lleva a cabo mediante una *RESTful API* a la cual solicitamos operaciones.
- *Docker Runtime*: este servicio, que se ejecuta como un demonio en un sistema operativo, construye, ejecuta y descarga las imágenes de los contenedores. El demonio puede ejecutarse en el mismo sistema que el cliente *Docker* o de forma remota.

- *Docker Registry*: aquí se almacenan imágenes para uso público o privado. El conocido registro público es *Docker Hub*, y almacena múltiples imágenes desarrolladas por la comunidad. Existe también la posibilidad de crear registros privados.

La arquitectura final basada en *Docker* que se ha implementado, queda de la siguiente forma:

- **API Flask**: contiene la aplicación desarrollada en *Flask* y esta conectado con el contendedor *PostGIS*.
- **PostGIS**: contiene la base de datos y esta conectado con los contendores *API Flask* y *PgAdmin*.
- **PgAdmin**: contiene la aplicación *PgAdmin 4*, para la gestión de la base de datos y esta conectado con el contendedor *PostGIS*.

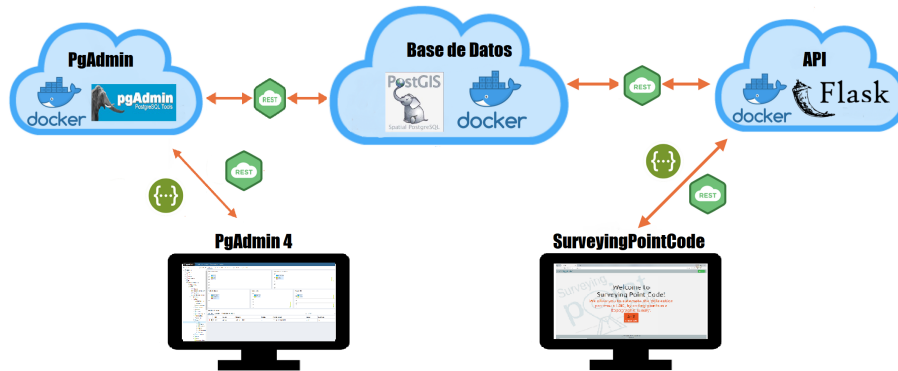


Figura C.6: Arquitectura final de la aplicación con *Docker*

C.4. Diseño procedimental

En este apartado se presenta la especificación procedimental del algoritmo utilizado en forma de diagrama de flujo y diagrama de secuencia de la aplicación.

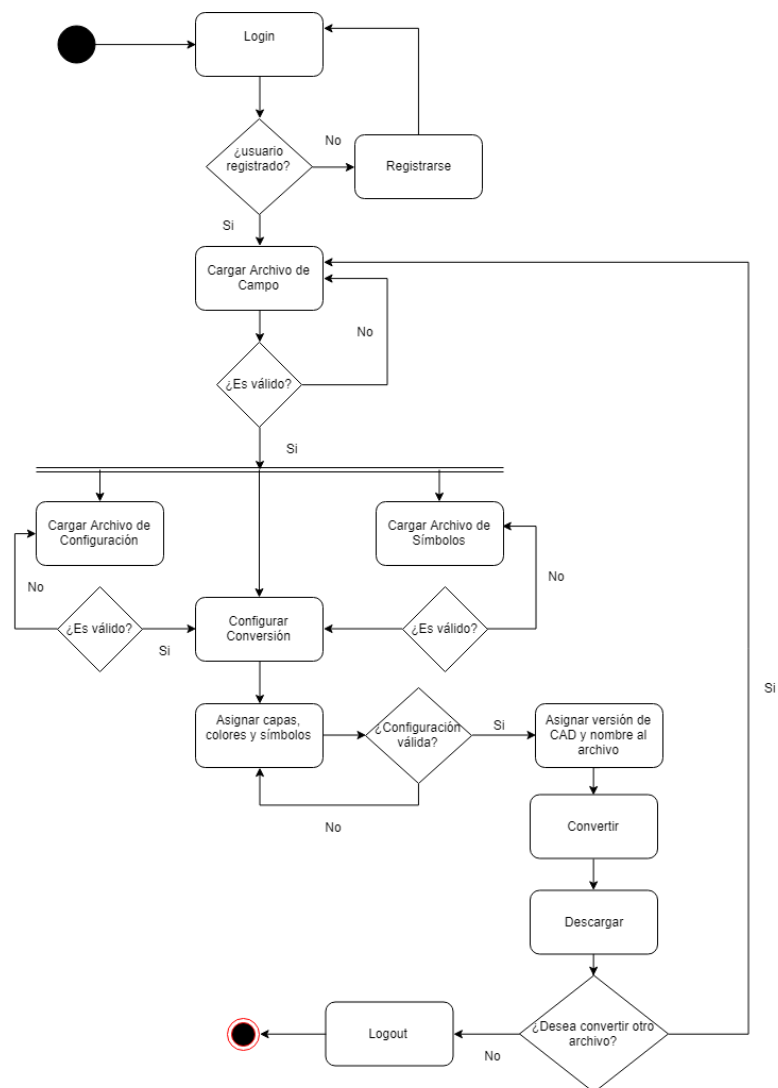


Figura C.7: Diagrama de flujo de la aplicación

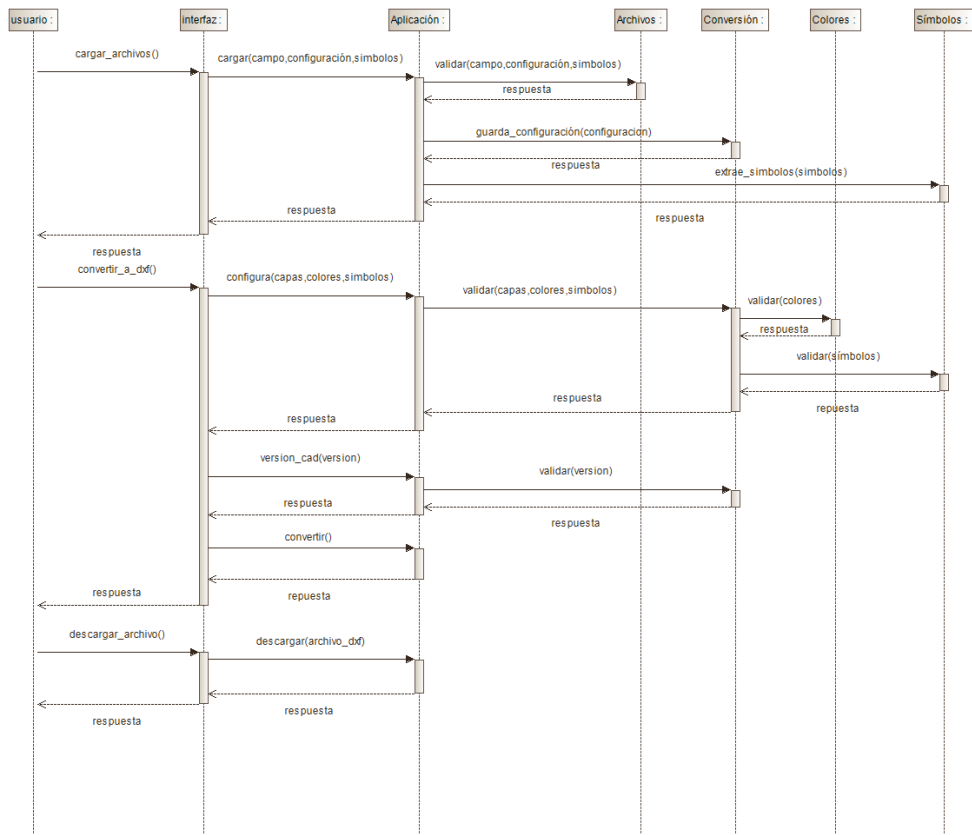


Figura C.8: Diagrama de secuencia general

Documentación técnica de programación

D.1. Introducción

En este apartado se detalla la documentación técnica de programación, estructura de la aplicación, instalación de la aplicación, test realizados, etc. Datos necesarios que pueden servir de guía a cualquier programador pretenda continuar con este proyecto.

D.2. Estructura de directorios

Flask permite una gran libertad a la hora de estructurar un proyecto, aunque por convención sigue una estructura determinada. Esta aplicación ha seguido esa estructura. A continuación se expone:

- `SurveyingPointCode/`: directorio raíz de la aplicación que contiene archivos como: `surveyingpointcode.py` y `config.py`, que son de inicio y configuración de la aplicación, respectivamente. También se encuentra un archivo con los requerimientos necesarios `requirements.txt` y un archivo con las variables de entorno que se deben cargar al sistema `app-env`. Por otra parte, se incluyen los archivos necesarios para el despliegue con *Docker*, estos se detallarán más adelante.
- `SurveyingPointCode/app/`: contiene todos los archivos que definen la lógica del negocio.
- `SurveyingPointCode/app/templates/`: contiene todos los archivos con las plantillas *html*.

- `SurveyingPointCode/app/static/`: contiene todos los archivos estáticos.
- `SurveyingPointCode/app/static/css`: contiene todos los archivos con los estilos.
- `SurveyingPointCode/app/static/js`: contiene todos los archivos *JavaScript*.
- `SurveyingPointCode/app/static/images/`: contiene todos los archivos con las imágenes de la que se usan en la aplicación Web.
- `SurveyingPointCode/app/static/webfonts/`: contiene todos los archivos con las fuentes de la que se usan en la aplicación Web.
- `SurveyingPointCode/tmp/`: contiene todos los archivos temporales que se generan en cada sesión, tanto los archivos subidos por el usuario, como las conversiones.
- `Archivos_Prueba/`: contiene diferentes archivos de ejemplo para cargar en la aplicación, tanto válidos como con errores.
- `Archivos_Prueba/Datos_de_campo/`: contiene diferentes archivos de campo, procedentes de un levantamiento topográfico.
- `Archivos_Prueba/Configuración_usuario/`: contiene diferentes archivos de configuración de la conversión.
- `Archivos_Prueba/Simbolos/`: contiene diferentes archivos DXF, algunos válidos con símbolos definidos y otros vacíos.
- `Test_unitarios/`: contiene diferentes archivos para realizar los test unitarios.

D.3. Manual del programador

Este manual tiene como objetivo servir de referencia a futuros programadores que trabajen en la aplicación o a cualquier persona interesada en su construcción.

Existen dos vías a seguir:

1. Configuración del entorno de desarrollo en un equipo instalando todas las herramientas necesarias.
2. Utilizar *Docker* como entorno de desarrollo.

En este proyecto se han seguido las dos vías. Se llevó a cabo todo el proceso con la primera, hasta que la aplicación estuvo montada completamente en *Docker*, por lo que se siguió con la segunda vía. El proyecto se ha desarrollado utilizando un equipo con sistema operativo *Linux* en su distribución *Linux Mint 18.3 Sylvia*. Los comandos a continuación detallados son para ese sistema operativo.

Configuración del entorno de desarrollo en un equipo

Esta es la forma más convencional de realizar un desarrollo, instalar las herramientas, bibliotecas y módulos en el equipo. A continuación se describen brevemente los pasos seguidos:

1. En primer lugar descargar el código de la aplicación que se encuentra en *GitHub* o clonar el repositorio al equipo local.
2. Instalación de *Anaconda* con Python 3.6. La opción de usar *Anaconda* es para facilitar la opción de trabajar con entornos virtuales personalizados, sin necesidad de instalar dependencias directamente en nuestro equipo.
3. Creación de un entorno virtual en *Anaconda*.

```
conda create -n nombre_de_entorno python=3.6 anaconda
```

4. Activar el entorno virtual en *Anaconda*.

```
source activate nombre_de_entorno
```

5. Instalar las dependencias dentro del entorno virtual creado. Para ello necesitamos el archivo

```
requirements.txt, mencionado anteriormente. pip install -r requirements.txt
```

Para trabajar con la base de datos *PostGIS*, se utiliza un contenedor *Docker*. En este punto se debe tener instalado *Docker*. El contenedor de *PostGIS* se crea con este comando:

```
sudo docker run --name=postgis --hostname=postgres --network=pgnetwork
-d -e POSTGRES_USER=tfgr -e POSTGRES_PASS=f04f1b4d7734f0dc3c4da46f19c0a9f49b56
-e POSTGRES_DBNAME=tfgr -e ALLOW_IP_RANGE=0.0.0.0/0
-p 5432:5432 -v pg_data:/var/lib/postgresql --restart=always mdillon/postgis
```

siendo sus parámetros de conexión con la BBDD:

```
user=tfgr
password=f04f1b4d7734f0dc3c4da46f19c0a9f49b56
db_name=tfgr
```

```
port=5432
host=postgis
```

El entorno de desarrollo está configurado. El *IDE* o editor de texto utilizado es una opción personal del desarrollador, en este proyecto se ha usado *PyCharm*.

Utilizar *Docker* como entorno de desarrollo.

Entre las múltiples ventajas de usar *Docker*, es importante destacar que se puede usar tanto en la etapa de desarrollo como, finalmente en el despliegue. Garantiza que el entorno de desarrollo va a ser siempre el mismo, independientemente del equipo en que se trabaje.

En este proyecto, el uso de *Docker* facilitará la continuidad del desarrollo, solamente hace falta tener instalado en el equipo, la aplicación *Docker*. Siguiendo las instrucciones del archivo `README.txt` que a continuación se enumeran:

1. Instalación de *Docker*.
2. Instalación de *Docker Compose*.
3. Clonar el repositorio.
4. Ejecutar en el directorio `SurveyingPointCode/`:

```
sh app_install.sh
```

Cada vez que realicemos un cambio en el código, debemos actualizar este cambio en el contenedor correspondiente, y lo haremos de la siguiente forma:

- a) Ejecutamos `docker-compose down` para detener el contenedor.
- b) Ejecutamos `docker-compose up -d` para volver a construir los contenedores y levantarlos.

Los archivos `Dockerfile` y `docker-compose.yml`, ya comentados en el documento de la memoria (ver apartado Despliegue de la aplicación — *Docker Compose*), son la base para la construcción de los contenedores y por consiguiente, de los servicios ofrecidos.

Como contrapunto, se ha encontrado una desventaja al usar *Docker* como entorno de desarrollo, el tiempo invertido en volver a construir el contenedor. A medida que el proyecto crece y se van aumentando las dependencias, el tiempo de creación del contenedor se alarga. Es conveniente seguir investigando sobre como solucionar este problema.

D.4. Compilación, instalación y ejecución del proyecto

Como en el apartado anterior la instalación de la aplicación es sencillo gracias al uso de *Docker* y tambien garantiza que se va a comportar igual independientemente del equipo donde se instale. Siguiendo las instrucciones del archivo `README.txt` que a continuación se enumeran, se realiza la instalación:

1. Instalación de *Docker*.
2. Instalación de *Docker Compose*.
3. Clonar el repositorio.
4. Ejecutar en el directorio `SurveyingPointCode/`:
 - Si se desea solo instalar a aplicación *SurveyingPointCode*:
`sh app_install.sh`
 - Si se desea instalar ademas la aplicación *PgAdmin4*:
`sh app_install_pgadmin.sh`
5. Acceder en el navegador a la dirección <http://0.0.0.0:5000> para *SurveyingPointCode*:
6. Acceder en el navegador a la dirección <http://0.0.0.0:80> para *PgAdmin 4*:

D.5. Pruebas del sistema

Para comprobar el funcionamiento de las funciones principales de la aplicación se han realizado una serie de test unitarios.

Se ha utilizado la biblioteca de python `unittest`.

Caso de prueba CP-01

Descripción: Comprueba si el número de capas creadas en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `create_layers()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida Esperada	Ok
<code>test_create_layers_number()</code>	¿Es igual?	8	Sí
	¿No es igual?	0	Sí

Tabla D.1: Caso de prueba CP-01.

Caso de prueba CP-02

Descripción: Comprueba el nombre de las capas creadas es correcto una vez cargado y procesado un archivo de campo, con la función `create_layers()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida Esperada	Ok
<code>test_create_layers_types()</code>	¿Contiene?	Sí	Sí

Tabla D.2: Caso de prueba CP-02.

Caso de prueba CP-03

Descripción: Comprueba si el número de puntos insertados en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_points()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida Esperada	Ok
<code>test_create_points_number_file_correct()</code>	¿Es igual?	42	Sí
	¿No es igual?	0	Sí

Tabla D.3: Caso de prueba CP-03.

Caso de prueba CP-04

Descripción: Comprueba si el número de textos insertados en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_points()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida Esperada	Ok
<code>test_create_points_texts_file_correct()</code>	¿Es igual?	126	Sí
	¿No es igual?	0	Sí

Tabla D.4: Caso de prueba CP-04.

Caso de prueba CP-05

Descripción: Comprueba si el número de círculos insertados en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_circles()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida Esperada	Ok
<code>test_create_circles_number()</code>	¿Es igual?	2	Sí
	¿No es igual?	0	Sí

Tabla D.5: Caso de prueba CP-05.

Caso de prueba CP-06

Descripción: Comprueba si el número de *splines* insertados en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_splines()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida Esperada	Ok
<code>test_create_splines_number()</code>	¿Es igual?	1	Sí
	¿No es igual?	0	Sí

Tabla D.6: Caso de prueba CP-06.

Caso de prueba CP-07

Descripción: Comprueba si el número de líneas insertadas en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_lines()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida	Ok
		Esperada	
<code>test_create_lines_number()</code>	¿Es igual?	5	Sí
	¿No es igual?	0	Sí

Tabla D.7: Caso de prueba CP-07.

Caso de prueba CP-08

Descripción: Comprueba si el número de cuadrados insertados en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_squares()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida	Ok
		Esperada	
<code>test_create_squares_number()</code>	¿Es igual?	5	Sí
	¿No es igual?	0	Sí

Tabla D.8: Caso de prueba CP-08.

Caso de prueba CP-09

Descripción: Comprueba si el número de rectángulos insertados en un modelo de dibujo es correcto una vez cargado y procesado un archivo de campo, con la función `get_rectangles()`.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida	Ok
		Esperada	
<code>test_create_squares_number()</code>	¿Es igual?	1	Sí
	¿No es igual?	0	Sí

Tabla D.9: Caso de prueba CP-09.

Caso de prueba CP-10

Descripción: Comprueba que un archivo sin círculos no crea ningún círculo en un modelo de dibujo una vez cargado y procesado un archivo de campo, con la función `get_circles()`.

Datos de entrada: `Example_topographic_2.txt`

Función	Condición	Salida Esperada	Ok
<code>test_not_create_circles()</code>	¿Es igual?	0	Sí

Tabla D.10: Caso de prueba CP-10.

Caso de prueba CP-11

Descripción: Comprueba que un archivo sin *splines* no crea ninguna *spline* en un modelo de dibujo una vez cargado y procesado un archivo de campo, con la función `get_splines()`.

Datos de entrada: `Example_topographic_2.txt`

Función	Condición	Salida Esperada	Ok
<code>test_not_create_splines()</code>	¿Es igual?	0	Sí

Tabla D.11: Caso de prueba CP-11.

Caso de prueba CP-12

Descripción: Comprueba que un archivo sin líneas no crea ninguna línea en un modelo de dibujo una vez cargado y procesado un archivo de campo, con la función `get_lines()`.

Datos de entrada: `Example_topographic_2.txt`

Función	Condición	Salida Esperada	Ok
<code>test_not_create_lines()</code>	¿Es igual?	0	Sí

Tabla D.12: Caso de prueba CP-12.

Caso de prueba CP-13

Descripción: Comprueba que un archivo sin cuadrados ni rectángulos, no los crea en un modelo de dibujo una vez cargado y procesado un archivo de campo, con la función `get_lines()`.

Datos de entrada: `Example_topographic_2.txt`

Función	Condición	Salida Esperada	Ok
<code>test_not_create_squares_rectangles()</code>	¿Es igual?	0	Sí

Tabla D.13: Caso de prueba CP-13.

Caso de prueba CP-14

Descripción: Comprueba que use calculan correctamente el acimut y la distancia entre dos puntos, a y b, con la función `calculate_azimut_distance()`

Datos de entrada: `a = [1, (0, 0, 0), 'E']` y `b = [2, (100, 100, 0), 'E']`

Función	Condición	Salida Esperada	Ok
<code>test_azimut_distance()</code>	¿Es igual?	az=45° dist=141.4213	Sí
	¿No es igual?	az=50° dist=145	Sí

Tabla D.14: Caso de prueba CP-14.

Caso de prueba CP-15

Descripción: Comprueba que se calculan correctamente el incremento de X y de Y, entre dos puntos, conociendo el acimut y la distancia entre ellos, con la función `calculate_increment_x_y()`

Datos de entrada: `az = 45°` y `dist = 141.4213562373095`

Función	Condición	Salida Esperada	Ok
test_increment_x_y()	¿Es igual?	In_x=100 In_y=100	Sí
	¿No es igual?	In_x=150 In_y=145	Sí

Tabla D.15: Caso de prueba CP-15.

Caso de prueba CP-16

Descripción: Comprueba que se calculan correctamente un angulo, introduciendo un angulo y una distancia positiva o negativa, con la función `calculate_angle()`

Datos de entrada: az = 125° y dist =-10

Función	Condición	Salida Esperada	Ok
test_angle_direction()	¿Es igual?	35°	Sí
	¿No es igual?	215°	Sí

Tabla D.16: Caso de prueba CP-16.

Caso de prueba CP-17

Descripción: Comprueba si se extraen correctamente los símbolos de un archivo DXF subido por el usuario y también comprueba que se han extraído todos, con la función `upload_symbols_file()`

Datos de entrada: Example_simbology.dxf

Función	Condición	Salida Esperada	Ok
test_angle_direction()	¿Incluye?	'Farola', 'Arbol', 'Vertice'	Sí
	¿No incluye?	'Casa', 'Banco'	Sí
test_angle_direction()	¿Es igual?	3	Sí
	¿No es igual?	0	Sí

Tabla D.17: Caso de prueba CP-17.

Caso de prueba CP-18

Descripción: Comprueba el número de líneas o puntos que contiene un archivo de campo, con la función `upload_topographical_file()`

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida	
		Esperada	Ok
<code>test_number_lines_topographical_file()</code>	¿Es igual?	42	Sí
	¿No es igual?	0	Sí

Tabla D.18: Caso de prueba CP-18.

Caso de prueba CP-19

Descripción: Comprueba el número de líneas o puntos que contiene un archivo de campo que está vacío, con la función `upload_topographical_file()`

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida	
		Esperada	Ok
<code>test_number_lines_topographical_file()</code>	¿Es igual?	0	Sí
	¿No es igual?	5	Sí

Tabla D.19: Caso de prueba CP-19.

Caso de prueba CP-20

Descripción: Comprueba que el archivo DXF a generar tiene la versión correcta de CAD.

Datos de entrada: `Example_topographic_1.txt`

Función	Condición	Salida	
		Esperada	Ok
<code>test_version_cad()</code>	¿Es igual?	AC1018	Sí
	¿No es igual?	AC1019	Sí

Tabla D.20: Caso de prueba CP-20.

Documentación de usuario

E.1. Introducción

En este anexo se describen los requisitos *hardware* y *software* que debe cumplir cualquier usuario que desee hacer uso de la aplicación, así como un manual de uso de la aplicación. Además, se ha realizado un vídeo-tutorial que se puede consultar completo en:

<https://www.youtube.com/playlist?list=PLM0y-okbdA0Gkw9JrtwosMJ5bfMa3aNX8>

E.2. Requisitos de usuarios

El objetivo final de la aplicación es que sea una aplicación Web, desplegada en un servidor accesible vía Internet. El usuario solo deberá disponerse un equipo con un navegador actualizado y conexión a Internet. Se ha testado el funcionamiento de la aplicación en los siguientes navegadores:

- *Google Chrome* Versión 74.0.3729.169 (Build oficial) (64 bits)
- Microsoft Edge 42.17134.1.0
- Firefox Quantum 67.0 (64-bits)

Otra opción es si el usuario desea tenerlo instalado en su equipo, en este caso solo necesita disponer de las herramientas *Docker* y *Docker Compose* instaladas. Hay que mencionar que *Docker* no funciona en la versión de *Windows 10 Home Edition*, no es compatible con la tecnología de virtualización *Hyper-V*. Existe la posibilidad de usar la herramienta *Docker Toolbox*¹, pero no se garantiza su correcto funcionamiento en todos los casos. *Docker* funciona perfectamente con las versiones *Windows 10 Pro* y *Windows 10 Enterprise*.

¹*Docker Toolbox*: https://docs.docker.com/toolbox/toolbox_install_windows/

En *Linux* y *Mac*, su funcionamiento es correcto.

E.3. Instalación

Si se desea instalar en un equipo, hay que seguir las instrucciones detalladas en el apartado, **D.4 Compilación, instalación y ejecución del proyecto**, en la página 47. También se pueden consultar en el archivo README.md del repositorio de *GitHub*.

E.4. Manual del usuario

En esta sección se describe el uso de las diferentes funcionalidades de la aplicación.

Acceso a la aplicación.

Cuando se accede a la aplicación se nos informa del uso de las *cookies* mediante un mensaje.

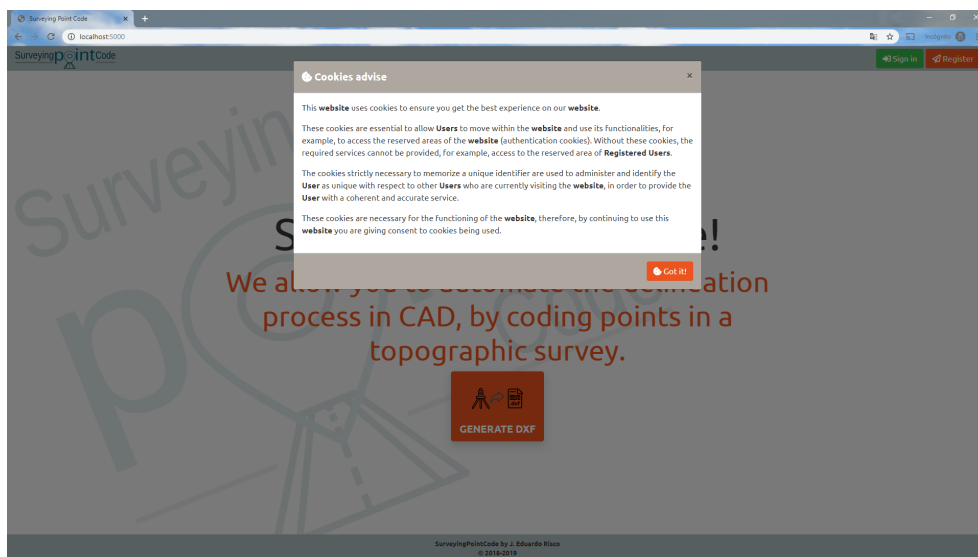


Figura E.1: Mensaje de información *cookies*.

Cerramos el mensaje y tenemos pantalla de bienvenida de la aplicación. En ella tenemos varias posibilidades:

- Si no estamos *logeados*, podemos acceder a la pantalla de *login* pulsando el botón **Sign in**.

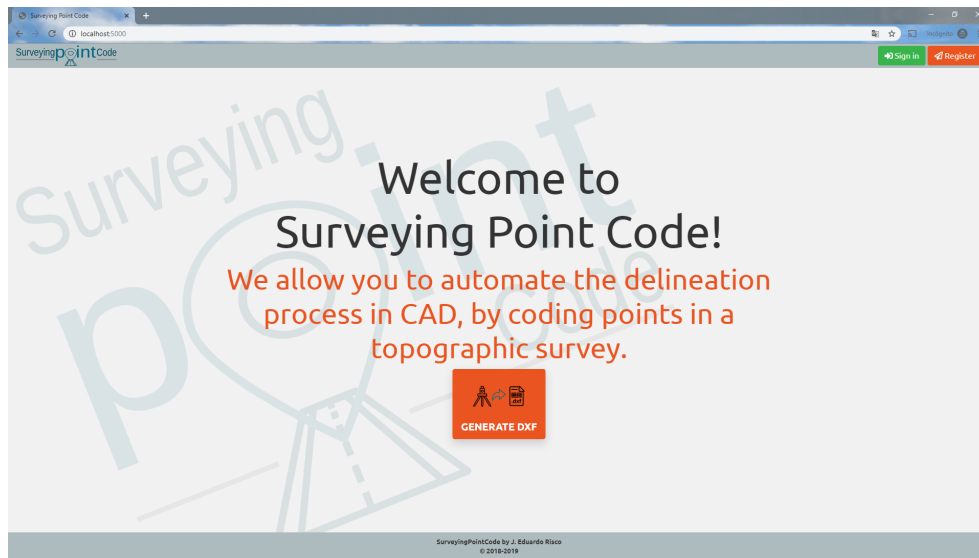


Figura E.2: Pantalla de bienvenida.

- Si no estamos *logeados*, podemos acceder a la pantalla de registro pulsando el botón **Register**.
- Si pulsamos el botón **GENERATE DXF** y no estamos *logeados*, accedemos a la pantalla de *login* y si estamos *logeados* accedemos a la pantalla de *Upload files*.

Registro de usuario.



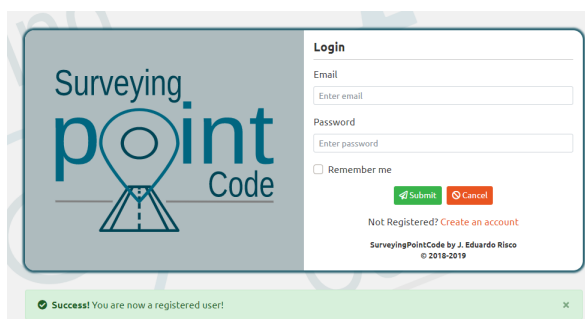
The screenshot shows the 'Register' form of the SurveyingPointCode application. On the left is a logo with the text 'Surveying point Code' and a stylized surveying instrument. The form on the right has the title 'Register' and contains five input fields: 'Username' (placeholder: 'Enter username'), 'Email' (placeholder: 'Enter email'), 'Password' (placeholder: 'Enter password'), and 'Repeat Password' (placeholder: 'Repeat password'). Below the fields are two buttons: a green 'Register' button and a red 'Cancel' button. At the bottom, it says 'SurveyingPointCode by J. Eduardo Risco © 2018-2019'.

Figura E.3: Registro de usuario.

Pasos para registrarse:

1. Introducir nombre de usuario.
2. Introducir e-mail.
3. Introducir contraseña.
4. Repetir contraseña.
5. Pulsar el botón **Register**.

Una vez completado el registro, la aplicación irá a la pantalla de *login*, mostrando un mensaje informando que el registro se ha realizado con éxito.



The screenshot shows the 'Login' form of the SurveyingPointCode application. On the left is the same logo as in Figure E.3. The form on the right has the title 'Login' and contains two input fields: 'Email' (placeholder: 'Enter email') and 'Password' (placeholder: 'Enter password'). Below the fields is a 'Remember me' checkbox. At the bottom are two buttons: a green 'Submit' button and a red 'Cancel' button. Below the buttons, it says 'Not Registered? [Create an account](#)'. At the very bottom, it says 'SurveyingPointCode by J. Eduardo Risco © 2018-2019'. A green success message banner at the bottom of the screen reads 'Success! You are now a registered user!' with a close button (X) on the right.

Figura E.4: Registro con éxito.

Errores en el registro:

No se completará el registro y mostrará mensajes de error en los siguientes casos:

- El nombre de usuario ya existe.
- El e-mail ya existe.
- El e-mail no tiene un formato correcto de e-mail.
- No coinciden las dos contraseñas introducidas.
- No se hayan completado todos los campos requeridos.

Otras opciones:

- Para salir de esta pantalla sin registrarse, pulsar el botón **Cancel**.

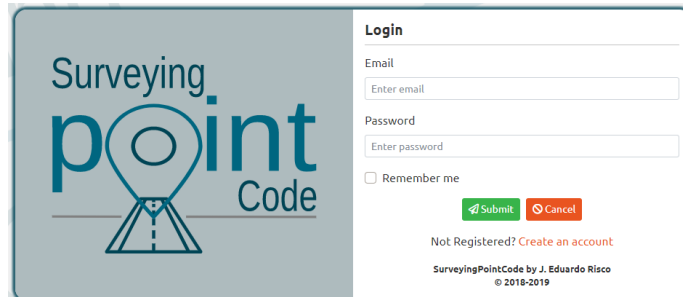
Login de usuario.**Pasos para *logearse*:**

Figura E.5: *Login* de usuario.

1. Introducir e-mail.
2. Introducir contraseña.
3. Pulsar el botón **Submit**.

Una vez *logado*, la aplicación irá a la pantalla de *Upload files*.

Errores en el *login*:

No se completará el *login* y mostrará mensajes de error en los siguientes casos:

- El nombre de usuario sea incorrecto.
- El e-mail sea incorrecto.
- No se hayan completado todos los campos requeridos.

Otras opciones:

- Para salir de esta pantalla sin *logearse*, pulsar el botón **Cancel**.
- Para ir a la pantalla de registro, pulsar en el *link* **Create an account**.

Carga de archivos.

El usuario tiene la opción de cargar 3 tipos de archivos: archivo topográfico, archivo de configuración de la conversión y archivo con símbolos. En los 3 casos el usuario puede buscar el archivo mediante el explorador que se abre al pulsar o arrastrar el archivo hasta la caja.

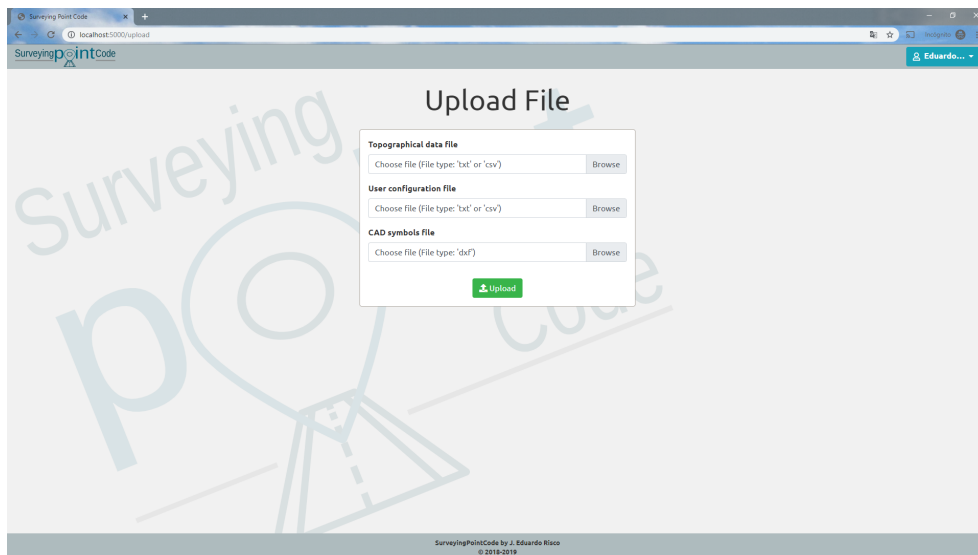


Figura E.6: Carga de archivos.

Carga de archivo topográfico.

El archivo topográfico es **obligatorio**. El archivo estará compuesto por una o varias líneas, cada línea equivale a un punto medido en campo. El punto obligatoriamente estará formado por los siguientes campos, separados por comas:

- Número de punto.
- Coordenada *X*.
- Coordenada *Y*.
- Coordenada *Z*.
- Código del punto.

La interpretación de la codificación se explica con detalle en el apartado «3.3. Codificación de los puntos», de la memoria.

A continuación vemos un ejemplo de un archivo:

```
1,355776.180,4611015.011,691.055,E I
2,355773.203,4611028.546,691.055,E
3,355781.359,4611129.076,691.055,TR REG
4,355786.052,4611135.542,691.055,TC TEL
5,355783.215,4611143.179,691.055,TX 2 SAN
6,355754.037,4611145.893,691.055,A IC
7,355755.345,4611150.953,691.055,A C
8,355857.822,4611095.993,691.055,E -14.1 20.5 -25.75
```

Pasos para cargar un archivo topográfico:

1. Elegir un archivo topográfico.
2. Pulsar el botón *Upload*

Si el archivo es correcto, la aplicación irá a la pantalla *Convert File to DXF*.

Errores en la carga del archivo:

La carga no será correcta si al pulsar el botón *Upload*, se presentan los siguientes errores:

- El archivo no tiene el formato correcto. En el mensaje aparecen las líneas del archivo donde están los errores. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

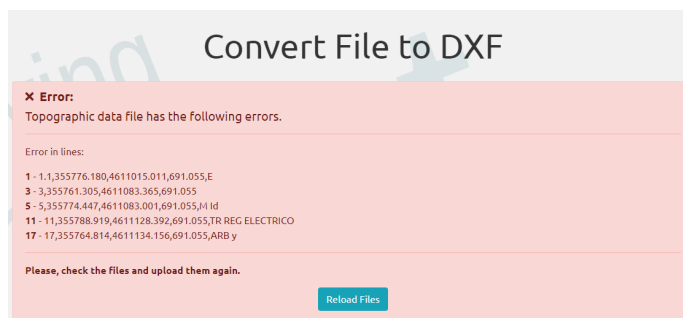


Figura E.7: Error formato de archivo topográfico, *parser*.

- El archivo está vacío. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

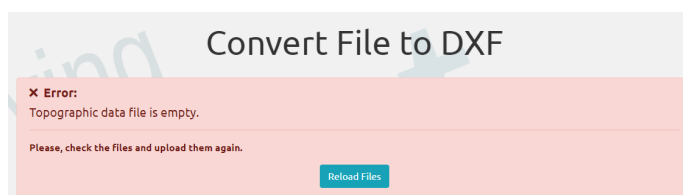


Figura E.8: Error archivo topográfico vacío.

- El número de puntos para definir cuadrados con el código «TC» no es un número par, por lo que no se puede garantizar la generación correcta de estos elementos. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

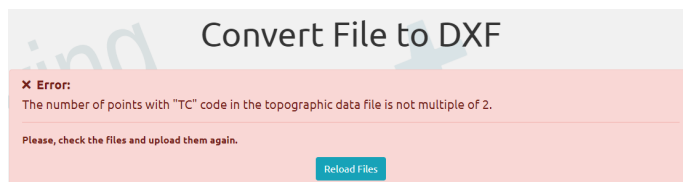


Figura E.9: Error archivo topográfico, dibujo de cuadrados.

- El número de puntos para definir rectángulos con el código «TR» no es múltiplo de 3, por lo que no se puede garantizar la generación correcta de

estos elementos. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

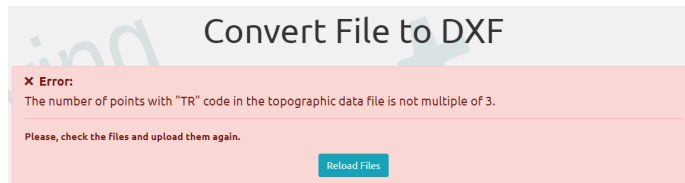


Figura E.10: Error archivo topográfico, dibujo de rectángulos.

Carga de archivo de configuración de la conversión.

El archivo de configuración de la conversión **no es obligatorio** para generar un archivo DXF a partir de un levantamiento topográfico. El archivo estará compuesto por una o varias líneas, cada línea equivale a una configuración para un código de campo. Cada línea o configuración estará formado por los siguientes campos, unos obligatorios y otro opcional, separados por comas:

- Código topográfico.
- Capa de CAD.
- Color.
- Símbolo. Este campo es opcional

A continuación vemos un ejemplo de un archivo:

```
CA,Cañada,(255,0,0)
VA,Valla,(255,255,0)
ARB,Arbol,(0, 255, 127),Arbol
P,Hidrología,(0,127,255),Pozo
L,Hidrología,(0,127,255)
E,Edificio,(255, 0,255)
T0,Torreta,(0,0,0),Torreta_Media
EXD,Expropiación,(0,0,0)
EXI,Expropiación,(0,0,0)
CAD,Canal,(0,0,255)
CAI,Canal,(0,0,255)
LEM,Línea_Eléctrica_Media,(255,63,0),Torreta_Media
LEA,Línea_Eléctrica_Alta,(204,204,102),Torreta_Alta
SAN,Saneamiento,(255, 0, 255)
AC,Acera,(255,255,0)
```

Pasos para cargar un archivo de configuración de la conversión:

1. Debe haber un archivo topográfico seleccionado.
2. Elegir un archivo de configuración de la conversión.
3. Pulsar el botón *Upload*

Si el archivo es correcto, la aplicación irá a la pantalla *Convert File to DXF*.

Errores en la carga del archivo:

La carga no será correcta si al pulsar el botón *Upload*, se presentan los siguientes errores:

- El archivo no tiene el formato correcto. En el mensaje aparecen las líneas del archivo donde están los errores. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

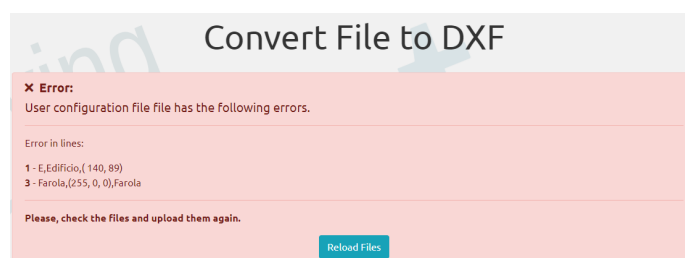


Figura E.11: Error formato de archivo configuración de la conversión, *parser*.

- El archivo está vacío. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

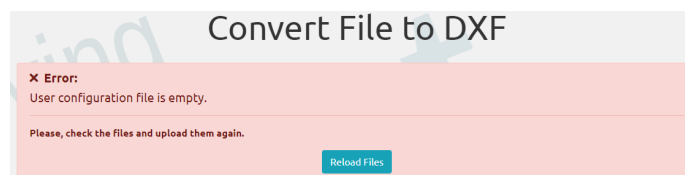


Figura E.12: Error archivo configuración de la conversión vacío.

- Un mismo tipo de código topográfico, aparece repetido en diferentes líneas. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

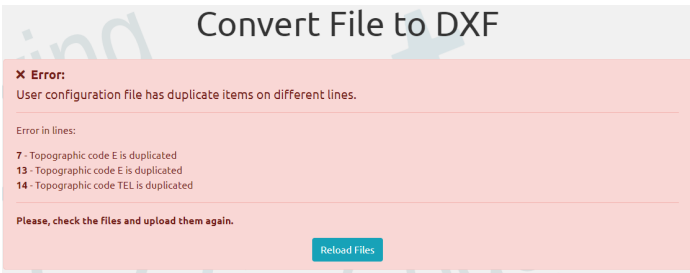


Figura E.13: Error archivo configuración de la conversión, códigos repetidos.

- Los errores siguientes permiten al usuario corregir estos a través de la interfaz, sin necesidad de volver a cargar el archivo:
 1. Una capa con el mismo nombre, tiene diferentes colores asignados. El mensaje de error muestra la capa que es errónea.
 2. El archivo de configuración de la conversión contiene colores que no pertenecen a la paleta de CAD. El mensaje de error muestra la capa que es y el color asignado erróneo.

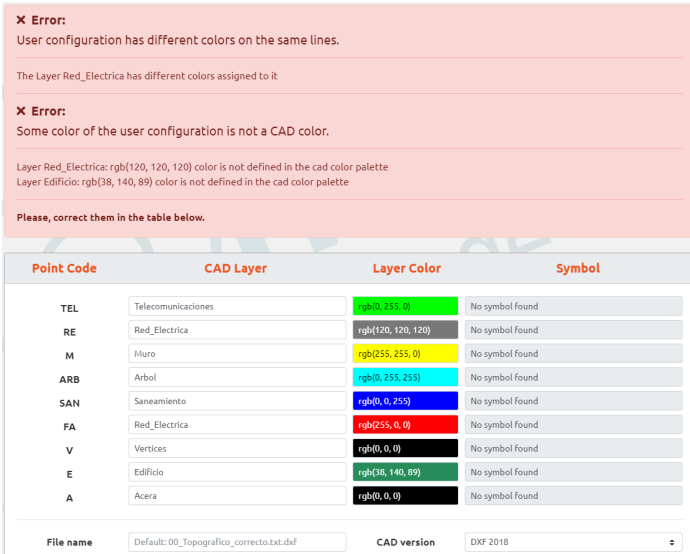


Figura E.14: Errores corregibles a través de la interfaz.

Carga de archivo de símbolos.

El archivo de símbolos **no es obligatorio** para generar un archivo DXF a partir de un levantamiento topográfico. El archivo estará compuesto por uno o varios bloques.

1. Debe haber un archivo topográfico seleccionado.
2. Elegir un archivo de símbolos.
3. Pulsar el botón *Upload*

Si el archivo es correcto, la aplicación irá a la pantalla *Convert File to DXF*.

Errores en la carga del archivo:

La carga no será correcta si al pulsar el botón *Upload*, se presentan los siguientes errores:

- El archivo está vacío. Puede ser un archivo con formato DXF correcto, pero **no contener ninguna entidad bloque**, por lo que la aplicación no podrá interpretar los símbolos. Se da la opción de volver a cargar el archivo, pulsando el botón *Reload Files*.

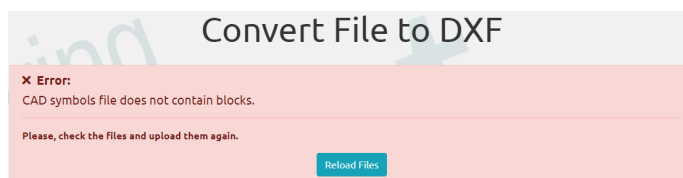


Figura E.15: Error archivo de símbolos, no contiene bloques.

Conversión a DXF.

Pasos para realizar la conversión a un archivo DXF:

1. El usuario podrá introducir o modificar, los nombres de las capas, colores y símbolos (si se ha cargado un archivo con simbología), a través de la interfaz.

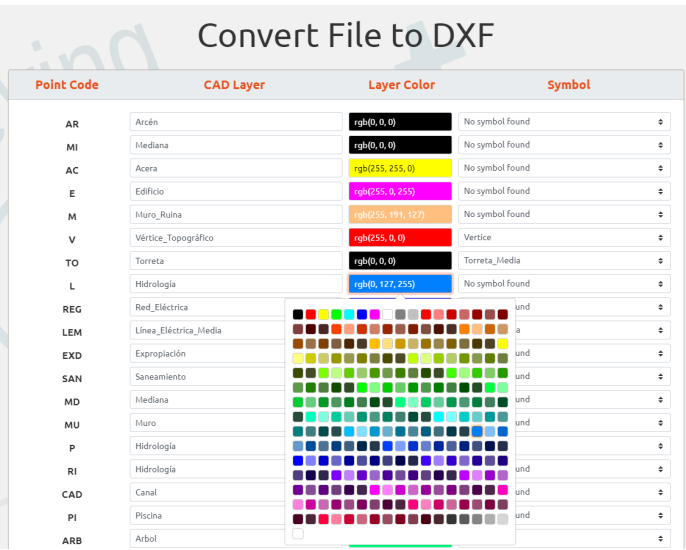


Figura E.16: Elección de colores.

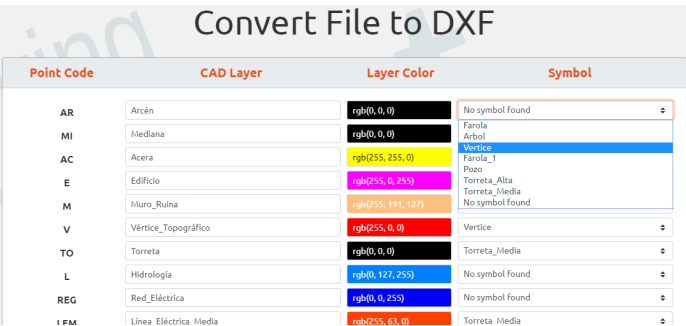


Figura E.17: Elección de símbolos.

2. El usuario podrá asignar un nombre al archivo DXF generado, introduciendo el nombre en la caja *File name*. Si no le asigna un nombre, se le asignará por defecto el nombre del archivo topográfico cargado.
3. El usuario podrá elegir la versión de CAD que desea que tenga el DXF generado. Si no le asigna ninguna, se le asignará por defecto la versión CAD 2018 (AC1032).

The screenshot shows a web interface for converting a file to a specific CAD version. It features a table for mapping layers from the source file to the target CAD version. Below the table, there are fields for 'File name' and 'CAD version', and a 'Convert' button.

Layer	Source Layer	Target Layer
LEA	Linea_Eléctrica_Alta	Torreta_Alta
PA	Parcela	No symbol found
FA	Farolas	Farola_1
EXI	Expropiación	No symbol found
CAI	Canal	No symbol found

File name: Default: Topografico_completo.txt.dxf

CAD version: DXF 2018, DXF 2018, DXF 2013, **DXF 2010**, DXF 2007, DXF 2004

Convert

Figura E.18: Elección de versión de CAD.

4. Por último, el usuario pulsará el botón **Convert** y si todo es correcto, la aplicación irá a la pantalla *Download File* y mostrará un mensaje informando que la conversión se ha realizado con éxito.

The screenshot shows the 'Download File' screen. At the top, there is a green success message: 'Success! File successfully converted!!!'. Below this, there is a section titled 'List of converted files' which contains a table with one file: 'Parcela_15.dxf'. There are two buttons: 'Download all' and 'One more'.

Download File

Success! File successfully converted!!!

List of converted files

Download all One more

Parcela_15.dxf

Figura E.19: Conversión con éxito.

Errores en la conversión del archivo:

La conversión no se podrá realizar si al pulsar el botón *Convert*, se presentan los siguientes errores:

1. Una capa con el mismo nombre, tiene diferentes colores asignados. El mensaje de error muestra la capa que es errónea.
2. El archivo de configuración de la conversión contiene colores que no pertenecen a la paleta de CAD. El mensaje de error muestra la capa que es y el color asignado erróneo.

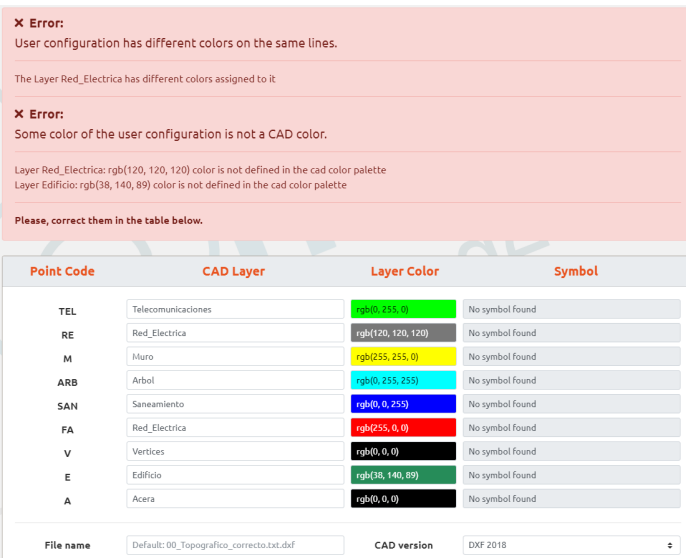


Figura E.20: Errores corregibles a través de la interfaz.

Descarga de archivos.

Pasos para realizar la descarga de archivos:

1. Si el usuario desea descargar un archivo de forma individual, pulsará sobre el nombre del archivo que aparece en la lista de archivos convertidos. A continuación el archivo se descargará a su equipo.
2. Si el usuario desea descargar varios archivos a la vez, en formato comprimido, pulsará sobre el botón *Download all*. A continuación el archivo comprimido se descargará a su equipo.

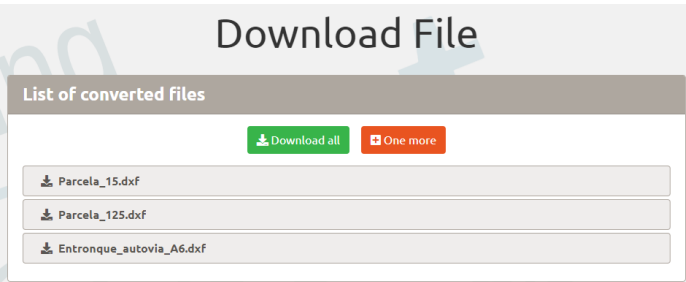


Figura E.21: Descarga de archivos.

Otras opciones:

- El usuario puede seguir convirtiendo archivos pulsando el botón *One more*.

Logout.

Cerrar la sesión de usuario:

- El usuario puede cerrar la sesión pulsando el botón *Sign out*, situado en el menú desplegable de la barra de navegación superior. La sesión de usuario se cerrará y la aplicación irá a la pantalla de bienvenida.



Figura E.22: *Logout*

- Una vez *logueado* el usuario, puede cerrar su sesión desde cualquier pantalla de la aplicación.
- La sesión de usuario se cerrará tras 5 minutos de inactividad en la aplicación y volverá a la pantalla de *login*

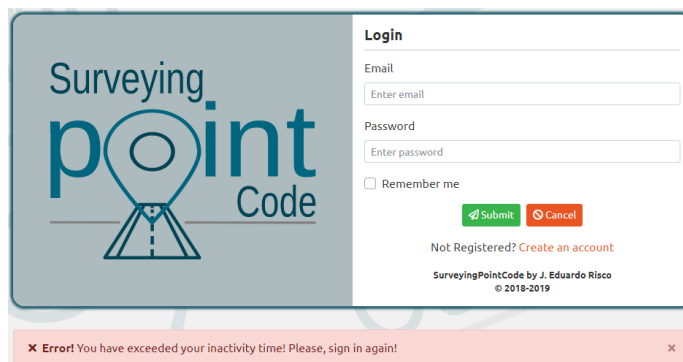


Figura E.23: *Cierre de sesión por inactividad*

E.5. Preguntas Frecuentes

- **¿Es necesario registrarse en la aplicación?** Sí, es necesario registrarse para poder usar la aplicación.
- **¿Se pueden convertir archivos sin códigos?** No, el archivo debe contener códigos. Todos los puntos deben tener asociado un código. La aplicación mostrará un error.
- **¿Es necesario cargar un archivo de configuración de la conversión para generar un archivo DXF?** No, no es necesario. Se puede configurar la conversión a través de la interfaz.
- **¿Es necesario cargar un archivo de símbolos para generar un archivo DXF?** No, no es necesario. El archivo generado no contendrá ningún símbolo.
- **¿Es necesario configurar una conversión, asociando capas, colores y símbolos, para generar un archivo DXF?** No, no es necesario. Si no se definen asociaciones, el archivo generado guardará todos los elementos por defecto en la capa **Layer 0**, con color **RGB(0,0,0)** y sin ningún símbolo asociado.
- **¿Es necesario dar un nombre al archivo DXF a generar?** No, no es necesario. Si no se le asigna nombre, el archivo generado se guardará por defecto con el nombre del archivo topográfico.
- **¿Es necesario seleccionar una versión de CAD para generar el archivo DXF?** No, no es necesario. Si no se selecciona una versión, el archivo generado se creará por defecto con la versión **CAD 2018**.
- **¿Es posible asociar diferentes códigos topográficos a una misma capa de CAD?** Sí, por ejemplo los códigos: P, CA y RI, que pueden significar: pozo, canal y río, se pueden asociar a la única capa Hidrología.
- **¿Es posible asociar el mismo código topográfico a diferentes capas de CAD?** No, no es posible. La aplicación mostrará un error.
- **¿Es posible asociar distintos colores a al misma capa de CAD?** No, no es posible. La aplicación mostrará un error.
- **¿Es posible asociar cualquier color a una capa de CAD?** No, no es posible. Solo los colores de la paleta CAD. La aplicación mostrará un error.

Los colores disponibles se pueden consultar en: <http://gohtx.com/acadcolors.php>

- **Si se nos olvida cerrar la sesión. ¿Nuestra sesión permanecerá permanentemente abierta?** No, la sesión se cerrará por seguridad, al cabo de 5 minutos de inactividad.

Bibliografía

- [1] Borbón Alexander and Mora Walter. *Edición de Textos Científicos LaTEX 2017*. Escuela de Matemática, Instituto Tecnológico de Costa Rica, 2017. ISBN 978-9977662275.
- [2] andrearrs. Cómo elegir la licencia correcta para tu proyecto open source, 2014. URL <https://hipertextual.com/archivo/2014/05/como-elegir-licencias-open-source/>. [Online; Accedido 17-Mayo-2019].
- [3] Creative Commons. Attribution-noncommercial-sharealike 4.0 international (cc by-nc-sa 4.0). URL <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en>. [Online; Accedido 18-Mayo-2019].
- [4] Pablo de la Cuesta. Desarrollando con docker. URL <https://medium.com/@pablodelacuesta/desarrollando-con-docker-ae9e93402ddd>. [Online; Accedido 19-Mayo-2019].
- [5] Ministerio de Trabajo Migraciones y Seguridad Social. Régimen general de la seguridad social, 2019. URL <http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537>. [Online; Accedido 11-Mayo-2019].
- [6] Jefatura del Estado. Ley orgánica 3/2018, de 5 de diciembre, de protección de datos personales y garantía de los derechos digitales., 2018. URL <https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673>. [Online; Accedido 23-Mayo-2019].
- [7] ARCOVÍA DIGITAL. Asistencia técnica por horas., 2019. URL <https://www.arcovia.com/asistencia-tecnica/modificaciones-por-horas>. [Online; Accedido 24-Mayo-2019].

- [8] Roger Dudler. git - la guía sencilla, 2018. URL <https://rogerdudler.github.io/git-guide/index.es.html>. [Online; Accedido 12-Enero-2019].
- [9] Python Software Foundation. unittest — unit testing framewor, 2019. URL <https://docs.python.org/3/library/unittest.html>. [Online; Accedido 22-Marzo-2019].
- [10] Moses Kim. License compatibility, 2017. URL <https://medium.com/shakuro/software-licenses-explained-77f4f18eb1>. [Online; Accedido 17-Mayo-2019].
- [11] Holger Krekel et al. pytest, 2017. URL <https://docs.pytest.org/en/latest/>. [Online; Accedido 22-Marzo-2019].
- [12] Diego C Martín. Tutorial de git. manual básico con ejemplos, 2018. URL <https://www.diegocmartin.com/tutorial-git/>. [Online; Accedido 12-Enero-2019].
- [13] Ocellum Consultoría TIC. Tarifa general de precios., 2019. URL <https://ocellum.net/tarifa-general-de-precios/>. [Online; Accedido 24-Mayo-2019].
- [14] Agencia Tributaria. Retenciones e ingresoos a cuenta del irpf en el ejercicio 2019, 2019. URL https://www.agenciatributaria.es/AEAT.internet/Inicio/La_Agencia_Tributaria/Campanas/Retenciones/Cuadro_informativo_tipos_de_retencion_aplicables__2019_.shtml. [Online; Accedido 11-Mayo-2019].



Esta obra está bajo una licencia Creative Commons Reconocimiento 4.0 Internacional ([CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)).