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 la Universidad de Burgos — 9 de mayo de 2019
Tutores: Dr. César García-Osorio
y Dr. Carlos López Nozal

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. César García-Osorio y D. Carlos López Nozal, profesores del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Exponen:
Que el alumno D. José Eduardo Risco Sánchez-Cortés, con DNI 40441256B , ha realizado el Trabajo final de Grado en Ingeniería Informática titulado: "SurveyingPointCode - Automatización del proceso de delineación a partir de datos de un Levantamiento Topográfico"

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 9 de mayo de 2019

Vº. Bº. del Tutor:

D. César García-Osorio

Vº. Bº. del tutor:

D. Carlos López Nozal

Resumen

En el campo de la Topografía, hay trabajos cuyo objetivo es obtener el plano de un levantamiento topográfico. Este trabajo se divide en dos partes, trabajo de campo y delineación. Es importante entender que la relación entre las dos partes es fundamental para poder reducir al máximo los recursos invertidos tanto en la parte del trabajo de campo como en la de delineación.

Con este trabajo, se pretende que mediante una codificación definida por el usuario en el trabajo de campo, (para cada punto medido), se optimice la medición en campo y posteriormente, la codificación permita la generación automática del plano final, en un archivo de **CAD** (con formato **DXF**), teniendo en cuenta la distribución de los puntos en diferentes capas, generación de líneas, curvas, simbología, etc.

Con este propósito se ha desarrollado la aplicación Surveying-PointCode, aplicación Web que realiza la conversión de un archivo que contiene los datos de un levantamiento topográfico a un archivo **DXF**.

Descriptores

Levantamiento topográfico, CAD, DXF, código punto, delineación, aplicación Web.

Abstract

In the field of Topography, there are projects whose objective is to obtain the plan of a topographic survey. This project is divided in two parts; survey work and delineation. It is important to understand that the relationship between these two parts will be fundamental to be able to reduce to the maximum the resources invested in both the survey work and the delineation work.

The main purpose of this work is to optimize the data collection through a coding defined by the user in the field work (for each measured point). Subsequently the coding will allow the automatic generation of the final plan, in a **CAD** file. (with **DXF** format), taking into account the distribution of points in different layers, generation of lines, curves, symbology, etc.

For this purpose, the SurveyingPointCode application has been developed, a Web application that performs the conversion of a file that contains the data from a topographic survey to a **DXF** file

Keywords

Topographical survey, CAD, DXF, point code, delineation, Web application.

Índice general

Índice de figuras

Índice de tablas

Introducción

1.1. Introducción

Uno de los trabajos que se realiza habitualmente en el campo de la Topografía y Cartografía, es la representación en un plano de los datos obtenidos en un levantamiento topográfico. Pongamos como ejemplo que se nos pide como producto final, un plano basado en un levantamiento topográfico de una zona de una ciudad para realizar el estudio de unas futuras obras. El plano debe contener todos los elementos existentes que sean de interés para este estudio, como, por ejemplo: edificios, aceras, viales, redes de saneamiento, abastecimiento, eléctricas, mobiliario urbano, etc.

Este trabajo se realizará en dos fases; la primera, el trabajo de campo, donde se realiza el levantamiento topográfico, adquisición de datos, que consiste en la medición georreferenciada de todos los elementos que sean de interés y que deben aparecer en el plano. La segunda fase es la de delineación, en la que se 'dibuja' el plano. Hay que aclarar que, en la fase del levantamiento topográfico, solo se miden puntos, y es en la fase de delineación donde tenemos que crear todos los elementos, como líneas, curvas, símbolos, etc

Un problema importante al que nos enfrentamos es la gran cantidad de tiempo y recursos que invertimos en gestionar los datos obtenidos en campo para obtener un plano, alargando en exceso el desarrollo y entrega de un proyecto. Con este planteamiento surge la necesidad de crear una aplicación que consiga automatizar este trabajo el máximo posible y así obtener un ahorro de tiempo y de recursos, fundamental en nuestros proyectos. Puede parecer a priori un trabajo sencillo y rápido de realizar, con una herramienta de CAD, uniendo puntos manualmente para crear líneas, u otros elementos, claro está, si tienes un levantamiento topográfico con 30 puntos y recuerdas

como lo has hecho en campo, pero cuando tienes un levantamiento con 1000 puntos, e intervienen varias personas en él, se puede convertir en un trabajo muy complicado e insufrible de realizar. Una herramienta así no existe mercado, algunos programas como AutoCAD disponen de algún módulo a mayores del programa, que permiten hacer algo similar, pero con un resultado muy pobre respecto a lo que aquí se plantea.

La principal ventaja que aporta el uso de esta aplicación es el importante ahorro de tiempo y recursos que obtenemos en la realización de un trabajo. Por ejemplo, un trabajo de campo con 1000 puntos, con una codificación compleja, es decir con múltiples y diferentes elementos, una persona con habilidad en manejo de aplicaciones CAD, puede invertir alrededor de 2 horas. Esta aplicación pretende que ese proceso sea automático e inmediato. No debemos olvidar que el uso de una codificación totalmente aleatoria por el usuario a la hora de realizar el trabajo de campo también va a permitir obtener un importante ahorro de recursos, en esa parte del trabajo. Otras ventajas son:

- Es una aplicación web, por lo que los usuarios no necesitan instalar nada en sus equipos y pueden acceder desde cualquier lugar.
- Facilidad de uso, no hacen falta conocimientos técnicos en el uso de herramientas CAD.

1.2. Contenido del proyecto

La estructura de la memoria es la siguiente:

- **Introducción:** breve descripción del contenido del trabajo y de la estructura de la memoria.
- **Objetivos del proyecto:** objetivos que se persiguen con la realización del proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos necesarios para el desarrollo de la solución propuesta.
- **Técnicas y herramientas:** presentación de las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.
- **Aspectos relevantes del desarrollo:** descripción de los aspectos más importantes ocurridos a lo largo del desarrollo del proyecto.

- **Trabajos relacionados:** pequeño resumen de los trabajos y proyectos ya realizados en el campo del proyecto en curso.
- **Conclusiones y líneas de trabajo futuras:** resumen acerca de los conocimientos adquiridos y posibles aspectos de mejora o expansión de la solución aportada.

Además, se han desarrollado los siguientes anexos:

- **Plan de Proyecto Software:** explicación sobre la planificación temporal llevada a cabo, así como un estudio de viabilidad del proyecto.
- **Especificación de requisitos:** descripción de la fase de análisis de la aplicación.
- **Especificación de diseño:** explicación y descripción del diseño de la aplicación.
- **Documentación técnica de programación:** recoge los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
- **Documentación de usuario:** manual de usuario que permita conocer el funcionamiento la aplicación.

Por último, hay que indicar que el proyecto se encuentra disponible en el siguiente enlace:
https://github.com/EduardoRisco/SurveyingPointCode

Objetivos del proyecto

A continuación, se detallan los diferentes objetivos que han motivado la realización del proyecto.

2.1. Objetivos generales

- Desarrollar una aplicación web que permita obtener un archivo DXF a partir de los datos de un levantamiento topográfico.
- Definir un tipo de codificación para los puntos medidos en campo, que sea interpretada por la aplicación, automatice el proceso de dibujo y mejore el rendimiento en el trabajo de campo.
- Permitir al usuario crear una cuenta.
- Permitir al usuario subir : archivos de campo, archivos de configuración personalizados de la transformación y archivos DXF, con símbolos creados por el usuario.
- Permitir al usuario modificar o elegir, en la interfaz, los nombres de las capas de CAD, colores y símbolos existentes, y asociarlos a los códigos de campo
- Permitir al usuario elegir la versión de CAD para generar el archivo DXF.
- Permitir al usuario almacenar en su equipo, el archivo DXF generado.

2.2. Objetivos técnicos

- Desarrollar una aplicación web utilizando el framework **Flask**.
- Utilizar **PLY**[1] (Python Lex-Yacc), como analizador sintáctico, para reconocer la validez de los archivos de entrada.
- Utilizar **ezdxf**[2] , para leer, modificar y crear archivos DXF.
- Utilizar **Bootstrap**, para crear el sitio web.
- Utilizar **TinyColor**[3] para poder definir los colores con los estándares de AutoCAD.
- Utilizar **PostGIS**[4] como sistema de base de datos.
- Realizar test unitarios.
- Hacer uso de la herramienta **CODEBEAT** para comprobar la calidad del código.
- Desplegar la aplicación usando **Docker** .
- Utilizar **Git** como sistema de control de versiones distribuido junto con la plataforma **GitHub**.
- Aplicar la metodología ágil **Scrum** para el desarrollo del software.
- Utilizar **ZenHub** como herramienta de gestión de proyectos.

2.3. Objetivos personales

- Mejorar los recursos invertidos y facilitar el trabajo, a la hora de realizar un levantamiento topográfico.
- Realizar el proyecto como un trabajo real, aplicando los conocimientos adquiridos en la realización del Grado.
- Utilizar herramientas y metodologías demandadas en el mercado laboral; Git, desarrollo Web, Docker, etc.

[1] *PLY*: https://pypi.org/project/ply/
[2] *ezdf*: https://ezdxf.readthedocs.io/en/latest/index.html
[3] *TinyColor*: https://github.com/bgrins/TinyColor
[4] *PostGIS*: https://postgis.net/

Conceptos teóricos

En este apartado se explican conceptos los básicos necesarios para entender el contexto en que se desarrolla este proyecto, respecto a lo que aquí se pretende, haciendo hincapié en la topografía y el dibujo en CAD.

3.1. Levantamiento topográfico

Un levantamiento topográfico consiste en la medición de distintos puntos, mediante cálculos topográficos, donde se obtienen las coordenadas de estos, generalmente las coordenadas X,Y y Z .Estas coordenadas nos indican su posición exacta en el espacio.

El software que tienen los equipos de topografía; como teodolitos, estaciones totales o GPS, permiten asociar a cada punto un texto en el momento de medir el punto y así, el punto queda registrado con ese campo.

Este campo de texto, que de aquí en adelante llamaremos código de punto, es la clave de este proyecto. El algoritmo diseñado debe saber interpretar este código de punto. Como veremos más adelante, este nos puede indicar una o múltiples cosas a la vez, lo que conlleva mayor dificultad a la hora de definir el formato que debe tener este. Tiene que aportar la máxima información posible, pero a la vez tiene que ser muy sencillo en su estructura y de una longitud lo más corta posible. No es muy viable tener que estar mucho tiempo tecleando, ya que se perdería mucho tiempo y alargaría en exceso el trabajo de campo.

3.2. Archivo DXF. Elementos del dibujo

DXF (acrónimo del inglés Drawing Exchange Format) es un formato de archivo para dibujos de diseño asistido por computadora, creado por AutoCAD. Formato muy extendido y utilizado por la mayoría de programas de CAD, como por ejemplo: AutoCAD, MicroStation ,FreeCAD, DraftSight,... y también por programas de GIS, como: ArcGIS,QGIS,gvSIG,..

Para entender de forma sencilla los elementos de los que está compuesto un archivo DXF, solo se trataran aquellos elementos que se utilizan en esta aplicación, de otra forma, la explicación sería muy extensa. Simplificando, se procede a hacer una diferenciación entre los elementos que queremos dibujar que llamaremos elementos gráficos, y los elementos no gráficos.

Elementos gráficos

- **Puntos:** En este caso es el elemento básico, ya que en una medición topográfica solo se obtienen puntos. El resto de los elementos que se exponen a continuación están basados en la posición de estos puntos.
- **Líneas:** Están formadas por una sucesión de puntos, pueden ser abiertas o cerradas. Deben tener un punto inicial y otro final. Los cuadrados y rectángulos, en esta aplicación, se pueden englobar en este grupo, al ser polígonos o líneas cerradas.
- **Splines:** Curvas interpoladas en una sucesión de puntos, pueden ser abiertas o cerradas. Deben tener un punto inicial y otro final.
- **Círculos:** En esta aplicación los definiremos por un punto, que es el centro del circulo y un radio.
- **Textos:** Es un texto cuya posición en el dibujo se define, por la posición de un punto y una distancia a este.

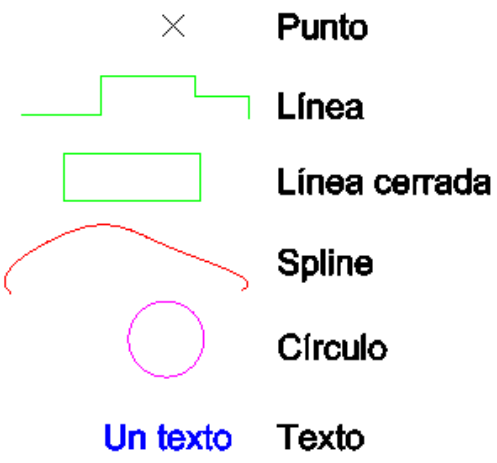

Figura 3.1: Elementos gráficos

Elementos no gráficos

- **Capas:** Sirven para organizar mejor los objetos en nuestro dibujo ayudando a un mejor manejo a la hora de mover, copiar o identificarlos en el plano. Estas son un aspecto fundamental en el desarrollo del dibujo para poder obtener un plano bien organizado y con excelente presentación. En un buen trabajo, es casi imprescindible el uso de capas.

  Las capas son una forma de clasificar objetos en CAD, si tienes un plano muy extenso con muchos objetos, la forma más fácil de identificarlos es mediante su capa, cada objeto dibujado en CAD se le asociara una capa y esa capa la puedes crear asignándole un nombre, un color, entre otras cosas. Es la mejor forma de identificar cierto objeto dentro de tu dibujo. En esta aplicación, las capas tendrán un nombre y un color.

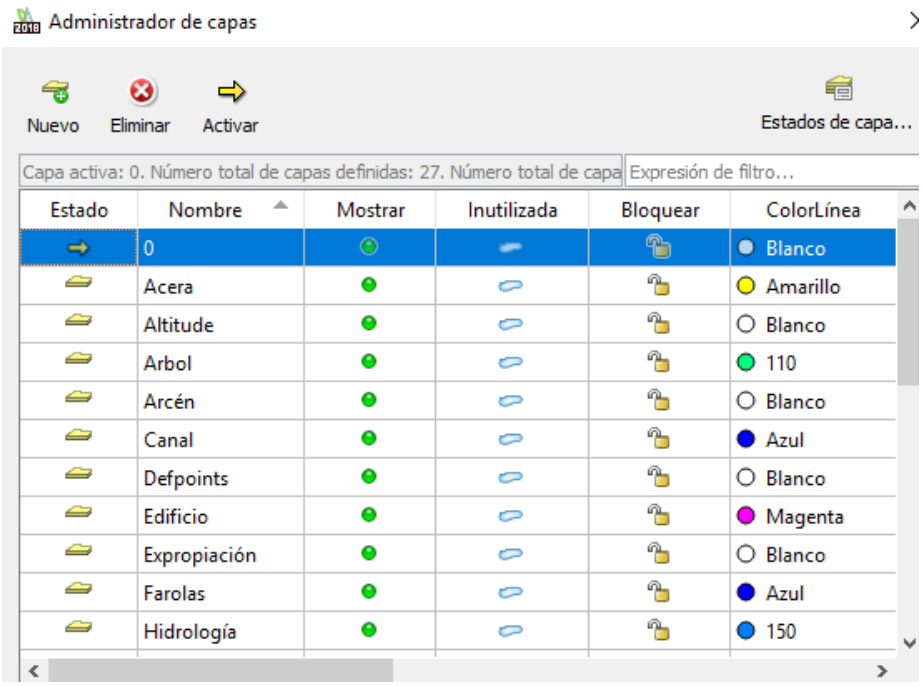

Figura 3.2: Capas en un programa de CAD

- **Bloques:** Un bloque en CAD es un conjunto de objetos (llamados entidades), agrupados como un todo. Es decir, que podemos dibujar líneas, arcos, círculos, cada uno con propiedades distintas que los demás y luego invocar un comando para "juntarlos" a todos bajo un mismo nombre y asignarle un punto de inserción.

  En esta aplicación se pretende que el usuario pueda subir un archivo DXF personalizado con sus bloques, y poder asignarlo a diferentes puntos del levantamiento topográfico a través del código de punto

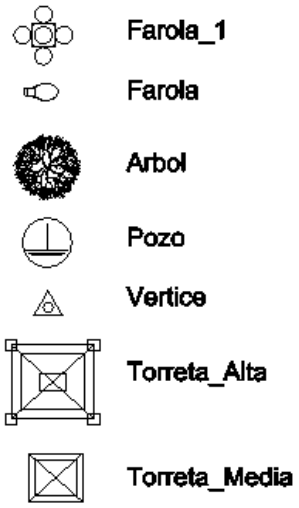

Figura 3.3: Símbolos creados como entidades bloques de CAD

3.3. Codificación de los puntos

Como se ha mencionado anteriormente, la codificación de los puntos es la parte clave para el desarrollo de este proyecto, ya que en ella se va a basar la aplicación a la hora de crear los elementos del dibujo. Debe poder crear líneas y splines, interpretando donde empiezan y acaban. Crear círculos, cuadrados y rectángulos, insertar símbolos o bloques en determinados puntos y, por último, asociar cada elemento su capa correspondiente, con su color asignado.

Todo esto, si eres un usuario con conocimientos en manejo de aplicaciones CAD, se haría de forma manual en su mayor parte. Con una buena definición del código de punto, se tratará de automatizar todo este proceso. Como ya sabemos, el código debe ser lo mas sencillo posible y a la vez aportar la máxima información posible, dejando que el usuario tenga libertad para elegir los códigos, salvo algunas partes, que deben de tener una sintaxis concreta.

A continuación, veremos algunos ejemplos de codificación para distintos elementos del dibujo:

- Se quiere dibujar la línea que define el arcén de una carretera, podría-

mos escribir en el código del punto; ‘ARCEN’, ‘ARC’, ‘A’, o lo que quisiéramos. Como se pretende que el código sea lo más sencillo posible, tomamos como mejor opción ‘A’. Para definir que la línea empieza en un punto concreto, ese texto debe ir seguido obligatoriamente del carácter ‘I’. Un ejemplo de cómo sería una línea de arcén formada por 4 puntos es: (‘A I’, ‘A’, ‘A’, ‘A’).

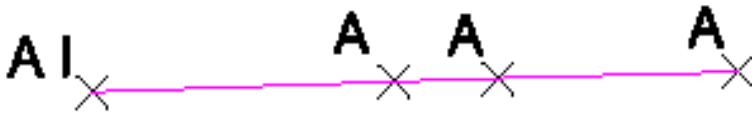

Figura 3.4: Representación de una línea

- Siguiendo con el ejemplo anterior, ahora queremos que dibuje una curva tipo spline. Para definir que la línea empieza en un punto concreto y que sea spline, ese texto debe ir seguido obligatoriamente del carácter ‘IC’ y el resto de los puntos de esa curva, el texto inicial debe ir seguido obligatoriamente del carácter ‘C’. Un ejemplo de cómo sería una línea de arcén tipo spline formada por 4 puntos es: (‘A IC’, ‘AC’, ‘AC’, ‘AC’).

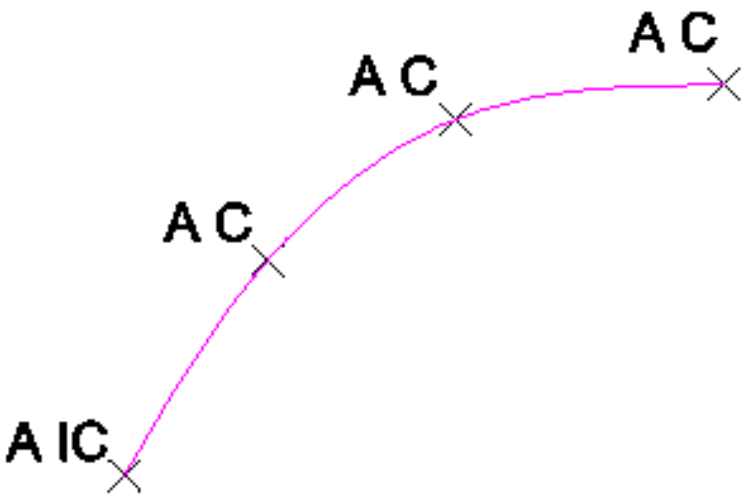

Figura 3.5: Representación de una curva Spline

- Se quiere dibujar un cuadrado, por ejemplo, una tapa de registro perteneciente a la red de saneamiento. El código de punto debe comenzar obligatoriamente por 'TC', seguido de la descripción que queramos asignarle, en este caso como es la red de saneamiento usaremos 'SAN'. Así finalmente, el código para ese punto queda 'TC SAN'. Como se pretende optimizar también el trabajo de campo, un cuadrado solo se definirá mediante 2 puntos y la aplicación al interpretar el código, deberá siempre dibujar un cuadrado a la derecha de esos dos puntos. La explicación es sencilla, si en un trabajo hay que medir 500 tapas de registro, no es lo mismo medir 1000 puntos que 2000. Con esta codificación nos hemos ahorrado la mitad del tiempo en el trabajo de campo.

- Lo mismo que en el caso anterior, para dibujar rectángulos. El código de punto debe comenzar obligatoriamente por 'TR', seguido de la descripción que queramos asignarle, en este caso como es la red de saneamiento usaremos 'SAN'. Así finalmente, el código para ese punto queda 'TR SAN'. Como se pretende optimizar también el trabajo de campo, un rectángulo solo se definirá mediante 3 puntos y la aplicación al interpretar el código, deberá siempre dibujar un rectángulo.

- Se quiere dibujar un círculo, por ejemplo, una tapa de registro perteneciente a la red eléctrica. El código de punto debe comenzar obligatoriamente por 'TX', seguido de la descripción que queramos asignarle, en este caso como es la red de eléctrica usaremos 'RE'. En este código necesitaremos también incluir en radio de la circunferencia, por ejemplo 0.5 metros, por lo que al final el código queda 'TX 0.5 RE'.

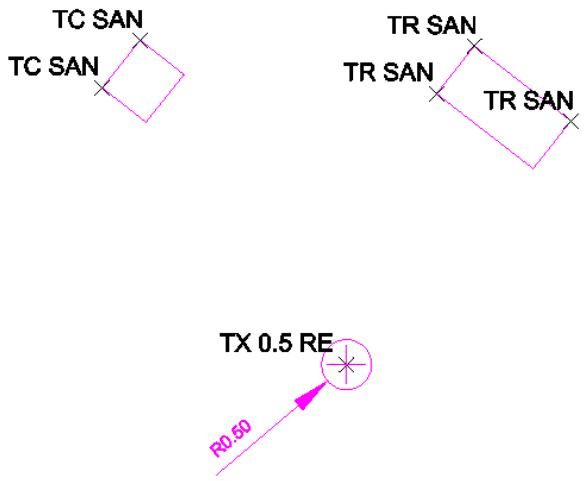

Figura 3.6: Representación de un rectángulo, un cuadrado y un círculo

- Por último, planteamos una opción más complicada y suele ser habitual en los trabajos de campo, la representación de elementos que realmente no se han medido en campo, porque pueden no ser accesibles. Queremos representar la línea de la fachada de un edificio, donde existen dos puntos inaccesibles, que no podemos medir topográficamente, pero conocemos su longitud (por ejemplo, hemos extraído estos datos de un plano catastral). Como es un edificio, podríamos representarlo con el texto 'E'. La siguiente secuencia de 4 puntos con su codificación, construirá la línea final formada por 6 puntos, ('E I','E 1.5 -2','E','E'). El código el punto 2, 'E 1.5 -2', indica que cuando la línea llegue a ese punto, debe continuar perpendicular a su derecha durante 1.5 metros, después, 2 metros, también en perpendicular, hacia su izquierda, y por último debe unirse con el siguiente punto que forma la línea.

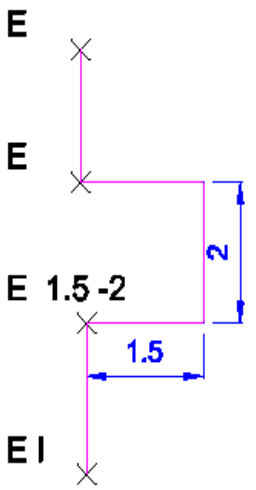

Figura 3.7: Representación de como se crearía una linea formada por puntos inaccesibles

Otra gran ventaja que aporta esta codificación es que los puntos en campo que forman un elemento no estás obligado a medirlos de forma consecutiva, lo que supone un importantísimo ahorro de tiempo. Vamos a explicar esto con un ejemplo.

Se pretende dibujar el plano de una carretera de 50 km, las líneas que nos interesan son; el eje de la carretera y las líneas de los arcenes. No podemos plantearnos el medir las líneas de forma consecutiva, por ejemplo, medir el arcén derecho, volver por el eje y, por último, el arcén izquierdo. Si hiciéramos esto, recorreríamos 150 km. Esta aplicación es capaz de crear distintas líneas, aunque no estén medidas en campo en un orden consecutivo. Una solución de codificación podría ser la siguiente secuencia de puntos: (‘AD I’,’E I’,’AI I’,’AI’,’E’,’AD’,’E’,’AD’,’AI’,. . .), donde ‘AD’ es el arcén derecho, ‘E’ el eje, ‘AI’ el arcén izquierdo. Así solo recorreríamos la carretera una sola vez.

Vemos que este tipo de codificación definida no solo aporta la ventaja de generar automáticamente el plano, si no que, el tiempo ahorrado en el trabajo de campo, es la mayor ventaja. Esta parte del trabajo es la que tiene mayores costes y, por lo tanto, más peso a la hora de decidirse por un proyecto.

Caracter	Significado
'I'	Punto inicio de línea
'IC'	Punto inicio de curva
'C'	Punto perteneciente a una curva
'TC'	Punto perteneciente a un cuadrado
'TR'	Punto perteneciente a un rectángulo
'TX'	Punto que define un círculo

Tabla 3.1: Caracteres obligatorios y significado

Técnicas y herramientas

4.1. Metodologías

Como metodología para el desarrollo del proyecto se ha utilizado **Scrum**. Scrum es un marco de trabajo, es el método ágil de desarrollo de Software más utilizado del mundo. Entre sus características principales están: Utilizar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto. La calidad del resultado obtenido depende más del conocimiento tácito de las personas que de la calidad de los procesos usados. Este método permite el solapamiento entre diferentes fases del proyecto. Algunos de sus componentes principales son: Sprint: Parte básica en este tipo de desarrollos, donde se desarrolla un incremento del producto que pueda ser utilizado. Pila del producto: donde están los requisitos de usuario, está información no es rígida, puede variar según va evolucionando el producto. Pila del sprint: lista con las tareas a realizar durante un sprint. Incremento: Parte del desarrollo obtenida al final de cada sprint.

Durante todo el proyecto se han ido realizando los diferentes sprints, en los cuales se han planificado las tareas siguientes y revisado, si se han ido cumpliendo los objetivos marcados en el sprint anterior.

4.2. Patrones de diseño

Flask utiliza como patrón de diseño, **MTV**. El patrón MTV es similar al conocido MVC, pero con algunas diferencias:

En el patrón MVC:

- **Modelo:** Es la parte que manipula la información.

- **Vista:** Decide como se mostrará la información.
- **Controlador:** Comunica el modelo con la vista.

Podemos ver a continuación sus equivalencias, para este caso, usando el framework Flask:

MVC vs MTV
Modelo=Modelo
Vista=Vista y Template
Controlador=Flask

Aquí, el framework Flask, es el que toma el papel del controlador. Resumiendo, en el patrón MTV, tenemos:

- **Modelo:** Es la parte que manipula la información, se encuentra en forma de clases de Python.
- **Vista:** Decide qué información se muestra y en que template.
- **Template:** Coge toda la información, la organiza y ve la manera en que esta se va a mostrar. Básicamente una página HTML con algunas etiquetas extras propias de Flask

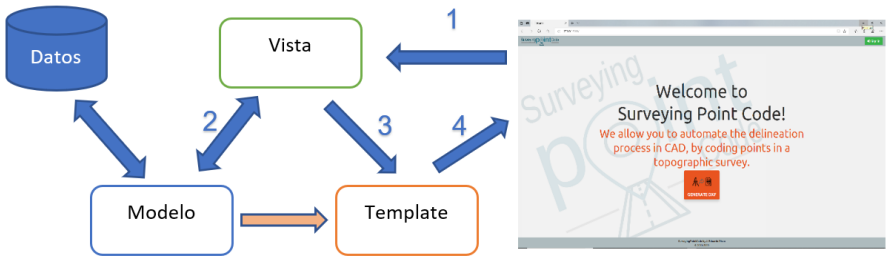

Figura 4.8: Esquema de un patrón MTV

En la imagen anterior podemos ver como son las relaciones en el patrón MTV:

1. El Navegador manda una solicitud
2. La vista interactúa con el modelo para obtener datos.

3. La vista llama a la plantilla.
4. La plantilla renderiza la respuesta a la solicitud del navegador

4.3. Control de Versiones

Existen varias herramientas en el mercado para el control de versiones:

- Git
- Subversion
- SourceSafe

Nos hemos decantado por **Git**, ya que es la que se ha venido usando durante la realización del grado y es una de las mas extendidas en el mercado.

Git es un sistema de control de versiones distribuido, libre y de código abierto. Git se distribuye bajo la licencia de software libre GNU LGPL v2.1.

4.4. Hosting del repositorio

Al igual que en el apartado anterior, aunque existen diversas posibilidades:

- GitLab
- GitHub
- Bitbucket

Como en el apartado anterior, por familiaridad con la plataforma durante el grado y también por ser una de las más usadas, de decidió usar **GitHub**.

GitHub permite tener cuentas gratuitas y de pago. Además, un tipo de cuenta para estudiantes, como alumno se puede solicitar acogerte al programa «GitHub education for students», que te permite tener repositorios privadas y da acceso a herramientas adicionales.

Además, GitHub ofrece otras funcionalidades y servicios como, por ejemplo; revisión de código, documentación, gestión de tareas, wikis e integraciones con otros servicios como por ejemplo CodeBeat.

4.5. Gestión del proyecto

Dentro de las muchas herramientas existentes como por ejemplo;

- ZenHub
- Trello
- VersionOne

Se ha optado por usar **ZenHub** ya que se encuentra integrado en el propio GitHub, funciona como una aplicación nativa en su interfaz y podemos controlar nuestros proyectos usando paneles de trabajo bastante intuitivos, así como conectar con varios repositorios en el panel de tareas y ver todos los temas abiertos que requieren de la atención.

Se trata de una plataforma de gestión de proyectos de software y entre las principales características de ZenHub están la de convertir los issues de GitHub en dinámicos tablones Kanban.

4.6. Entorno de desarrollo integrado (IDE)

Python

Como en los apartados anteriores se han tenido en cuenta diferentes herramientas para este trabajo:

- NotePad ++
- Sublime Text
- Microsoft Visual Studio Code
- PyCharm

Tanto Microsoft Visual Studio Code como PyCharm son perfectamente válidas y las más completas para realizar este tipo de proyectos, abarcan casi todos los lenguajes de programación, contienen bastantes plugins y facilitan el trabajo a la hora de realizar los test.

Al final nos hemos decantado por **PyCharm**. Al haberse realizado el proyecto con Python, este entorno ha sido diseñado para este lenguaje, por lo que ofrece algunas ventajas más, respecto a Microsoft Visual Studio

Code. PyCharm es soportado por Windows, Mac OS y Linux, y su versión gratuita es bastante completa. Existen también licencias individuales para estudiantes.

LaTeX

La documentación del proyecto ha desarrollado en LaTeX. Para este propósito se han tenido en cuenta las siguientes herramientas:

- MikTex, para SO Windows
- texlive, para SO Linux
- MacTex, para SO Mac
- Compatible con las 3 plataformas, Texmaker
- Editor en línea, overleaf

Se ha optado por **Texmaker**, como recomendación del tutor del proyecto. Integra la mayoría de herramientas necesarias para la escritura de documentos en LaTeX, es gratuito y tiene licencia GNU GPL v2.

4.7. Servicios de integración continua

Calidad del código

Las diferentes herramientas consideradas para este propósito han sido:

- SonarQube
- CodeBeat

Se ha optado por **CodeBeat**, ya que se integra perfectamente con GitHub. Es una herramienta gratuita, que soporta gran cantidad de lenguajes y permite personalizar totalmente, las métricas que usa para el análisis de la calidad del código.

4.8. Herramienta de análisis sintáctico

Como herramienta de análisis sintáctico, para poder dar validez a los archivos de entrada, se pensó en estas 3 posibilidades:

- Ply
- Antlr
- Flex Bison

La opción elegida ha sido **Ply**, ya que está implementada completamente en Python y encaja perfectamente con la filosofía de realizar la mayor parte del proyecto con este lenguaje.

Aunque Ply también es una librería y debería estar en el apartado de librerías, le hemos querido dar un apartado especial como herramienta de análisis sintáctico.

Ply es una biblioteca de Python. Es una implementación de lex y yacc de herramientas de análisis para Python, que proporciona la mayor parte de las características estándar de lex / yacc, incluyendo; el apoyo a las producciones vacías, las reglas de prioridad, la recuperación de errores y soporte para gramáticas ambiguas. Es muy sencillo de usar y también muy efectivo a la hora de realizar la comprobación de los errores. Ply utiliza el análisis LR, el cual puede incorporar grandes gramáticas fácilmente.

Ply, básicamente está compuesto de 2 módulos:

- ply.lex – Donde se trata la parte del análisis léxico
- ply.yacc - Este módulo es para crear un parser.

4.9. Librerías

A continuación, vemos el resto de las librerías necesarias en el desarrollo de este proyecto:

ezdxf

ezdxf es la librería más importante del proyecto. En un principio, se tuvieron en cuenta otras opciones como:

- dxfgrabber
- SDXF
- dxfwrite

resultando la opción más interesante ezdxf, ya que era la única que cubría todas las necesidades que aquí se planteaban.

ezdxf es una interfaz de Python para el formato DXF, desarrollado por Autodesk, y que permite a los desarrolladores leer y modificar dibujos DXF existentes o crear nuevos dibujos DXF.

Con sus métodos y atributos nos permite crear, modificar, eliminar o consultar las propiedades de todos los elementos que puede contener un archivo de CAD. ezdxf es independiente del sistema operativo y se ejecuta en todas las plataformas que proporcionan un intérprete de Python adecuado (>= 3.6). Esta bajo la licencia MIT-License.

Una parte a destacar, es que nos permite generar el DXF en distintas versiones de un archivo CAD, lo cual es muy interesante.

Version	AutoCAD Release
AC1009	AutoCAD R12
AC1012	AutoCAD R13 -> R2000
AC1014	AutoCAD R14 -> R2000
AC1015	AutoCAD R2000
AC1018	AutoCAD R2004
AC1021	AutoCAD R2007
AC1024	AutoCAD R2010
AC1027	AutoCAD R2013
AC1032	AutoCAD R2018

Figura 4.9: Versiones de CAD

SQLAlchemy

Se ha elegido la librería de Python **SQLAlchemy** por que es el ORM que mejor se adapta para trabajar con Flask y PostgreSQL

Flask-Login

Se ha elegido la librería de Python **Flask-Login** para poder realizar la gestión de sesiones de usuarios en la aplicación.

Bootstrap

Bootstrap es un framework para desarrollo front-end. Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es una herramienta Open Source

Colorpicker Bootstrap

Colorpicker Bootstrap, es un plugin de selector de color modular para Bootstrap 4. Esta bajo la licencia MIT-License.

TinyColor

Es un micro framework que permite la manipulación y conversión de color en JavaScript. Permite muchas formas de entrada, mientras que proporciona conversiones de color y otras funciones de utilidad de color. No tiene dependencias. Esta bajo la licencia MIT-License

bs-custom-file-input

bs-custom-file-input ,es un plugin para Bootstrap 4. Ayuda a crear una entrada de selección de archivos personalizada con el botón del navegador para cargar archivos. También es compatible con múltiples selecciones de archivos y arrastrar y soltar. No tiene dependencias. Esta bajo la licencia MIT-License

4.10. Desarrollo Web

Para el desarrollo web se ha optado por utilizar **Flask**, siguiendo, como se ha comentado anteriormente con la idea de englobar lo máximo posible

en Python. Flask es un "micro" Framework escrito en Python y concebido para facilitar el desarrollo de aplicaciones Web bajo el patrón MTV.

Entre algunas de sus ventajas, están:

- Desarrollo de aplicaciones web de una forma ágil y rápida, tiene una buena curva de aprendizaje
- No se necesita una infraestructura con un servidor web.
- Compatible con wsgi.
- Soporta de manera nativa el uso de cookies seguras.
- Se pueden usar sesiones.
- Flask es Open Source y tiene una licencia BSD.
- Extensa documentación.

4.11. Base de Datos. PostGIS

Como sistema de base de datos se ha decidido utilizar **PostGIS**. PostGIS convierte al sistema de administración de bases de datos **PostgreSQL** en una base de datos espacial mediante la adición de tres características: tipos de datos espaciales, índices espaciales y funciones que operan sobre ellos.

Puede parecer en un principio, que el uso de esta base de datos no está justificada en este proyecto, ya que de momento solo sirve para gestionar a los usuarios de la aplicación. Habitualmente trabajo con bases de datos espaciales y PostGIS es la más usada en en el campo de los sistemas de información geográfica. A parte de poder usarla en desarrollos particulares, como este, programas de GIS como QGis, gvSIG o ArcGis, que son aplicaciones de escritorio, permiten una conexión directa con PostGIS. La idea es que en un futuro ese contenedor pueda almacenar bases de datos espaciales, de otras aplicaciones de futuros desarrollos o de las que uso habitualmente. Ya que la vamos a tener en un contenedor Docker con una base de datos alojada en el, darle el máximo uso en el futuro.

En España casi todas las instituciones que trabajan con datos geoespaciales , trabajan con PostGIS, es una buena oportunidad para familiarizarse con ella.

La aplicación **SurveyingPointCode**, en un futuro, aumentará sus funcionalidades y para ello será necesario trabajar con PostGIS.

4.12. Despliegue de la aplicación

Para el despliegue de la aplicación se ha optado por utilizar **Docker**. Principalmente, porque es una herramienta de mucha actualidad y con un gran potencial, tanto en el desarrollo, como en el despliegue de aplicaciones.

La idea básica de su funcionamiento es la creación de una serie de contenedores muy ligeros y portables, así las aplicaciones, podrán ejecutarse en cualquier equipo, solamente con tener Docker instalado, con independencia del sistema operativo.

Entre algunas de sus ventajas, están:

- Es muy sencillo de usar.
- Ahorra tiempo, no necesitamos instalar diferentes softwares.
- Son muy ligeros, a diferencia de las máquinas virtuales ya que no necesitan un sistema operativo completo, cogen lo que necesitan de cada máquina anfitriona.
- Portabilidad.
- Los repositorios Docker, nos permite acceder a gran cantidad de imágenes y así poder de ejecutar prácticamente todas las aplicaciones.
- Es un entorno seguro y no ofrece variaciones, da igual cual sea el equipo ni el ambiente. Facilita las pruebas, el desarrollo y las visualizaciones por parte del cliente.
- Es Open Source.
- Extensa documentación.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más importantes del desarrollo del proyecto. Se tratará de exponer el camino que se ha tomado con sus correspondientes implicaciones, así como describir los diferentes problemas a los que ha habido que enfrentarse y las soluciones con las que se ha tratado de resolverlos.

5.1. Inicio del proyecto

Me considero una persona con muchas inquietudes y siempre pensando en mejorar tareas que suelo realizar con frecuencia. He desarrollado mi carrera profesional en el campo de la Topografía y la Geodesia, y actualmente está mas enfocado a temas relacionados con la Cartografía.

Trabajando en varios proyectos, he podido observar la cantidad de tiempo y recursos que se empleaban, en realizar levantamientos topográficos y posteriormente gestionar todos esos datos para producir un plano. De ahí surge la idea de como mejorar el trabajo de campo mediante un tipo de codificación, y que esta a su vez sirviera para automatizar al máximo el proceso de delineación, a la hora de crear el plano.

El profundizar en los conocimientos adquiridos cursando el Grado en Ingeniería Informática y a la vez poder facilitar el trabajo en este tipo de proyectos, me ha animado ha hacer viable este proyecto. A la hora de decidirme, también me ha parecido importante, la idea de que la aplicación desarrollada es de gran utilidad, y seguro que va a ser bien recibida por la comunidad de topógrafos.

La idea fue aceptada por los tutores, y comenzamos con el proyecto.

Figura 5.10: Logo de SurveyingPointCode

5.2. Gestión del proyecto

En la primera reunión con los tutores, se estableció cual iba a ser la metodología a seguir en la realización de este proyecto. Se iba a emplear la metodología ágil Scrum.

Se realizarían una serie de sprints llevados a cabo con una periodicidad media semanal, donde se entregaría una parte del producto operativo, el incremento. Al finalizar cada sprint se realizarían reuniones para planificar las tareas a realizar en el el sprint siguiente, en forma de pila del sprint, y revisar si se habían alcanzado los objetivos marcados en el sprint anterior.

En GitHub, con la herramienta ZenHub, se visualizarían el estado y prioridad de las tareas del sprint. Los avances y cambios en el desarrollo del proyecto, se almacenaría en GitHub, que nos permitiría seguir al detalle todo el histórico del proyecto.

Las reuniones de los sprints resultaron muy interesantes. En ellas hubo modificaciones sobre la idea inicial de la que se partía, en algunos casos

desechado alguna funcionalidad, pero en la mayoría de los casos, aportando nuevas funcionalidades que hacían crecer el valor de la aplicación, y sobre todo me aportaron una visión más realista a la hora de abordar unos determinados objetivos o ideas, preguntándome, si el tiempo o recursos invertidos merecían la pena o aportaban algún valor a la aplicación.

5.3. Formación

La realización del proyecto requería una serie de conocimientos técnicos, en general, en el desarrollo de aplicaciones web,y en particular en tecnologías como; Flask, HTML5, CSS3, JavasScript, Docker y librerías como Ply, ezdxf, Boostrap, TinyColor,...

Los mayores esfuerzos se pusieron en comprender el funcionamiento de Flask, Ply, ezdxf y Docker, ya que el óptimo funcionamiento de la aplicación en estos puntos, era fundamental para logar conseguir los objetivos finales.

Como la aplicación se ha desarrollado en Python, era importante conocer las guías de estilo y la convenciones de este lenguaje, **PEP 8** y **PEP 257** entre otras. Para ello se consultaron las siguientes fuentes:

- PEP 8 – Style Guide for Python Code [**?**]
- PEP 257 – Docstring Conventions [**?**]

Para la formación en **Flask** se consultaron las siguientes fuentes:

- Building Web Applications with Flask [**?**]
- Instant Flask Web Development [**?**]

Para la formación en **Ply** se consultaron las siguientes fuentes:

- Documentation for PLY [**?**]
- Prototyping Interpreters using Python Lex-Yacc [**?**]
- Parses chemical equations using the PLY parser generator [**?**]

Para la formación en **ezdxf** y las otras posibles alternativas,se consultaron las siguientes fuentes:

- Documentation for ezdxf [**?**]
- Documentation for dxfgrabber [**?**]
- Documentation for dxfwrite [**?**]
- Documentation for SDXF [**?**]

Para la formación en **SQLAlquemy** y **FLask-Login**, se consultaron las siguientes fuentes:

- Documentation for SQLAlquemy [**?**]
- Documentation for Flask-Login [**?**]

Para la formación en **HTML5** se consultaron las siguientes fuentes:

- HTML5 Tutorial [**?**]

Para la formación en **CSS3** se consultaron las siguientes fuentes:

- CSS Tutorial [**?**]
- Lenguaje CSS [**?**]

Para la formación en **JavaScript** se consultaron las siguientes fuentes:

- Eloquent JavaScript: A Modern Introduction to Programming [**?**]
- Javascript a fondo [**?**]

Para la formación en **Bootstrap** y librerías relacionadas, se consultaron las siguientes fuentes:

- Bootstrap [**?**]
- TinyColor JavaScript color tooling [**?**]
- Bootstrap Colorpicker [**?**]

Para la formación en **Docker** se consultaron las siguientes fuentes:

- Curso Docker [**?**]
- Docker for beginners [**?**]
- Introduction to Docker [**?**]
- Docker-postgis [**?**]
- Overview of Docker Compose [**?**]

5.4. Desarrollo de la aplicación

Los primeros pasos en el desarrollo de la aplicación consistieron en desarrollar un pequeño prototipo funcional, que fuera incrementando su funcionalidad progresivamente.

El primer prototipo, tenía que conseguir leer un archivo de campo y darlo por válido o como erróneo. Una vez que se tuvo clara como iba a ser la codificación de cada punto, (según hemos visto en el apartado 3.3 Codificación de los puntos), el siguiente paso era definir una gramática para ese tipo de archivo, para poder validarla. Entre los formatos de archivo más comunes que podemos obtener de los equipos topográficos, se decidió que los archivos deberían ser de tipo "csv" o "txt", separando sus campos por comas. Los campos son; número de punto, coordenada x, coordenada y, coordenada z y código del punto.

A continuación, vemos un ejemplo, con todas las posibilidades de codificación que aceptará la aplicación:

```
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
```

Se estudió detenidamente esta sintaxis y la formalización de la gramática quedo de la siguiente forma:

```
entrada: líneas

líneas: línea | líneas '\n' línea

línea: núm_punto coordenadas código

núm_punto: INT

coordenadas: FLOAT, FLOAT, FLOAT

INT: [0-9]+

FLOAT: -?( [0-9]*.[0-9]+)

ID: [a-zA-Z] +

código: código_capa CÓDIGO_GEOMÉTRICO
| CÓDIGO_ELEMENTO_SINGULAR código_valor_texto
| código_capa código_no_accesible
| código_elemento_singular
| código_capa

código_capa: ID

CÓDIGO_GEOMÉTRICO: "I" | "IC" | "C"

CÓDIGO_ELEMENTO_SINGULAR: "TC" | "TR" | "TX"

código_valor_texto: FLOAT ID
    | INT ID
    | FLOAT
    | INT
    | ID

código_no_accesible: (FLOAT | INT)+
```

El prototipo utilizando la librería **Ply** y con esta gramática, consiguió validar un archivo correcto.

Otro punto clave en el desarrollo del algoritmo fue descifrar los códigos que contenía el archivo de entrada. Como se ha comentado anteriormente,

este código puede tener distintos significados dependiendo de su estructura, por ejemplo:

- AR, puede ser un punto que se guarde en una futura capa llamada Árboles.
- TX 2 SAN, puede ser un punto que es el centro de un circulo de radio 2 y que se guarde en una futura capa llamada Saneamiento.
- M I, puede ser un punto donde comienza una línea que define un muro y que se guarde en una futura capa llamada Muros.
- ...

El algoritmo debía extraer del archivo todos los elementos, identificarlos y guardarlos en alguna estructura de datos, clasificados por:

- Capas
- Puntos
- Líneas
- Curvas
- Cuadrados
- Rectángulos
- Círculos

Añadiendo complejidad al archivo de entrada, se comprobó que el algoritmo resolvía de forma satisfactoria este paso.

Por último, para poder tener un prototipo básico que cumpliera los objetivos, debíamos comprobar que el prototipo era capaz de crear un archivo **DXF**, con los elementos del archivo de entrada. Incorporando la librería **ezdxf**, y haciendo uso de sus métodos y atributos, se testeó que dibujara en un primer momento solo líneas y curvas, siendo capaz de generarlas en el orden correcto.

Con estos datos de entrada:

```
1,355776.180,4611015.011,691.055,E I
2,355773.203,4611028.546,691.055,E
3,355761.305,4611083.365,691.055,E
4,355767.050,4611086.462,691.055,A I
5,355774.447,4611083.001,691.055,M I
6,355769.692,4611104.977,691.055,M
7,355763.294,4611103.572,691.055,A
8,355756.420,4611105.247,691.055,E
9,355781.359,4611129.076,691.055,TR RE
10,355782.355,4611131.314,691.055,TR RE
11,355788.919,4611128.392,691.055,TR RE
12,355766.146,4611121.294,691.055,M
13,355759.713,4611119.882,691.055,A
14,355750.443,4611133.259,691.055,E
15,355786.052,4611135.542,691.055,TC TEL
16,355786.849,4611136.145,691.055,TC TEL
17,355764.814,4611134.156,691.055,ARB
18,355783.215,4611143.179,691.055,TX 0.5 SAN
19,355761.943,4611140.629,691.055,M
20,355755.486,4611139.240,691.055,A
21,355754.037,4611145.893,691.055,A IC
22,355745.977,4611153.511,691.055,E
23,355755.345,4611150.953,691.055,A C
24,355758.897,4611153.511,691.055,A C
```

El resultado fue:

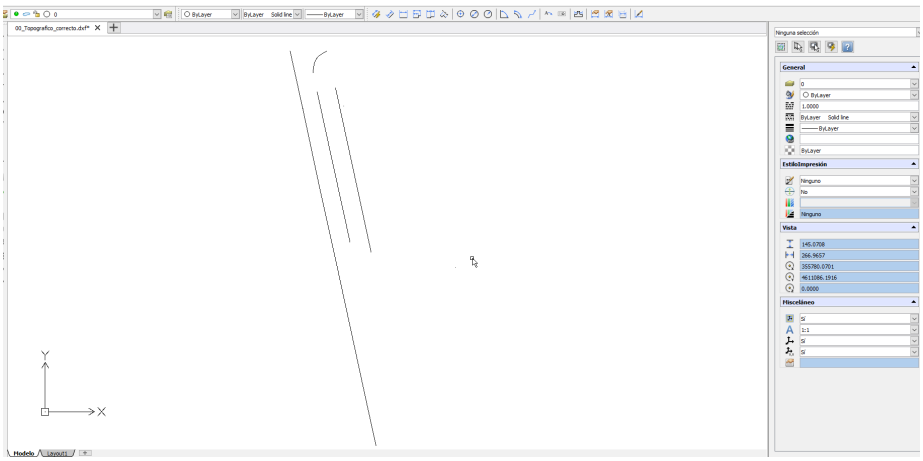

Figura 5.11: Dibujo obtenido con el prototipo inicial

Es resultado fue el esperado, por lo que se concluyó está primera etapa del desarrollo con un pequeño prototipo que cumplía las funcionalidades previstas inicialmente:

- Leer y validar un archivo con una gramática determinada.
- Interpretar y organizar todos los elementos del archivo, a partir de su codificación.
- Crear un archivo DXF con estos elementos.

A partir de aquí, como ya se había decidido que iba a ser una aplicación Web, el próximo objetivo era desarrollar la aplicación Web con **Flask**. El proceso iba a ser el mismo que con el prototipo anterior, desarrollar un prototipo de aplicación Web que fuera incrementando sus funcionalidades.

A continuación, se enumeran brevemente y ordenadas temporalmente, las funcionalidades más importantes que se han ido añadiendo al prototipo Web:

- Subir un archivo al servidor.
- Procesar y descargar un archivo al equipo del usuario.
- Conexión con una BBDD. (En este punto se comenzó a trabajar con Docker, la BBDD iba dentro de un contenedor).

Código de punto	Capa de CAD	Color	Símbolo
E	Edificación	Rojo	
AR	Vegetación	Verde	Árbol
SAN	Saneamiento	Amarillo	
P	Saneamiento	Amarillo	Pozo

Tabla 5.2: Esquema archivo de configuración

- Login, Logout y Registro. Para implementar esta parte se trabajó con la biblioteca Flask-Login.
- Sesiones de usuarios.

Estas funcionalidades cubrían las expectativas de que la aplicación fuera una aplicación Web, el prototipo Web ya se había conseguido.

A partir de aquí, se han ido desarrollando todas las funcionalidades de la aplicación tanto en Frontend como en Backend.

5.5. Problemas y curiosidades surgidos en el desarrollo del proyecto

A continuación, se exponen algunos puntos críticos, problemas surgidos, curiosidades, y como se abordaron.

Conflicto entre parsers

Más adelante surgió la idea que el usuario pudiera subir un archivo con una configuración personalizada para hacer la conversión de archivo de campo a archivo DXF.

Como vemos en la Tabla 5.2, relacionamos códigos de punto, con capas de CAD, colores y símbolos. Cuando exista un trabajo con múltiples códigos, capas y colores, se quiere dar al usuario la posibilidad de no tener que configurar cada vez esta conversión en la interfaz, sino que a través de un archivo personalizado, con sus códigos, capas, colores y símbolos esto sea también automático.

Para ello se formalizó otra gramática para validar este tipo de archivos y detectar sus errores. La gramática elegida fue la siguiente:

```
entrada: líneas

líneas: línea | líneas '\n' línea

línea: TEXT COMA TEXT COMA color COMA TEXT
| TEXT COMA TEXT COMA color

color: LPAREN INT COMA INT COMA INT RPAREN

TEXT: [a-zA-Z0-9_] +

INT: [0-9] +

LPAREN = '\('

RPAREN = '\)'
```

Los parsers estaban definidos en diferentes módulos, el parser para el archivo del campo estaba en módulo *conversor.py* y el del archivo de configuración en el módulo *uploadoptionalfiles.py*. Cuando se ejecutan, se crean los archivos *parser.out* y *parsetab.py*. Probando a cargar los dos archivos a la vez, se pudo comprobar que se parseaba un archivo con el parser que no le correspondía, esto no sucedía siempre, si no que pasaba de una forma aleatoria, como si en algún momento se cruzara la información entre parsers.

Esto se solucionó añadiendo un parámetro personalizado para cada archivo, al ahora de ejecutar el parser. En un principio estaba definido de esta forma para ambos casos:

```
lex.lex() # lexer part
...
punto = parser.parse(line) # parser part
```

Definiendo parámetros independientes en cada caso, quedó de la siguiente forma.

Para el archivo de campo:

```
lexer_topographycal=lex.lex() # lexer part
```

```
...
# añadiendo el nuevo parámetro
punto = parser.parse(line,lexer=lexer_topographycal)
```

Para el archivo de configuración:

```
lexer_config = lex.lex() # lexer part
...
# añadiendo el nuevo parámetro
c_line = parser.parse(line, lexer=lexer_config)
```

Para dar con esta solución, se consultó la documentación de PLY, concretamente el apartado **Multiple Parsers and Lexers**[**?**].

Codificación UTF-8

Hasta ahora la aplicación funcionaba correctamente a la hora de subir los archivos y validarlos. Se decidió modificar la gramática del archivo de configuración, para que aceptará la letra 'ñ' y todos los tipos de tildes, ya que se suelen utilizar al poner los nombres de las capas de CAD.

Simplemente se sustituyó:

```
TEXT: [a-zA-Z0-9_] +
```

por;

```
TEXT: [a-zA-ZÀ-ÿ0-9ñÑ_] +
```

Con esto debería funcionar perfectamente, pero no fue así, el archivo que contenía el carácter 'ñ', daba errores. Se estaba probando la aplicación simultáneamente en 2 sistemas operativos Windows y Linux, y para mayor sorpresa en Linux funcionaba y en Windows no.

Al final, dedujimos que se podía tratar de un problema de codificación del archivo, e investigando en esa linea decidimos probar dos configuraciones, UTF-8 y Latin-1, en los 2 sistemas , y en cada uno sucedía lo contrario, como se puede ver en la Tabla 5.3.

Sistema Operativo	UTF-8	Latin-1
Windows	Fallo	Correcto
Linux	Correcto	Fallo

Tabla 5.3: Test Windows vs Linux para UTF-8 y Latin-1

La solución encontrada fue añadir en la función que abre el archivo, el parámetro *encoding* indicando el tipo de codificación específicamente , en este caso UTF-8.

En el código se sustituyó:

```
with open(input_file) as f:
```

por;

```
with open(input_file, encoding='utf-8') as f:
```

Se comprobó en ambos sistemas operativos, y el resultado fue correcto, por que que se dio esta solución como válida.

Paleta de colores en CAD

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.