UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Generador automático de Metrominuto Documentación Técnica

Presentado por Guillermo Paredes Muga
en Universidad de Burgos — 20 de marzo de 2020
Tutor: Dr. Álvar Arnaiz González y Dr. César Ignacio García Osorio

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apéndice se va a analizar todo aquello necesario para que un proyecto se desarrolle con el menor número de imprevistos posible. Para conseguir esto, es necesaria una fase de planificación donde se estimen los tiempos, la cantidad de trabajo y el dinero que es necesario invertir para sacar adelante el proyecto. Dicha planificación se divide en dos partes:

- **Planificación temporal**: fase en la que se analiza y planifica el trabajo necesario para desarrollar cada parte del proyecto, marcando fechas de inicio y final para cada una de ellas.
- **Estudio de viabilidad**: fase en la que se analizan las repercusiones legales y económicas del proyecto:
  - Económica: Análisis de los posibles costes y beneficios del proyecto.
  - Legal: Análisis de las repercusiones a efectos legales como la *Ley de Protección de Datos* o las licencias del proyecto.

A.2. Planificación temporal

Esta parte del proyecto es aquella en la que se planifica cómo va a ir avanzando el proyecto en función del trabajo requerido para cada una de las tareas de las que consta el mismo. Concretamente se estima el tiempo, es decir, cuáles van a ser los plazos para desarrollar determinadas tareas. Para

esta planificación o estimación se han empleado los conceptos generales de la metodología ágil Scrum, ya que en este caso en el proyecto sólo hay un único desarrollador aparte de los tutores. Las líneas generales que se han aplicado de esta metodología de gestión han sido:

- Desarrollo marcado por sucesivos *sprints* delimitados por dos reuniones: una al principio de cada uno y otra al final. Normalmente en la reunión de finalización de un *sprint* se marcaban los objetivos y la planificación del siguiente (siendo la primera reunión).
- La duración de los *sprints* fue de dos semanas al inicio y de una semana en la segunda mitad del mismo.
- Cada *sprint* produce un resultado o incremento del proyecto final.
- En cada *sprint* se divide el objetivo final en distintas tareas mas pequeñas.
- Las tareas se planifican y estiman en un tablero.

Sprint 0. (22/10/2019 – 28/10/2019)

En esta reunión el objetivo fundamental fue la presentación, a grandes rasgos, de en qué iba a consistir el proyecto por parte de los dos tutores: Álvar Arnaiz González y César Ignacio García Osorio. No se definió ninguna tarea, ya que simplemente se trataba de acordar si se había entendido bien el objetivo del proyecto.

Sprint 1. (29/10/2019 – 12/11/2019)

Esta reunión fue la primera en la que se comenzó a hablar de los requisitos del proyecto y de sus detalles para la planificación. Los objetivos de este sprint fueron:

- Crear correctamente el repositorio en GitHub.
- Elegir el entorno de desarrollo que se iba a utilizar y su posterior configuración para ejecutar una aplicación web con Flask.
- Hacer un primer proyecto «Hola Mundo» como primera toma de contacto con este framework.
- Incorporar en el proyecto un mapa proporcionado por el API de Google en el que fuésemos capaces de seleccionar diferentes puntos (marcadores).

Sprint 2. (13/11/2019 – 26/11/2019)

Los objetivos principales de este Sprint fueron:

- Guardar e imprimir (tanto en el cliente como en el servidor) los distintos marcadores seleccionados en el mapa.
- Dibujar la ruta entre los distintos puntos seleccionados.
- Seguir los estándares de programación
- Completar la documentación del proyecto.
- Cambiar de Visual Studio Code a Pycharm (licencia profesional).

Además de estos objetivos, debido a la posibilidad de añadir al proyecto nuevas funcionalidades y metodologías de Docker, se decidió cambiar de Sistema Operativo a Linux. Durante este sprint se siguió un curso «tutorial sobre Flask de Miguel Grinberg» //mejor cita?, en el cual, a medida que iba avanzando encontré varias mejoras para aplicar a mi proyecto y que fui incorporando. Al final de este sprint, dos de los objetivos no se consiguieron por completo, ya que daban algunos errores y se optó por una funcionalidad menor: en vez de dibujar la ruta entre todos los puntos seleccionados, sólo se dibujaba entre el primero y el ultimo; y al pasar los distintos marcadores, mediante un POST al servidor no podía pasar un objeto. Estas dos funcionalidades quedaron pendientes para el siguiente Sprint.

Sprint 3. (27/11/2019 – 10/12/2019)

El principal objetivo de este sprint fue poner al día la documentación al mismo tiempo que se seguía el tutorial de Flask mencionado en el A.2. Se encontraron distintas mejoras para realizar, así como la posibilidad de añadir el fichero `requirements.txt`, que después serviría para instalar las diferentes librerías utilizadas en el proyecto. También se acabaron las tareas que quedaron pendientes el sprint anterior.

Sprint 4. (10/12/2019 – 18/12/2019)

Los objetivos de este sprint fueron: mejorar el envío de los datos al servidor, buscar documentación y evaluar los resultados de las distintas funciones que proporciona el API de Google para Python, y ser capaces de diferenciar los datos «útiles» de dichas funciones.

Sprint 5. (19/12/2019 – 09/01/2020)

En este Sprint, los objetivos fueron:

- Mostrar la información de los diferentes puntos seleccionados.
- Obtener la matriz de distancias de todos con todos. Se obtiene como resultado una matriz en la que tenemos las distancias de todos los nodos entre sí.
- Dibujar el grafo en la parte del servidor, es decir, en Python.

Tras concluir el sprint, se alcanzaron todos los objetivos, no quedando nada pendiente para el siguiente.

Sprint 6. (10/01/2020 – 23/01/2020)

Los objetivos de sprint fueron:

- Actualizar la documentación.
- Evaluar el algoritmo *minimum spanning tree*.
- Valorar diferentes bibliotecas para dibujar el grafo en la web.

Tras concluir el sprint, quedó pendiente algunas preguntas sobre la parte de documentación, además de la implementación del grafo en el cliente.

Sprint 7. (24/01/2020 – 10/02/2020)

Durante este Sprint, los objetivos que se marcaron fueron los siguientes:

- La correcta implementación de la librería encontrada en el Sprint anterior.
- Creación de una estructura en la que a partir del árbol generado inicialmente y con la ayuda de la función minimum spanning tree de networkx, evaluar este camino mínimo en un conjunto aleatorio de arcos del árbol inicial.
- Crear el archivo JSON necesario para que la librería implementada anteriormente dibuje el grafo que necesitamos.

- Actualizar y corregir la documentación.

En este Sprint no se logró alcanzar por completo los objetivos marcados, quedando pendiente para el siguiente Sprint terminar la generación de la estructura del archivo JSON para dibujar el grafo.

Sprint 8. (10/02/2020 – 19/02/2020)

Los objetivos de este Sprint, que fue el primero en el que se pasó de realizar reuniones cada dos semanas a realizarlas cada semana, fueron:

- Terminar las tareas del sprint anterior.
- Actualizar la documentación del proyecto.
- Mejorar el sistema de generar un grafo por votos para que todos sus nodos estén siempre conectados.

Sprint 9. (20/02/2020 – 26/02/2020)

En la reunión de planificación de este sprint, que corresponde también con la de revisión del Sprint 8, se planteó la dificultad que presentaba emplear la biblioteca encontrada en los sprints anteriores para dibujar el mapa sinóptico. Esto se debe fundamentalmente a que la estructura de datos necesaria para ello era prácticamente imposible de generar de forma automática. Por esta razón, se decidió emplear otra forma para su dibujado, lo que lleva a los objetivos de este sprint:

- Eliminar referencias y usos de la biblioteca Tube Map.
- Generar el grafo en el servidor y exportarlo como svg para poder tratarlo a nuestro antojo en el cliente.
- Actualizar la documentación, junto con las correcciones del sprint anterior.

Sprint 10. (27/02/2020 – 04/03/2020)

Los objetivos de este sprint fueron:

- Generar el archivo SVG con el grafo generado. Se encontraron varias dificultades a la hora de escalar los puntos para dibujarlos en el SVG. Por ello, esta tarea quedaría pendiente para el siguiente Sprint.

- Generar la documentación.
- Investigar sobre los *layouts* para grafos.
- Dar al usuario la posibilidad de elegir el número de arcos que aparecen en el grafo.

Sprint 11. (05/03/2020 – 12/03/2020)

En este Sprint, los objetivos fueron:

- Generar toda la documentación pendiente del Sprint anterior junto con la nueva generada en este.
- Resolver el escalado del grafo en SVG.
- Añadir estilos al grafo en el cliente. Para ello fijarse en el Java script de la librería utilizada en Sprints anteriores.
- Añadir texto al grafo SVG.
- Dar al usuario la posibilidad de elegir el número de veces que se repite el bucle que genera el grafo de votos.

Durante este sprint, al igual que en el anterior, se encontró una gran dificultad en la transformación de las coordenadas geográficas de los marcadores a pixeles para su dibujado en SVG. En este Sprint se alcanzaron todos los objetivos.

Sprint 12. (13/03/2020 – xx/03/2020)

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apéndice se explican y especifican tanto los requisitos funcionales como los no funcionales del proyecto, así como los objetivos del mismo.

B.2. Objetivos generales

- Crear una aplicación cliente - servidor que permita la creación automática de metrominutos.
- Ofrecer control de usuarios (posible idea futura junto con lo comentado en la reunión de las API KEYs?)
- Permitir a los usuarios control sobre el mapa, de manera que puedan mover o eliminar los puntos seleccionados.
- Ofrecer al usuario un mapa final claro y sencillo.
- Que la aplicación final sea útil para el fomento de esta actividad.

B.3. Catálogo de requisitos

Requisitos funcionales

- **RF-1 Control de usuarios**

- **RF-2 Control sobre el mapa:** La aplicación debe poder ofrecer la selección de distintos puntos sobre el mapa.
  - **RF-2.1 Creación:** Debe poder añadir tantos puntos como desee.
  - **RF-2.2 Modificación:** Una vez creado un marcador, futuro nodo del grafo, el usuario debe poder moverlo.
  - **RF-2.3 Eliminación:** El usuario debe poder eliminar los puntos.
  - **RF-3 Generación de mapas:** Generación de un mapa a través de los puntos seleccionados en el *Requisito Funcional 2*.

Requisitos no funcionales

- **RNF-1 Usabilidad:** La aplicación tiene que poder usarse de forma sencilla y debe ser intuitiva.
- **RNF-2 Compatibilidad:** La aplicación tiene que poder ser compatible con los diferentes navegadores.

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado de la documentación se expone el diseño que ha dado lugar a la aplicación, el cual incluye el diseño de las distintas estructuras de datos, el diseño procedimental y diseño arquitectónico.

C.2. Diseño de datos

La estructura del proyecto podemos dividirla en varias partes ya que, debido al uso de bibliotecas como la de Google Maps o de la biblioteca de Networkx para la generación de grafos, se han usado todos o parte de los datos que nos devuelven como resultado de las consultas.

NetworkX

En este proyecto se usa networkX para generar, modificar y visualizar grafos, en los cuales el elemento «*nodes*» representa los diferentes puntos seleccionados por el usuario, y el elemento «*edges*» representa las conexiones entre ellos.

Google API

Google se ha usado para la obtención de todos los datos necesarios para el cálculo de distancias y tiempos, así como para la selección de los diferentes puntos en el mapa. Para ello, Google proporciona un API para *Python* y otro para *JavaScript*. Las funciones que se han usado han sido:

- distance_matrix(orígenes, destinos): devuelve las distancias de cada origen con cada destino.
- directions(): devuelve las direcciones que hay que seguir para llegar de un punto a otro.

C.3. Diseño procedimental

C.4. Diseño arquitectónico

La estructura del proyecto esta condicionada por el tipo de proyecto que es. Se trata de una aplicación web y por ello se ha seguido el patrón MVC (Modelo - Vista - Controlador), el cual permite separar en 3 componentes diferentes los datos, el interfaz y la lógica de la aplicación.

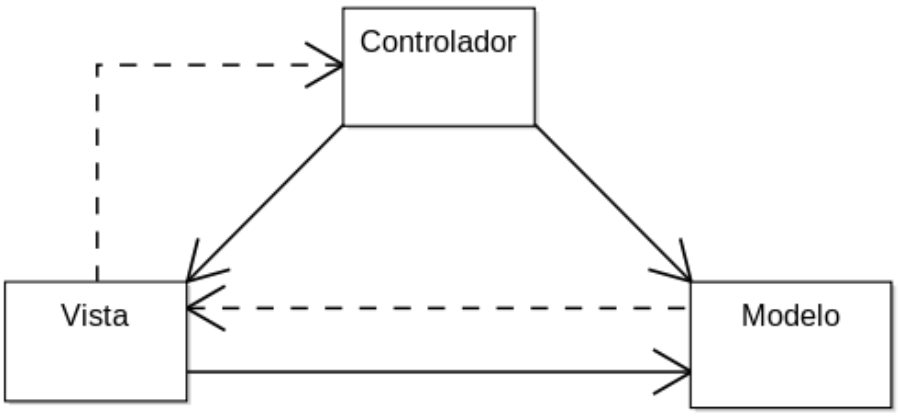

Figura C.1: Diagrama Modelo-Vista-Controlador

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se va a definir todo aquello que es necesario conocer para que se pueda continuar con el desarrollo del proyecto, desde su estructura hasta una breve descripción de como instalar la aplicación y configurar nuestro entorno de trabajo para llevar a cabo el desarrollo.

D.2. Estructura de directorios

La estructura del proyecto se divide en:

```
/ Directorio raíz
├── Documentacion/ -Documentación del proyecto
│   ├── img/ -Imágenes de la documentación
│   ├── tex/ -Secciones de la documentación
│   ├── anexos.pdf -Anexos del proyecto
│   └── memoria.pdf -Memoria del proyecto
├── HolaMundo/ -App web básica
└── Metrominuto/ -Aplicación web
    ├── static/ -ficheros JavaScript
    └── templates/ -ficheros HTML
```

D.3. Manual del programador

En este apartado se explican los puntos a tener en cuenta por futuros desarrolladores que tengan la intención de mantener o mejorar el proyecto.

D.4. Aplicaciones utilizadas

Para el desarrollo de este proyecto se tuvieron en cuenta principalmente dos editores de texto y dos herramientas para mantener el control de versiones.

- **Visual Studio Code**
- **PyCharm**
- **GitHub**
- **GitCraken**

Después de analizar y configurar ambos editores de texto, se llegó a la conclusión de que era mucho mas cómodo y útil utilizar PyCharm, ya que ofrece una configuración mas sencilla, además de permitir importar diversas librerías de una manera mas amigable. También ofrece la posibilidad de seguir los estándares de programación *PEP8* y *ECMAScript*.

D.5. Instalación y configuración

Para la instalación del proyecto se explicarán los pasos a seguir en un sistema operativo de Linux, que en este caso se trata de la versión Linux Mint 18.3 Sylvia.

Python

Este proyecto está desarrollado con la versión 3.6.3. Python se puede descargar desde el siguiente enlace: https://www.python.org/downloads/

Instalación y configuración de PyCharm

Este *IDE* tiene distribución para Linux, además de permitir a estudiantes usar la versión profesional. Para su instalación podemos usar la [**?**] de

Linux, que en caso de no tenerla instalada tenemos que ejecutar el siguiente comando:

Listing D.1: Instalar snapd

```
$ sudo apt update
$ sudo apt install snapd
```

Después de tener instalado esto, ejecutaríamos:

Listing D.2: Instalar PyCharm

```
$ sudo snap install
        pycharm-community|professional --classic
```

Una vez instalado *PyCharm*, la forma más cómoda de obtener el código del proyecto es mediante *git*, usando para ello el comando:

Listing D.3: Descargar el repositorio

```
$ git clone <url_del_repositorio>
```

Siendo https://github.com/gpm0009/TFG_MetrominutoWeb.git la URL del repositorio.

Para instalar las dependencias ejecutar el comando:

Listing D.4: Instalar requirements.txt

```
$ pip install -r requirementes.txt
```

Claves de Google

Para la obtención de un *Google API KEY* es necesario obtener los credenciales en https://console.developers.google.com/apis/credentials. Después, dichos credenciales deben activarse para las APIs que utiliza este proyecto:

- Directions.
- Distance Matrix.
- Geocoding.
- Maps JavaScript.

- Places.

Una vez que la tenemos, debemos incluirla en el proyecto. Para ello:

Listing D.5: Añadir API_KEY

```
google_maps=googlemaps.Client(key='GOOGLE_API_KEY')
```

Además, no hay que olvidar incluirla en los templates:

Listing D.6: Añadir API_KEY a los templates

```
<script
 src="https://maps.googleapis.com/maps/api/
          js?key=API_KEY&libraries=places"
          type="text/javascript"</script>
```

También podemos incluirla como variable de entorno en nuestro editor. De esta manera nos aseguramos de no compartirla al realizar los commints en el control de versiones.
En este caso, en PyCharm se configura de la siguiente manera:

1. Abrir selector Run Configuration (arriba a la derecha)
2. Edit Configurations...
3. Environmental variables
4. Add or change variables, then click OK

TeXstudio

Esta herramienta para la compilación de documentación LaTeX permite la instalación de diccionarios para aplicar las reglas al texto. Para ello, debemos acceder a:

Listing D.7: Añadir diccionario

```
Options -> Configure TeXstudio
Language checking
```

Una vez ahí, vemos que nos ofrece dos opciones para buscar diccionarios: https://extensions.openoffice.org/de/search?f[0]=field_project_tags o https://extensions.libreoffice.org/extensions?getCategories=Dictionary&getCompatibility=any. Elegimos cualquiera de ellas y descargamos el paquete del diccionario que queramos, y después lo importamos.

D.6. Librerías

NetworkX

Como ya he mencionado, esta biblioteca nos permite trabajar de una forma muy amplia y completa con grafos. A lo largo del proyecto se han usado:

- `graph()`: para crear un grafo no dirigido al que se añadirán nodos y arcos.
- add_node(): Diferentes nodos junto con atributos como la posición obtenida del API de Google y el nombre.
- add_edge(): Arco que conecta dos nodos. Además, los arcos contienen atributos como la distancia real que hay de nodo a nodo o el número de votos que tendrá.
- get_edge_attributes(): para obtener los valores de un atributo perteneciente a los arcos. Devuelve una lista que contiene el nodo de origen, el nodo destino y el atributo deseado.
- edges(data$\bar{\text{T}}$rue): devuelve los arcos junto con los atributos.
- nodes(data$\bar{\text{T}}$rue): devuelve los nodos junto con los atributos.
- minimum_spanning_edges(): devuelve un iterador con los arcos que forman el grafo de tal manera que la suma de distancias es la mínima.
- draw_networkx(): para dibujar el grafo.

D.7. Compilación, instalación y ejecución del proyecto

D.8. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario