



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Generador automático de
Metrominuto**



Presentado por Guillermo Paredes Muga
en la Universidad de Burgos
el 29 de junio de 2020

Tutores

Dr. Álvar Arnaiz González

Dr. César Ignacio García Osorio



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



Los doctores Álvar Arnaiz González y César Ignacio García Osorio, profesores del departamento de Ingeniería Informática, Área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Guillermo Paredes Muga, con DNI 13174210-V, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Metrominuto.

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

En Burgos, 29 de junio de 2020

Vº. Bº. del Tutor:

Dr. Álvar Arnaiz González

Vº. Bº. del co-tutor:

Dr. César Ignacio García Osorio

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice general	III
Índice de figuras	V
Índice de tablas	VI
Introducción	1
Objetivos del proyecto	3
Conceptos teóricos	5
3.1. Mapa de tránsito	5
3.2. Metrominuto	7
3.3. Teoría de grafos	10
3.4. Scalable Vector Graphics (SVG)	11
Técnicas y herramientas	13
4.1. Técnicas	13
4.2. Herramientas	14
4.3. Bibliotecas	21
4.4. Despliegue	23
Aspectos relevantes del desarrollo del proyecto	25
Trabajos relacionados	33
Conclusiones y Líneas de trabajo futuras	39

Índice de figuras

3.1.	Mapa del metro de Londres de 1928.	6
3.2.	Mapa del metro de Londres de 1933.	7
3.3.	Metrominuto de Pontevedra	8
4.4.	Lucidchart.	15
4.5.	Logo Metrominuto.	16
4.6.	Planes de Firebase.	18
4.7.	Inicio de sesión.	19
4.8.	Control de usuarios.	19
5.9.	Grafo inicial.	27
5.10.	Grafo inconexo.	28
5.11.	Puntos en el mapa.	30
5.12.	Tiempo entre los puntos.	31
6.13.	Metrominuto de Pontevedra.	34
6.14.	Pasominuto.	35
6.15.	Metroplaya.	37
6.16.	Metro Pie de Santander.	38

Índice de tablas

Introducción

Hoy en día la movilidad en los centros urbanos es mayoritariamente motorizada y ocupa alrededor del 70 % del espacio público. Es por ello, que en las ciudades o áreas urbanas, a excepción de algunas áreas residenciales, las zonas peatonales escasean, impidiendo, al menos dificultando, los desplazamientos a pie. Como solución a este problema, surge en la localidad de Pontevedra la idea de Metrominutoo [12].

Un Metrominuto consiste en un mapa sinóptico que une diferentes puntos de la ciudad en función de la distancia existente entre cada uno de ellos, y cuyo propósito fundamental es justamente promover que la gente camine y se desplace a pie. Nuevamente, como mencionaba arriba, para que esto pueda hacerse realidad el propio diseño de las áreas urbanas influye notablemente.

Actualmente ya existen ciudades con Metrominutos como Pontevedra (pionera en esta idea), Sevilla, Madrid o León, pero este proyecto lo que trata es de automatizar este proceso de creación de mapas de manera que es el propio usuario quien selecciona los puntos que van a aparecer en él.

Este proyecto se centra en el desarrollo de una aplicación web para la generación automática de estos mapas sinópticos, o mejor dicho, Metrominutos. De este modo, el propio usuario será el encargado de decidir qué puntos aparecen en el mapa y personalizarlo a su antojo: podrá mover puntos, cambiar los nombres si fuese necesario o elegir qué trayectos aparecen o no. Así, el usuario podrá generar en cualquier momento un mapa que se adapte a su necesidad real, eliminando elementos que no son de su interés o que, simplemente, no quiere visitar.

Objetivos del proyecto

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

Objetivos principales

- **Generar Metrominutos de manera automática.**
 - Generar un grafo no dirigido a partir de los puntos seleccionados.
 - Manipulación de los grafos para calcular distancias y posiciones.
 - Generar mapas sinópticos.
- **Desarrollar una aplicación web para la generación y edición de Metrominutos.**
 - Desarrollar una aplicación web en la que los diferentes usuarios puedan seleccionar diversos puntos en un mapa (ciudad) con el fin de visualizar al mismo tiempo las opciones existentes para llegar a los distintos puntos seleccionados.
 - Visualizar los puntos con el API de Maps de Google [4].
 - Visualizar los puntos seleccionados: información acerca de la ubicación, marcador en el mapa.
 - Permitir al usuario añadir y eliminar líneas (conexiones) entre puntos.
 - Permitir al usuario interaccionar con el mapa.

Objetivos técnicos

- Desarrollar una aplicación cliente-servidor en Python utilizando Flask.
- Hacer uso de las APIs de Google para obtener la localización de puntos de interés sobre sus mapas y para obtener datos sobre las distancias entre ellos.
- Hacer uso de Git como sistema de control de versiones del proyecto.
- Aplicar la teoría de grafos.

Objetivos a nivel personal

- Realizar una aportación al turismo derivada de una necesidad personal.
- Poner en práctica los conocimientos adquiridos durante el Grado para el correcto desarrollo de este trabajo.
- Adquirir nuevos conocimientos acerca del uso de APIs y servicios web proporcionados por plataformas Cloud como Azure o Google.

Conceptos teóricos

Aquí se explican los conceptos teóricos en los que se basa el proyecto. Aborda diferentes aspectos, desde los mapas sinópticos de tránsito a la teoría de grafos, así como su dibujado.

3.1. Mapa de tránsito

Un mapa de tránsito consiste en un mapa topológico y esquemático utilizado para mostrar trayectos y estaciones en el ámbito urbano, como puede ser el metro o el autobús. Los elementos principales de este tipo de mapas son:

- Líneas de diferentes colores y grosores que indican las distintas líneas del medio de transporte en cuestión.
- Iconos o puntos que indican las paradas o estaciones del medio en el que se vaya a viajar.
- Diferentes iconología para señalar características significativas.

Como aplicación para este proyecto, las estaciones o paradas del mapa corresponderán con los puntos seleccionados por el usuario sobre el mapa, y las líneas serán las distancias correspondientes entre dichos puntos.

Harry Beck

Harry Beck fue un ingeniero electrónico del metro de Londres que trabajaba diseñando diagramas del circuito eléctrico, y que comenzó a diseñar un

nuevo mapa de las líneas y estaciones de metro de su ciudad. El objetivo de la solución estaba claro: tenía que ser sencillo de leer para el público y que este pudiese reconocer claramente las distintas estaciones, salidas y traslados. Realizó varias versiones antes de llegar a la que conocemos hoy en día, como por ejemplo las que podemos observar en las imágenes 3.1 y 3.2.

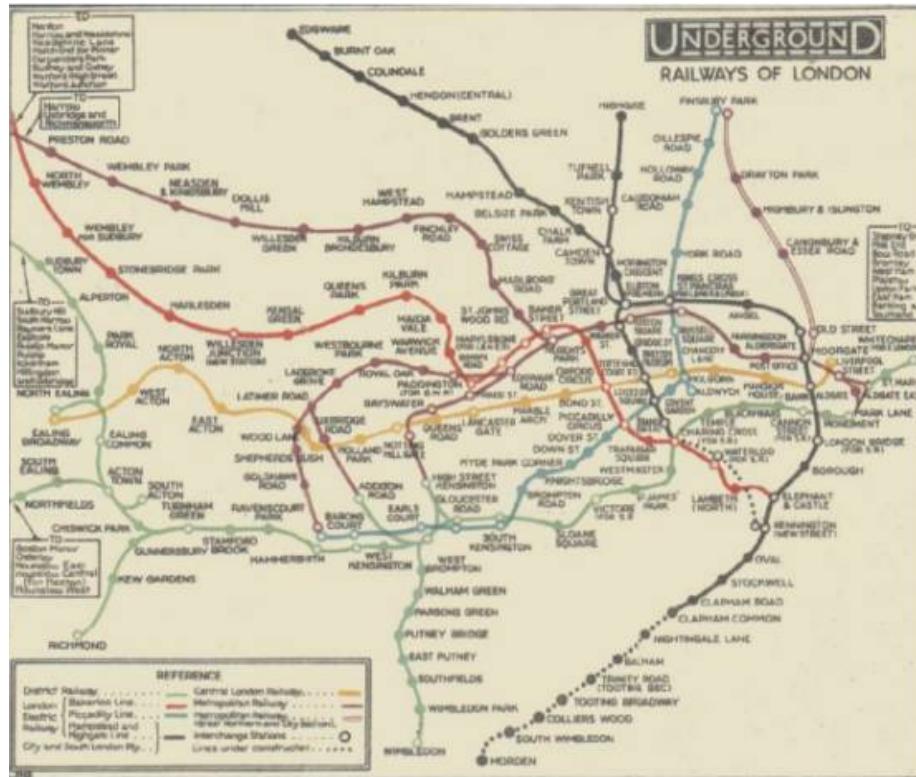


Figura 3.1: Mapa del metro de Londres de 1928.



Figura 3.2: Mapa del metro de Londres de 1933.

3.2. Metrominuto

El concepto de Metrominuto surgió como resultado de diversas ideas sobre movilidad en la ciudad de Pontevedra. Este concepto hace referencia a un mapa sináptico, como si de un mapa de metro se tratase, que representa las distancias y los tiempos existentes entre los diferentes puntos de una ciudad.



Figura 3.3: Metrominuto de Pontevedra

Metrominuto no solo ofrece información de cara a la gente que quiere visitar la ciudad, sino que también fomenta caminar como medio de transporte en una ciudad, donde de una manera sencilla y curiosa nos muestra como llegar de un sitio a otro. Caminar, como ya sabemos, es la mejor solución para evitar el gran flujo de automóviles en el área urbana, y lo que ello conlleva: una constante emisión de elementos contaminantes.

En los orígenes de este sistema de movilidad se encuentra el estudio, por medio de la técnica DAFO (Debilidades, Amenazas, Fortalezas, Oportunidades):

Debilidades: Como el estado cambiante del tiempo, diferente ritmo al caminar dependiendo de las personas, y la comodidad de coger el coche para moverse.

Amenazas: Prejuicios de la población.

Fortalezas: Cuidado del medio ambiente, mayor salud y al reducir los desplazamientos en automóvil se produce como resultado una mayor seguridad en los pocos que haya.

Oportunidades: Mejorar la ciudad, bienestar.

Estos planos no solo nos incitan a caminar, sino que también incluyen información útil acerca de líneas de autobús, estaciones de ferrocarril o de metro.

Proceso de elaboración

Los pasos a seguir para crear un *Metrominuto* de forma manual son los siguientes:

Paso 1: Consiste en la selección, dentro de una ciudad, de los puntos que se quieren representar en el mapa. Estos puntos pueden elegirse en función de su importancia, interés turístico o de los ciudadanos.

Paso 2: Decidir qué ruta peatonal es la más adecuada para unirlos.

Paso 3: Considerar cómo se va a dibujar el mapa. Puede ser más o menos preciso respecto a la realidad cartográfica.

Paso 4: Situar un punto central que sirva como punto de origen y de orientación para todos los usuarios.

Paso 5: Realizar por medio de herramientas de mapas, como Google Maps en nuestro caso o los mapas de Bing, el cálculo de las distancias entre los diferentes puntos.

Paso 6: Establecer una relación entre las distancias con el tiempo medio que lleva recorrerlas. Tenemos que tener en cuenta que toda la población no camina al mismo ritmo.

Paso 7: Una vez establecidas las diferentes rutas, hacer un estudio sobre ellas para corregir errores que puedan surgir, así como la variación en el tiempo si el terreno no es uniforme o si las condiciones de tráfico y semáforos varía.

Paso 8: Reflejar accidentes naturales o elementos de la ciudad como parques, costa, ríos... A través de elementos muy sencillos y con un código de colores al que estamos acostumbrados.

Paso 9: Reflejar aspectos de la movilidad intermodal, es decir, elementos como estaciones de metro, autobús, tren, etc.

Paso 10: Advertir de los espacios con condiciones adversas para personas con problemas de movilidad.

Paso 11: Simplicidad, claridad y facilidad de lectura a la hora de dibujar el mapa.

Paso 12: No sólo mostrar conexiones con el punto central establecido como referencia, sino que también debe aparecer información sobre la interconexión entre los diferentes puntos.

El objetivo de este proyecto es automatizar este proceso, recogido en el documento publicado por el Concello de Pontevedra [12], ya que actualmente los Metrominutos existentes se realizan de esta forma totalmente manual. Con el proceso automatizado sería el mismo usuario quien realice su propio Metrominuto con los puntos de interés personalizados que él decida. Evitando que aparezcan puntos o información que no le resulta interesante.

3.3. Teoría de grafos

Es aquella rama de las matemáticas que, junto con la ciencia de computación se encarga de estudiar las propiedades de los grafos. Primeramente debemos saber que un grafo $G=(V,E)$ es un conjunto de vértices o nodos unidos por enlaces llamados arcos. Existen varios tipos de grafos, pero en ese proyecto se han usado grafos no dirigidos para realizar las distintas operaciones con los nodos, es decir, los nodos corresponden con los puntos o lugares que selecciona el usuario y los arcos hacen referencia a la distancia existente entre ellos [23].

Uso de grafos

El resultado final de este proyecto es un mapa sinóptico formado por lugares y el «camino» o distancia entre ellos.

Dicho mapa se genera tras realizar varias operaciones (listadas a continuación) con grafos, donde los nodos serán los lugares previamente seleccionados y los arcos los trayectos que unen dichos lugares.

Paso 1: Calcular las distancias de todos con todos.

Paso 2: Eliminar un nodo de la lista.

Paso 3: Minimum spanning tree [25] del grafo obtenido en el paso 2.

Paso 4: Sumar un voto a cada arco resultante.

Paso 5: Repetir pasos 2, 3 y 4 hasta que se haya eliminado cada nodo del grafo una única vez.

Este proceso se detalla en la sección de aspectos relevantes del proyecto (ver [Calculo de grafos](#)).

3.4. Scalable Vector Graphics (SVG)

SVG o gráficos vectoriales escalables, es un término que hace referencia a un formato de gráficos vectoriales bidimensionales bien sean estáticos o dinámicos en formato XML [21]. Permite tres tipos de elementos:

1. Elementos geométricos vectoriales, como líneas, rectas o círculos.
2. Imágenes de mapa de bits/digitales.
3. Texto.

Las ventajas que presenta SVG son que a estos elementos se les pueden aplicar diferentes estilos, agrupar o transformar, bien sea antes de la compilación o dinámicamente.

Este formato de representación gráfica será el elegido para el dibujado de los mapas sinópticos, ya que con librerías como [SVGWRITE](#) y [Snap.svg](#) podemos manipular los distintos elementos.

Técnicas y herramientas

4.1. Técnicas

En esta sección se explicarán las distintas técnicas utilizadas a lo largo del proyecto.

Scrum

Scrum es una metodología de desarrollo ágil la cual proporciona un marco de trabajo y desarrollo de productos. No es un solo proceso, sino que en esta metodología se aplican un conjunto de buenas prácticas y procesos para que el producto final sea de la mejor calidad posible. El principal elemento del Scrum consiste en los llamados Sprints, que son ciclos de trabajo de una semana de duración. Este periodo sirve para producir un desarrollo o mejora del producto final. Estos sprints están marcados por dos reuniones:

- Planificación: en ella se presentan los requisitos o avances que tiene que cumplir el proyecto, a la vez que se estiman los tiempos y se realiza la planificación.
- Reunión de revisión: entrega de los requisitos acordados en la reunión de planificación y el equipo analiza el sprint.

El uso de esta metodología, junto con las diversas reuniones que se realizan, permite que el producto final sea de mejor calidad ya que en todo momento se conoce el feedback del cliente y se pueden realizar distintos cambios incrementales a medida que avanza el proyecto. Es una metodología pensada para el trabajo en equipo, por lo que en este proyecto se han mantenido

las bases pero se ha adaptado la forma de trabajar, de manera que las reuniones han sido entre los tutores y el alumno y la fecha de la reunión de planificación del Sprint coincide con la fecha de revisión del sprint anterior.

4.2. Herramientas

En esta sección se explicarán las herramientas utilizadas en el desarrollo del proyecto.

Estándar Python Enhancement Proposal 8 – PEP8

En este proyecto se ha seguido la guía de estilo *PEP8*¹, guía única que define cómo debería estar escrito el código *Python* y la forma de nombrar variables, funciones, clases o los comentarios del mismo.

Entorno de desarrollo Integrado (IDE)

Para el desarrollo del proyecto, se valoraron inicialmente dos editores:

- Visual Studio Code².
- PyCharm³.

PyCharm es un IDE desarrollado por la empresa JetBrains para el lenguaje Python. Existen dos ediciones, *Community* y *Professional*. Gracias al programa For Students⁴ de JetBrains se está usando la versión *Professional* en el proyecto, aunque la versión *Community* podría usarse también para el desarrollo del proyecto..

Ofrece soporte para Flask de serie y soporte para desarrollo web mediante complementos, además del control de estándares en el lenguaje (tanto código como comentarios), la repetición de fragmentos de código o sugerencias en el indexado del mismo.

¹<https://www.python.org/dev/peps/pep-0008/>

²<https://code.visualstudio.com/>

³<https://www.jetbrains.com/pycharm/>

⁴<https://www.jetbrains.com/student/>

GitHub

Para el control de versiones de este proyecto he utilizado GitHub, que es un repositorio en línea que emplea Git. De esta manera tenemos acceso en línea a los diferentes cambios de nuestro proyecto. Git maneja los distintos archivos del proyecto como un conjunto de copias instantáneas.

Lucidchart

*Lucidchart*⁵ es un espacio de trabajo visual diseñado para la elaboración de distintos tipos de diagramas y gráficos. En el caso de este proyecto, se ha usado para realizar los diagramas de casos de uso, clases, y secuencia, pero tiene una gran variedad de plantillas como vemos en la siguiente figura:

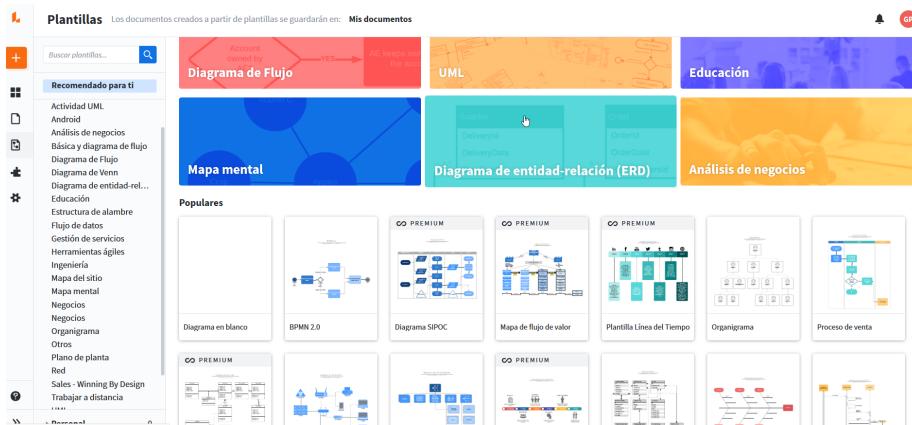


Figura 4.4: Lucidchart.

Además, permite iniciar sesión con nuestro usuario de la Universidad y exportar nuestro trabajo a distintos formatos: png, jpg, pdf ...

Adobe illustrator

*Adobe illustrator*⁶ como editor de gráficos, en este caso, para la creación del logo de la aplicación web.

⁵<https://www.lucidchart.com>

⁶<https://www.adobe.com/es/products/illustrator.html>

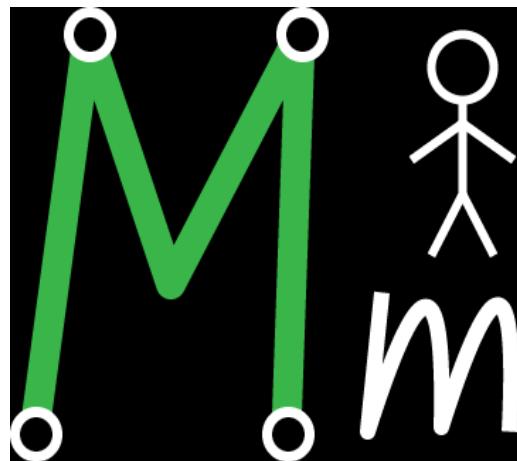


Figura 4.5: Logo Metrominuto.

TeXstudio

TeXstudio es un editor de código abierto para crear documentos L^AT_EX. Posee numerosas características para su desarrollo. Además, para su correcto funcionamiento es necesario instalar MiK_TeK.

Google API – Google Console

En este proyecto, para la selección de los distintos puntos a recorrer por parte del usuario he empleado los mapas de Google. Google proporciona una plataforma para los desarrolladores en la que se puede encontrar una gran cantidad de documentación⁷. Para poder integrar en la aplicación web tanto los mapas como las diferentes funcionalidades que ofrecen debemos adquirir lo que llama API Key⁸, la cual se trata de una clave «privada» para tener acceso a los servicios de su API. Para su obtención es necesario incluir los datos bancarios, ya que durante el primer año el uso de los servicios es gratis y luego comienza a pagarse a partir de un determinado número de peticiones. Una vez obtenida la clave, puede restringirse su uso para ciertas direcciones o dominios, de modo que puedes mantener el control de quien la usa. Además, no vale con conseguir una clave y ya está, sino que para usar los diferentes servicios que proporciona Google hay que activar diferentes APIs. Las APIs que se usan en este proyecto son:

⁷<https://cloud.google.com/maps-platform/>

⁸<https://developers-dot-devsite-v2-prod.appspot.com/maps/documentation/geocoding/get-api-key>

- **Maps JavaScript API:** Se utiliza en el cliente, de manera que se muestra el mapa al cargar la página y permite realizar diferentes acciones en él; tales como buscar, seleccionar puntos o moverte a través de él. Algunas de estas acciones implican el uso de algunas de las funcionalidades que proporcionan las APIs explicadas a continuación.
- **Geocoding API:** este API consta de dos elementos:
 - Geocodificación: Consiste en convertir direcciones en coordenadas.
 - Geocodificación inversa: Consiste en convertir coordenadas en una dirección legible.
- **Places API:** este servicio devuelve como resultado de la petición toda la información acerca de un lugar.
- **Distance Matrix API:** este API proporciona tanto la distancia como el tiempo de viaje que hay entre una lista de orígenes y una de destinos. En otras palabras, como resultado devuelve la distancia y tiempo que hay entre cada origen y cada destino.
- **Directions API:** como respuesta nos devuelve las indicaciones a seguir para llegar desde el punto de inicio hasta el punto de destino. Además, puede configurarse para diferentes modos de transporte, diferentes momentos de salida o llegada.

Firebase – Firebase Console

Firebase ⁹ es una plataforma móvil creada por Google para el desarrollo de aplicaciones, disponible para IOS, Android y web. Su función principal es facilitar la creación de aplicaciones de elevada calidad de una más rápida, permitiendo añadir una base de usuarios que mejore la rentabilidad económica, además de la posibilidad de añadir funciones de estadísticas, bases de datos o mensajería.

Además, cuenta con varios planes para adecuar el precio con las funcionalidades que queramos incluir. En este proyecto he incluido sólo el servicio de autenticación, que forma parte del *Plan Spark* ¹⁰, el cual es gratuito e incluye, entre otros, los servicios que vemos en la figura 4.6.

⁹<https://firebase.google.com/>

¹⁰<https://firebase.google.com/pricing>

Productos	Sin cargo	Pago por uso
	Plan Spark Generous limits to get started	Plan Blaze Calcula los precios de las apps a gran escala. ✓ Se incluye el uso gratuito del plan Spark*
Pruebas A/B	Sin cargo	
Analytics	Sin cargo	
App Distribution	Sin cargo	
App Indexing	Sin cargo	
Authentication		
Autenticación telefónica: Canadá, EE.UU. y la India ?	10,000 por mes	USD 0.01 por verificación
Autenticación telefónica: Todos los demás países ?	10,000 por mes	USD 0.06 por verificación
Otros servicios de autenticación	✓	✓

Figura 4.6: Planes de Firebase.

FirebaseUI (User Interface) for Web – Firebase Auth

FirebaseUI, desarrollado sobre *Firebase Auth*¹¹ es una librería de código abierto para la web que proporciona componentes simples y personalizables además de los *software development kit* o SDKs, que permiten la eliminación de código repetido.

Proporciona soluciones de autenticación integrada de manera que con unas pocas líneas de código podemos iniciar sesión con direcciones de correo electrónico y contraseñas, números de teléfono o proveedores como Google, Facebook, GitHub, Twitter, Apple, Microsoft, Yahoo ... Además, también nos proporciona la posibilidad de registrarse en nuestra aplicación o casos de recuperación de contraseñas. en la figura 4.7 podemos apreciar cuál sería el resultado de incluirlo en nuestra aplicación.

¹¹<https://firebase.google.com/docs/auth>

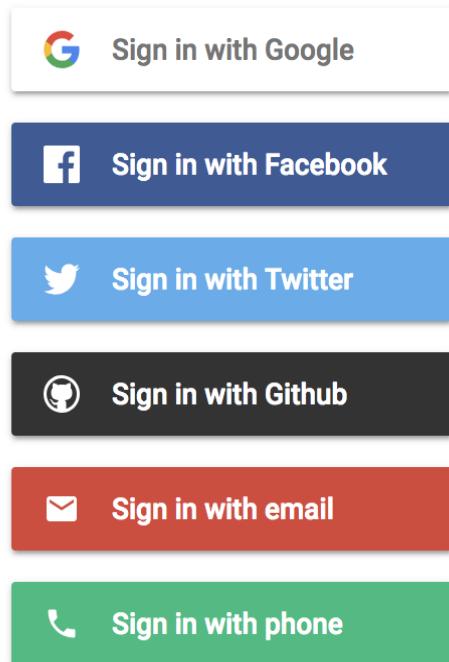


Figura 4.7: Inicio de sesión.

Como mencionaba arriba, estas funcionalidades que conllevan muy pocas líneas, dotan a nuestra aplicación de seguridad y nos permite tener un control absoluto sobre qué usuarios acceden a ella, como se ve en la figura 4.8.

Identifier	Proveedores	Creado	Accediste a tu cuenta	UID de usuario
rollerguille@gmail.com		13 jun. 2020	19 jun. 2020	4MWo58WjzAQY6z2WpXUV9U...
user-prueba@outlook.es		14 jun. 2020	19 jun. 2020	WU5prmlNsZkchygE0Jg3TbCO...

Figura 4.8: Control de usuarios.

Frameworks

El término *framework* [22] significa entorno o marco de trabajo, y consiste en una estructura conceptual y tecnológica que puede servir de base para el desarrollo software. Incluye programas, bibliotecas y lenguajes interpretados.

En este proyecto se han usado para el desarrollo del mismo los frameworks **Flask**, **Jinja2** y **Vue.js**.

Flask

Flask ¹² es un *framework* de Python que nos permite crear aplicaciones cliente-servidor de una manera más sencilla, y que no impone ninguna limitación respecto a estructura del proyecto, ni a los componentes que usar durante el desarrollo [14], aunque cabe destacar que en este proyecto se ha seguido la estructura planteada por *Azure* para poder realizar el despliegue.

Jinja2

Para que *Flask* pueda hacer uso de los contenidos *HTML* requiere de la utilización de este motor de plantillas, *Jinja2* ¹³, que consiste en un lenguaje de que permite insertar datos procesados y texto predeterminado en una página *HTML* mediante las etiquetas `{{variable}}` ó `{% expresión %}`.

Vue.js

Vue ¹⁴ es un framework progresivo utilizado para construir interfaces de usuario. Está enfocado únicamente a la capa de visualización, por lo que resulta sencillo integrarlo con otras librerías o incluso en proyectos ya existentes, como es este caso. *Vue* ofrece la posibilidad de instalar el cliente mediante *Node.js*¹⁵, aunque también permite integrarlo directamente en la capa de visualización incluyendo en los templates del proyecto el *CDN*:

Vue.js cdn 4.1: Versión de desarrollo.

```
<script
src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"
</script>
```

o con

¹²<https://flask.palletsprojects.com/en/1.1.x/>

¹³<https://jinja.palletsprojects.com/en/2.11.x/>

¹⁴<https://vuejs.org/>

¹⁵<https://es.vuejs.org/v2/guide/installation.html>

Vue.js cdn 4.2: Versión de producción.

```
<script  
src="https://cdn.jsdelivr.net/npm/vue"  
</script>
```

La versión de desarrollo ofrece distintas alertas o advertencias que permiten realizar diferentes trazas a la hora de localizar posibles errores, mientras que la versión de producción está optimizada en cuanto a tamaño y velocidad. Para poder utilizar *Vue* en nuestros templates, además de incluir el *CDN* debemos crear un componente¹⁶, al que le añadiremos una variable. Esto se debe a que realmente la estructura y el nombre de los elementos de *Vue* es algo diferente a los que ya conocemos de *HTML5*. También es importante tener en cuenta que *Vue* utiliza los mismos delimitadores que *Flask*: `{{variable}}` . Es por esto por lo que a nuestro componente o variable debemos añadir la línea `delimiters: ['[[',']]']` para cambiarlos.

4.3. Bibliotecas

En este apartado se explicarán brevemente las bibliotecas utilizadas para en el desarrollo.

JQuery

JQuery [3] es una biblioteca JavaScript que hace que la manipulación de documentos HTML, el manejo de eventos y las peticiones Ajax sean más sencillas de usar y manipular. Además, una de sus ventajas principales es que funciona en prácticamente todos los navegadores web.

NetworkX

NetworkX es la biblioteca por excelencia de Python para trabajar con grafos y redes. Permite crear, manipular y estudiar su estructura.

Características:

- Estructuras de datos para grafos simples, grafos dirigidos y multigrafos.
- Contiene la gran parte de algoritmos utilizados para el estudio y modificación de grafos.

¹⁶<https://vuejs.org/v2/guide/components.html>

- Los nodos pueden ser «cualquier cosa» como por ejemplo texto, imágenes o números.
- Los arcos pueden tener diferentes atributos o datos, como el peso, distancia....
- Es de código abierto.

Tube Map - D3

Esta biblioteca permite dibujar mapas muy similares a los mapas que hoy podemos ver en el metro, con sus estaciones, paradas e intersecciones [1]. Se intentó implementar en el proyecto durante dos Sprints, pero se llegó a la conclusión que no se podía generar la estructura necesaria en el archivo JSON para el dibujado del mapa. Esta estructura debía contener coordenadas con números enteros y estar ordenadas de tal forma que se indicasen las esquinas y cruces que debía haber entre las diferentes líneas, o en este caso, recorridos.

SVGWRITE

SVGWRITE [7], como su propio nombre sugiere, sirve para la creación y dibujado de gráficos vectoriales escalables. Su principal inconveniente es que no permite leer o modificar un archivo ya existente, es decir, sólo permite su creación desde cero.

Snap.svg

Snap.svg [6] es una biblioteca JavaScript que permite crear gráficos vectoriales interactivos y fácilmente manipulables en el lado del cliente. Su uso es similar al uso de jQuery.

L^AT_EX

Como herramienta para realizar la documentación se ha escogido L^AT_EX [20], lenguaje orientado a la creación de documentos que presenten una alta calidad tipográfica.

MiK^TE_X

MiK^TE_X es una distribución libre de L^AT_EX que incluye tipografías compiladores y herramientas para generar bibliografías e índices.

4.4. Despliegue

La idea de plantear el proyecto como una aplicación web sugiere que ésta tiene que ser fácilmente accesible por los usuarios, cosa que no es cierta si para poder usarla tienen que descargarse el código, configurar el proyecto y ejecutarlo en su propio dispositivo. Por esta razón, valoré desplegar la aplicación en Azure¹⁷.

Elegí esta plataforma porque con el correo de la Universidad podemos tener acceso a un plan de estudiantes, llamado *Azure for Students*¹⁸, que permite, además de acceso a diversos tutoriales sobre computación en la nube, desplegar nuestra aplicación de manera gratuita, aunque por supuesto también tiene varios planes de pago.

Link de la aplicación: <http://tfgmetrominuto.azurewebsites.net/>.

¹⁷<https://azure.microsoft.com>

¹⁸<https://azure.microsoft.com/es-es/free/students/>

Aspectos relevantes del desarrollo del proyecto

En este apartado se van a recoger y explicar los aspectos más importantes del desarrollo del proyecto. Desde las implicaciones de las decisiones que se tomaron, hasta los numerosos y variados problemas a los que hubo que enfrentarse.

Elección del proyecto

El año pasado fui uno de los privilegiados de poder disfrutar de una beca Erasmus, en concreto con destino en la ciudad polaca de Gliwice [24]. Esto me llevó a conocer nuevas culturas, pero también a conocer nuevas ciudades, y en la mayoría de ocasiones el tiempo del que disponíamos para recorrerlas era muy breve. Por ello, al ver las posibilidades que ofrecía el resultado final de este TFG, me llamó la atención, ya que es una aplicación que de haberla tenido, nos habría ahorrado muchos desplazamientos, quizá inútiles al recorrer estas ciudades sin una ruta fija.

Formación

Para poder realizar el proyecto se necesitaban unos conocimientos no adquiridos sobre desarrollo web, tanto de la parte de servidor, en Flask, como la parte del cliente, en HTML, CSS y JavaScript. Además de para aprendizaje, los recursos se han usado también como material de consulta durante el desarrollo.

Para la parte del servidor se siguieron los siguientes libros y tutoriales:

- Flask Web Development [14].
- The Flask Mega-Tutorial (2017) [15].
- Deploy Python apps to Azure App Service on Linux from Visual Studio Code [17].
- Flask Tutorial in Visual Studio Code [18].

A medida que se añadían nuevas herramientas al proyecto, su documentación oficial también ha sido consultada en varias ocasiones, están disponibles en:

- W3Schools Tutorials [8].
- Documentación de Flask [2].
- Documentación de Bootstrap [10].
- Documentación de NetworkX [16].
- Tutorial FirebaseUI ¹⁹.

Calculo de grafos

Para calcular el mapa sinóptico se han empleado grafos, junto con la librería NetworkX para realizar las operaciones que se explican a continuación:

Paso 1: Calcular las distancias de todos con todos, grafo no dirigido al que llamaremos α : añadir un arco entre cada par de nodos del grafo, asignando a cada nodo la posición que ocupa en el mapa y a cada arco el valor de la distancia que representa. Estos datos de distancia y posición los obtenemos de los APIs de Google de *Places* ²⁰ y *Distance Matrix* ²¹.

Paso 2: Eliminar un nodo: eliminamos un nodo del grafo junto con sus arcos asociados.

¹⁹<https://github.com/firebase/FirebaseUi-Web#installation>

²⁰<https://developers.google.com/maps/documentation/distance-matrix/start?hl=es>

²¹<https://developers.google.com/places/web-service/intro?hl=es>

Paso 3: Minimum spanning tree del grafo obtenido en el paso 2: evaluamos el camino mínimo para recorrer los nodos que nos quedan en el grafo.

Paso 4: A cada arco obtenido en el grafo resultante del paso 3.

Paso 5: Guardamos en una lista los votos de cada arco.

Paso 6: Repetir pasos 2, 3 y 4 hasta que se haya eliminado cada nodo del grafo una única vez. De este modo, en la lista final de arcos tendremos el número de veces que aparece cada arco en el recorrido mínimo.

Paso 7: Para cada valor (ϵ) comprendido entre el mínimo y máximo de los votos anteriores:

1. Calculamos un subgrafo (β) con todos los nodos y aquellos arcos con un número de votos mayor o igual a ϵ .
2. Comprobamos que el subgrafo β sea conexo: comprobamos que todos los nodos estén conectados. En el caso de que no lo estén, mediante NetworkX obtenemos los subconjuntos de β y, recuperando el grafo inicial α , unimos los subconjuntos por los arcos de menor coste. Esto quiere decir, que si por ejemplo nuestro grafo inicial α es (imagen 5.9):

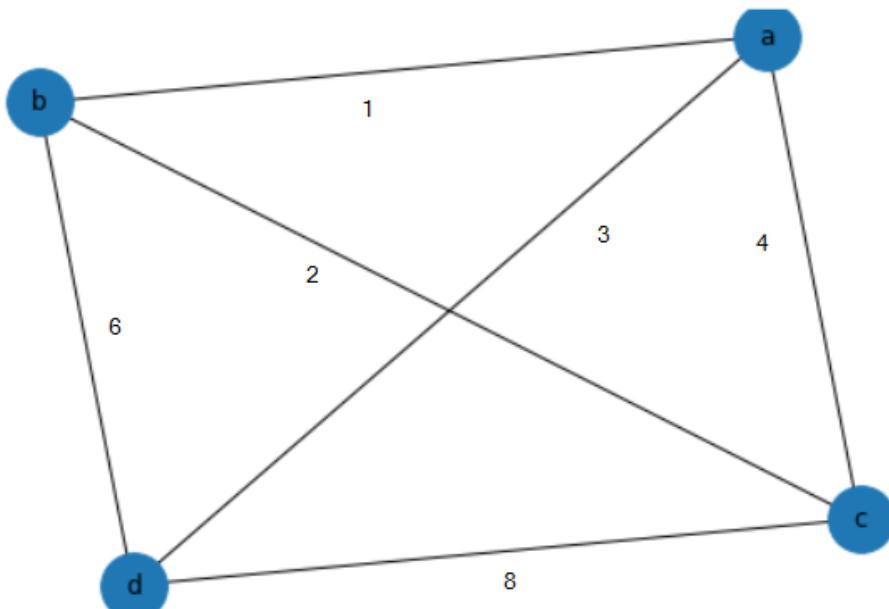


Figura 5.9: Grafo inicial.

y el resultado de comprobar si β es conexo o no es (imagen 5.10):

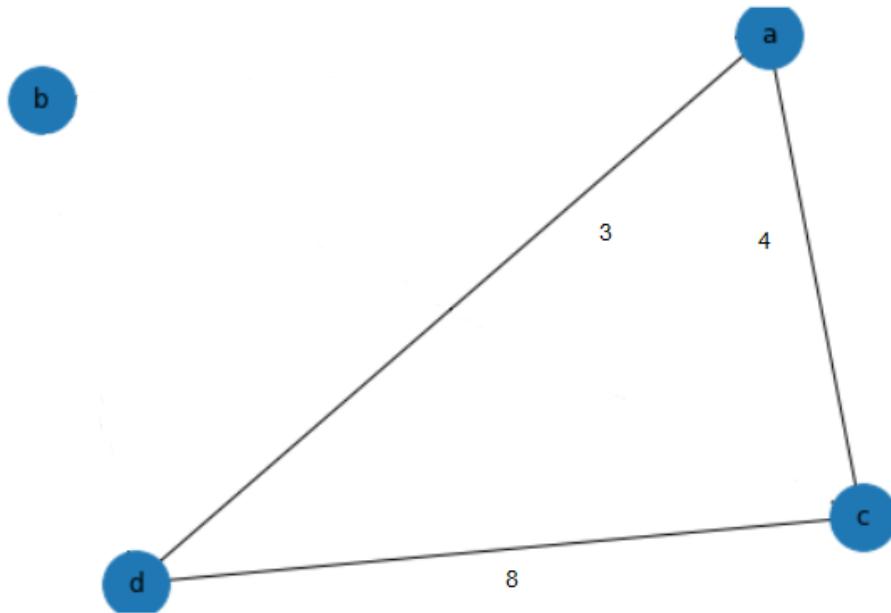


Figura 5.10: Grafo inconexo.

La unión entre ambos subconjuntos (nodo «b» con el resto del grafo) se realizaría por el arco que tenga una menor distancia, b-a.

3. Guardar el subgrafo conexo obtenido.

Paso 8: Generar el grafo SVG correspondiente a cada subgrafo.

Dibujado de grafos

El principal objetivo de este proyecto se basa en obtener un grafo final de manera que éste sea fácilmente entendible por todos. Es por ello que a la hora de dibujar dicho grafo se plantean cuestiones y problemas como:

- Dónde colocar los textos y a qué distancia de la línea.
- Cómo orientar esos textos.
- Evitar la superposición de los textos, tanto los referentes al punto como los referentes a la información del trayecto, con las líneas o puntos.

- Evitar la superposición de unos textos con otros.
- Tener en cuenta los colores a la hora de agrupar elementos.
- Mantener la posición de los puntos lo más similar posible a los marcados en el mapa.
- Repositionar los puntos que vertical u horizontalmente sean casi similares para evitar líneas diagonales.

Estos problemas han resultado de gran complejidad en el desarrollo final del proyecto ya que, como mencionaba antes, es el resultado final de la aplicación web diseñada y programada para el TFG. Para la resolución de cómo calcular la posición de los textos, se han tenido en cuenta dos opciones (explicadas en detalle en *Anexos–Manual del programador–Dibujado de grafos*):

1. **Discretización de las líneas:** Este método consiste en que, una vez hemos calculado el punto medio de la línea que une dos nodos, dividimos dicha línea en varios puntos separados una distancia δ (algo menor a la altura del recuadro que contiene al texto). Después, para cada posible posición del texto en la línea vertical, por el punto medio, a la que une los nodos, calculamos si en el interior del rectángulo que contiene al texto hay uno de los puntos que hemos calculado previamente. Si lo hay, pasamos a valuar la siguiente posición.
2. **Superposición de cuadrados:** Este método consiste en calcular primeramente la dirección de la línea, y una vez que la sabemos, calcular el punto medio. Después, construimos 4 cuadrados usando los extremos y el punto medio, de tal modo que descartamos los dos en los que la línea coincide con la diagonal. Los otros dos, serán las dos posibles «zonas» en las que podemos colocar el texto.

Un aspecto clave a tener en cuenta en el resultado final de la aplicación es que, a pesar de que la unión entre dos puntos es una línea recta, el recorrido entre ambos puntos en la realidad no tiene por qué ser recto. Dos puntos que parecen estar cerca el uno del otro pueden tener una distancia mayor que dos puntos que parecen más alejados. Para entender mejor esto fijémonos en la imagen 5.11. Vemos que los puntos relativos a el *Arco de San Martín* y *Mirador del Castillo* están más cerca el uno del otro, mientras que el punto relativo a *La Estación* se encuentra más alejado de ellos.

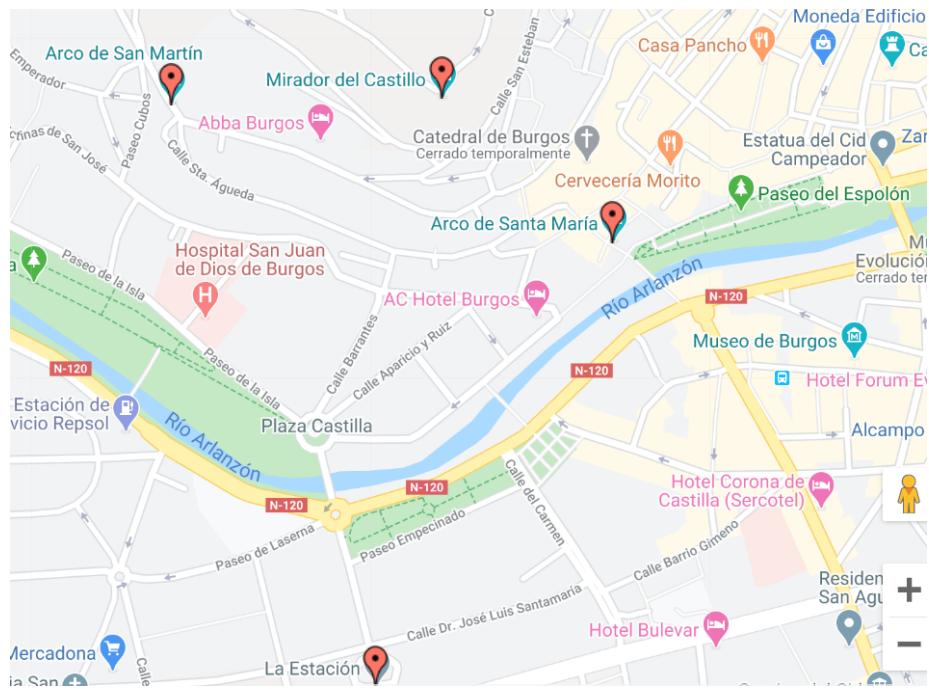


Figura 5.11: Puntos en el mapa.

Sin embargo, si nos fijamos en el mapa anterior podemos ver que para llegar desde el *Arco de San Martín* al *Mirador del Castillo* tenemos que dar un rodeo más largo de lo que parece, y puede ser el caso de que los puntos no se encuentren al mismo nivel (en términos de tiempo no es lo mismo hacer un kilómetro subiendo que un kilómetro bajando). Esto hace que, como vemos en la imagen 5.12, el tiempo que tardamos entre estos dos puntos y el que tardamos desde el primero a *La Estación*, que visualmente parece estar más lejos, sea el mismo.



Figura 5.12: Tiempo entre los puntos.

Decoradores de Flask

Los decoradores son funciones que envuelven y reemplazan a otra función, y al hacerlo, deben utilizar los valores de los argumentos de la función original a la nueva.

En Flask, cada vista o ruta es una función de Python, lo que significa que mediante decoradores podemos añadir funcionalidades adicionales. El

más utilizado es `@route()`, que sirve para especificar la ruta y qué tipo de peticiones admite.

En el desarrollo del proyecto se incluyó un decorador para comprobar, antes de acceder a cada vista, si el usuario había iniciado sesión o no mediante la comprobación de una variable de sesión, y en el caso de que no, redirigir a dicho usuario a la página de inicio. Se trata de algo similar a lo que hace algunos paquetes de Flask de validación de usuarios como Flask-Login [?], pero que en este proyecto se ha hecho de forma manual porque se ha hecho la validación de usuarios a través de Firebase.

Blueprints

Blueprints [5] o «planos» es un concepto usado en Flask para crear componentes y modular la aplicación. Su funcionalidad principal es proporcionar un módulo para registrar funciones en la aplicación.

Trabajos relacionados

En este apartado se explicarán y comentarán brevemente algunas aplicaciones de la idea original de Metrominuto, en las que la finalidad de todas ellas es el fomento del «arte de caminar» [11].

Metrominuto Pontevedra

Pionera en esta idea, la ciudad de Pontevedra fue la primera en publicar un metrominuto (ver figura 6.13). Hizo de caminar, un producto, convirtiéndolo en la mejor alternativa para evitar el uso de vehículos motorizados.

Además, cuenta con una cuenta de facebook <https://www.facebook.com/metrominuto/> en la que añaden distintas publicaciones relacionadas con su producto, o simplemente publicaciones de metrominutos de otras ciudades, a las que se ha extendido la idea, que según el artículo *Al menos 57 ciudades han copiado el Metrominuto pontevedrés* [9], ya son, como bien indica el título, más de cincuenta y siete ciudades las que ya tienen uno.



Figura 6.13: Metrominuto de Pontevedra.

Aplicaciones similares

Conforme iba pasando el tiempo, no solo copiaron la idea de metrominuto aplicándola al ámbito urbano, sino que también trasladándola a otros ámbitos. Es por esto que surgen una serie de aplicaciones o herramientas similares que veremos a continuación.

Pasominuto

Pasominuto: <https://pontevedraviva.com/web/uploads/archivos/pasominuto.pdf>. Esta alternativa a Metrominuto consiste en una guía de recorridos para pasear por «los espacios más agradables» de la ciudad, teniendo en cuenta sus distancias, pasos y tiempos. El artículo [19] menciona hasta un total de 29 recorridos diferentes.



Figura 6.14: Pasominuto.

FIXME: los espacios los mete solo no entiendo por qué. La imagen es realmente un pdf y entiendo que debería salir en la página entera, pero no.

Metroplayas

Aplicación considera como «hijo de metrominuto» y que sitúa a Pontevedra en el punto equidistante de las mejores playas de las *Rías Baixas*.

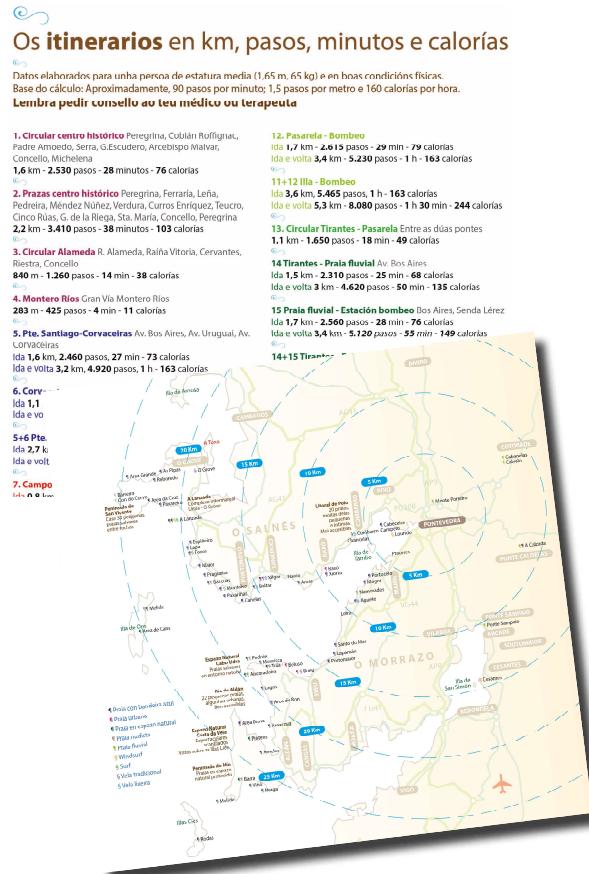


Figura 6.15: Metroplaya.

Metro Pie

Metro Pie consiste en un conjunto de trayectos predefinidos en los que se especifica el tiempo que se tarda en recorrerlos. ¿Su objetivo?. El mismo

que antes, fomentar los desplazamientos a pie. Como ejemplo, una de las ciudades que ha incorporado esta idea es Santander, como nos muestra el artículo *Santander lanza el Metropie: un mapa con itinerarios por la ciudad para fomentar los desplazamientos peatonales* [13].



Figura 6.16: Metro Pie de Santander.

Conclusiones y Líneas de trabajo futuras

En este apartado se exponen las conclusiones obtenidas tras finalizar el proyecto, además de explicar posibles líneas de mejora futuras para el mismo.

Conclusiones

Durante los ocho o nueve meses que he estado desarrollando este proyecto, también he estado trabajando fuera de la Universidad. Durante este tiempo, como se puede observar en el repositorio del proyecto²², el trabajo ha sido constante, y como consecuencia de ello, considero que la aplicación final de todo el trabajo puede llegar a ser realmente útil, no solo para fomentar los desplazamientos a pie en la propia ciudad, si no que también a la hora de viajar y conocer nuevas ciudades.

Hay que destacar también la gran cantidad de conocimientos nuevos aprendidos, no solo en cuanto a programación, sino también en cuanto a diseño, teoría de grafos, uso de APIs, uso de plataformas Cloud... y de la capacidad para reunir información o buscar documentación. He intentado usar funcionalidades y herramientas nuevas, como los *frameworks* y bibliotecas explicados en el apartado [Herramientas](#).

Finalmente, a nivel personal, me genera una gran satisfacción haber sido capaz de desarrollar, desde su planificación e inicio, hasta su finalización, una aplicación entregable, funcional y sobretodo, útil.

²²https://github.com/gpm0009/TFG_Metrominutoweb

Líneas de trabajo futuras

Como futuras evoluciones del proyecto, se plantean las siguientes ideas:

- Mejorar la seguridad de la aplicación: ocultar en la consola del navegador las claves mediante algún sistema de seguridad.
- Mejorar la tolerancia a fallos: mejorar la respuesta de la aplicación ante fallos o errores.
- Añadir una base de datos: con esta mejora podríamos conseguir personalizar la aplicación para cada usuario, de tal modo que este pueda guardar y editar en cualquier momento sus metrominutos.
- Mejorar la apariencia y maquetación de la aplicación: mejorar los estilos de la aplicación.
- Añadir decoración al mapa sinóptico como estaciones de transporte público, parques, lagos, ríos o zonas de costa, en el caso de que existan.
- Añadir geolocalización (provista por otro API de Google, Geolocation²³): de este modo el usuario puede identificar fácilmente en qué punto del recorrido se encuentra.
- Permitir al usuario en el SVG agrupar zonas, dibujar parques o añadir elementos que considere interesantes.
- Tras la modificación de la posición de textos o puntos por parte del usuario, dar la opción al usuario de recuperar el «magnetismo» entre textos y nodos manteniendo la posición actual de los mismo, de tal modo que si vuelve a mover el punto, los textos se muevan junto a él.

²³<https://developers-dot-devsite-v2-prod.appspot.com/maps/documentation/geolocation/intro>

Bibliografía

- [1] D3 tubemap documentation. <https://www.npmjs.com/package/d3-tube-map/v/1.2.0>.
- [2] Flask documentation. <http://flask.pocoo.org/docs/1.0/>.
- [3] Jquery documentation. <https://api.jquery.com/>.
- [4] Maps javascript api. <https://developers.google.com/maps/documentation/javascript/tutorial>.
- [5] Modular applications with blueprints. <https://flask.palletsprojects.com/en/1.1.x/blueprints/>.
- [6] Snap.svg documentation.
- [7] Svgwrite documentation. <https://svgwrite.readthedocs.io/en/latest/overview.html>.
- [8] W3Schools. <https://www.w3schools.com/>.
- [9] Serafín Alonso. Al menos 57 ciudades han copiado el metrominuto pontevedrés. *Diario de Pontevedra*.
- [10] Bootstrap Contibutors. Bootstrap documentation. <https://getbootstrap.com/docs/3.3/>.
- [11] Concello de Pontevedra. Metrominuto. <http://metrominuto.pontevedra.gal/es>.
- [12] Concello de Pontevedra. Haz tu propio Metrominuto. 2013.

- [13] eldiario.es Cantabria. Santander lanza el metropie: un mapa con itinerarios por la ciudad para fomentar los desplazamientos peatonales. *eldiario.es Cantabria*, 2020. 10/06/2020.
- [14] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, 2014.
- [15] Miguel Grinberg. The Flask Mega-Tutorial. <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>, 2017.
- [16] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.
- [17] Microsoft. Deploy Python apps to Azure App Service on Linux from Visual Studio Code. <https://docs.microsoft.com/en-us/azure/developer/python/tutorial-deploy-app-service-on-linux-01>.
- [18] Microsoft. Flask Tutorial in Visual Studio Code. <https://code.visualstudio.com/docs/python/tutorial-flask>.
- [19] Oskar Viéitez. El pasominuto y las 10.000 zancadas por pontevedra. *Pontevedraviva.com*.
- [20] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 30-septiembre-2015].
- [21] Wikipedia. Extensible markup language — wikipedia, la enciclopedia libre, 2020. [Internet; descargado 11-marzo-2020].
- [22] Wikipedia. Framework — wikipedia, la enciclopedia libre, 2020. [Internet; descargado 21-junio-2020].
- [23] Wikipedia. Teoría de grafos — wikipedia, la enciclopedia libre, 2020. [Internet; descargado 20-marzo-2020].
- [24] Wikipedia contributors. Gliwice — Wikipedia, the free encyclopedia, 2020. [Online; accessed 11-March-2020].
- [25] Wikipedia contributors. Minimum spanning tree — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Minimum_spanning_tree&oldid=960990730, 2020. [Online; accessed 27-June-2020].