



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA  
SUPERIOR  
Gº en Ingeniería Informática



**TFG Ingeniería Informática:**

**GII17.0T Trayectorias Semánticas**



Presentado por Hector Cogollos Adrian  
en Burgos el 3 julio de 2019  
Tutores D. Bruno Baruque Zanón  
y D. Santiago Porras Alfonso

---

# Índice General

---

Índice General .....	1
Índice de figuras .....	4
Índice de Tablas.....	6
Plan de Proyecto Software .....	7
A.1.    Introducción .....	7
A.2.    Planificación temporal .....	7
Sprint 1 .....	7
Sprint 2 .....	8
Sprint 3 .....	8
Sprint 4 .....	8
Sprint 5 .....	8
Sprint 6 .....	9
Sprint 7 .....	9
Sprint 8 .....	9
Sprint 9 .....	9
Sprint 10 .....	10
Sprint 11 .....	10
Sprint 12 .....	10
Sprint 13 .....	10
Sprint 14 .....	11
Sprint 15 .....	11
A.3.    Estudio de viabilidad.....	12
Viabilidad económica.....	12
Viabilidad legal .....	13
Especificación de Requisitos.....	14
B.1.    Introducción .....	14
B.2.    Objetivos generales .....	14
B.3.    Catálogo de requisitos .....	14
Requisitos funcionales.....	14

Requisitos no funcionales.....	15
Limitaciones Funcionales.....	15
B.4. Especificación de requisitos.....	16
Actores.....	16
Diagramas de casos de uso .....	16
Especificación de los casos de uso .....	17
Especificación de diseño .....	22
C.1. Introducción .....	22
C.2. Diseño de datos .....	22
C.3. Diseño arquitectónico.....	23
Modelo.....	24
Controlador.....	25
Vista.....	26
Base de Datos .....	27
C.4. Diseño procedimental.....	28
SQL .....	28
Predicción.....	29
Vista.....	30
Documentación técnica de programación .....	32
D.1.Introducción.....	32
D.2. Estructura de directorios .....	32
D.3. Manual del programador .....	33
Requisitos del sistema .....	33
Desarrollo de la aplicación .....	34
Instalación .....	35
Aplicación .....	38
D.4. Pruebas del sistema .....	41
Pruebas de detección de paradas .....	41
Pruebas clasificador.....	43
Pruebas clustering.....	44
Documentación de usuario .....	47
E.1. Introducción .....	47
E.2. Requisitos de usuarios.....	47
E.3. Instalación .....	47

E.4. Manual del usuario .....	48
Cargar .....	48
Predicciones.....	51
Exportar .....	56

---

## Índice de figuras

---

Ilustración 1 Casos de uso Usuario .....	16
Ilustración 2 Casos de uso Administrador .....	17
Ilustración 3 Base de Datos .....	22
Ilustración 4 Paquetes MVC.....	24
Ilustración 5 Paquetes Modelo .....	24
Ilustración 6 Paquetes Controlador .....	26
Ilustración 7 Paquetes Vista .....	27
Ilustración 8 Paquetes Base de Datos.....	28
Ilustración 9 Paquete SQL.....	29
Ilustración 10 Paquete Predicción .....	30
Ilustración 11 Paquete Vista y Run .....	31
Ilustración 12 Estructura de directorios.....	32
Ilustración 13 Debug .....	35
Ilustración 14 Configuración Aplicación Base de Datos .....	40
Ilustración 15 Configuración IP Nominatim .....	41
Ilustración 16 Configuración IP Salida .....	41
Ilustración 17 Trayectoria en Bruto.....	42
Ilustración 18 Trayectoria Conceptual .....	42
Ilustración 19 Gráficos FMeasure Predicciones.....	43
Ilustración 20 PCA 2D .....	44
Ilustración 21 KMeans 3 Clústeres .....	45
Ilustración 22 DBSCAN.....	45
Ilustración 23 GaussianMixture .....	45
Ilustración 24 Información del Clúster.....	46
Ilustración 25 Index.....	48
Ilustración 26 Ejemplo CSV.....	48
Ilustración 27 Formulario Cargar .....	49
Ilustración 28 Cargar mensaje error .....	50
Ilustración 29 Cargar Espera .....	50
Ilustración 30 Carga Tabla Resultado .....	50
Ilustración 31 Predicción Cargar .....	52
Ilustración 32 Predicción error SQL .....	53
Ilustración 33 Predicción error faltan datos .....	53
Ilustración 34 Predicción espera.....	53
Ilustración 35 Predicción información .....	54
Ilustración 36 Información estado 1 .....	55
Ilustración 37 Información estado 2 .....	55
Ilustración 38 Tabla de predicciones.....	56
Ilustración 39 Exportar SELECT .....	57
Ilustración 40 Exporta consulta invalidar.....	57

Ilustración 41 Exportar Diagrama .....	57
--	----

---

## Índice de Tablas

---

Tabla 1 Sprint 1 .....	7
Tabla 2 Sprint 2 .....	8
Tabla 3 Sprint 3 .....	8
Tabla 4 Sprint 4 .....	8
Tabla 5 Sprint 5 .....	8
Tabla 6 Sprint 6 .....	9
Tabla 7 Sprint 7 .....	9
Tabla 8 Sprint 8 .....	9
Tabla 9 Sprint 9 .....	9
Tabla 10 Sprint 10 .....	10
Tabla 11 Sprint 11 .....	10
Tabla 12 Sprint 12 .....	10
Tabla 13 Sprint 13 .....	10
Tabla 14 Sprint 14 .....	11
Tabla 15 Sprint 15 .....	11
Tabla 16 Sprints resultado de tiempos .....	11
Tabla 17 Costes de empleados .....	12
Tabla 18 Costes de material .....	12
Tabla 19 Costes mensuales de explotación .....	13
Tabla 20 Tabla de licencias .....	13
Tabla 21 CU1 – Agregar datos al sistema .....	17
Tabla 22 CU1.1 – Configurar estructura del CSV .....	17
Tabla 23 CU2 – Predecir Trayectorias .....	18
Tabla 24 CU2.1 – Entrenar clasificador .....	18
Tabla 25 CU2.2 - Reiniciar .....	19
Tabla 26 CU2.3 – Cargar Trayectorias .....	19
Tabla 27 CU3 – Exportar datos de la aplicación .....	20
Tabla 28 CU4 – Agregar datos de OSM .....	20
Tabla 29 Tokens Fechas .....	49

---

# Plan de Proyecto Software

---

## A.1. Introducción

En este apartado se expone la planificación temporal del proyecto y el estudio de viabilidad económico y legal. En el apartado de planificación temporal se explica que técnicas se han utilizado y como se han utilizado. En viabilidad económica veremos los costes de la puesta en marcha del proyecto y en viabilidad legal estudiaremos si las licencias de todos los componentes del proyecto son compatibles legalmente.

## A.2. Planificación temporal

La planificación temporal está basada en **Scrum** una metodología ágil para la gestión de proyectos. Este ha sido adaptado a las circunstancias del proyecto ya que la disponibilidad horaria no ha sido la misma a lo largo del desarrollo y a el hecho de que nos encontramos en un contexto educativo. En este contexto no hay un cliente y tampoco hay reuniones diarias ya que solo hay un desarrollador.

Algunas de las cosas que hay que tener en cuenta antes de pasar a ver los detalles de la planificación son las siguientes:

- El proyecto se divide en sprints o hitos, cada hito tiene una duración similar, aunque adecuándose a las circunstancias.
- Los sprints tienen tareas asociadas con un cálculo de duración en fracciones de tiempo indeterminadas.
- Se persigue que cada sprint tenga una carga de trabajo similar a la del resto de sprints.
- No se ha podido determinar el grado de cumplimiento de los tiempos debido a la falta de horarios.

El proyecto se inicia el 13/11/2018

### Sprint 1

Su finalidad es investigar acerca del proyecto aspectos tales como artículos sobre el tema de las trayectorias semánticas, aplicaciones relacionadas, herramientas para su desarrollo, etc.

Fecha de fin: 30/11/2018

*Tabla 1 Sprint 1*

Tarea	Estimación
Filtrar trayectorias por tiempo	5
Extraer datos de las rutas	3
Estudiar proyectos similares	3



<b>Estudiar artículos de trayectorias</b>	13
<b>Investigar herramientas</b>	8
<b>Añadir memoria al repositorio</b>	1

#### Sprint 2

En este sprint se buscaba prepara algunos de los elementos básicos para la aplicación. Fundamentalmente los sistemas de bases de datos y las bases de datos geoespaciales.

Fecha de fin: 17/12/2018

*Tabla 2 Sprint 2*

<b>Tarea</b>	<b>Estimación</b>
<b>Documentarse sobre la instalación de PostGIS</b>	3
<b>Instalar PostGIS</b>	3
<b>Estudiar como cagar datos en PostGIS</b>	5
<b>Buscar información de como cargar los datos de OSM</b>	5
<b>Cargar datos de OSM</b>	3

#### Sprint 3

Investigar los datos de OSM para ver cuáles son útiles para nuestro proyecto y tratar de construir trayectorias

Fecha de fin: 11/01/2019

*Tabla 3 Sprint 3*

<b>Tarea</b>	<b>Estimación</b>
<b>Seleccionar datos relevantes de OSM</b>	8
<b>Anotar Trayectorias</b>	5

#### Sprint 4

Procesar las rutas para ver que tenemos y si se pueden relacionar con los datos de OSM.

Fecha de fin: 01/02/2019

*Tabla 4 Sprint 4*

<b>Tarea</b>	<b>Estimación</b>
<b>Preprocesar las rutas</b>	13

#### Sprint 5

Implementar un algoritmo de detección de paradas.

Fecha de fin: 15/02/2019

*Tabla 5 Sprint 5*

<b>Tarea</b>	<b>Estimación</b>
<b>Obtener stops en las rutas</b>	13

## Sprint 6

Implementar un segundo algoritmo de detección de paradas.

Fecha de fin: 01/03/2019

*Tabla 6 Sprint 6*

<b>Tarea</b>	<b>Estimación</b>
<b>implementación de un segundo algoritmo de detección de paradas</b>	<b>5</b>

## Sprint 7

El objetivo de este sprint es crear un primer modelo para los datos de la aplicación que son las trayectorias conceptuales y crear una primera base de datos que las contenga.

Fecha de fin: 14/03/2019

*Tabla 7 Sprint 7*

<b>Tarea</b>	<b>Estimación</b>
<b>Refactorizar código</b>	<b>5</b>
<b>Crear Modelo Conceptual</b>	<b>5</b>
<b>Crear Base de Datos</b>	<b>8</b>

## Sprint 8

Modificar el diseño de la base de datos del anterior sprint, empezar con la documentación e implementar funciones para cargar datos en la base de datos.

Fecha de fin: 29/03/2019

*Tabla 8 Sprint 8*

<b>Tarea</b>	<b>Estimación</b>
<b>Hacer documentación</b>	<b>5</b>
<b>Adaptar la base de datos a los cambios</b>	<b>2</b>
<b>Implementar la carga de datos en la base de datos</b>	<b>8</b>

## Sprint 9

Cargar los datos en la base de datos y ampliar la documentación.

Fecha de fin: 10/04/2019

*Tabla 9 Sprint 9*

<b>Tarea</b>	<b>Estimación</b>
<b>Memoria rellenar conceptos teóricos</b>	<b>5</b>
<b>Cargar datos de la BD</b>	<b>3</b>

## Sprint 10

Preparar un servidor con la aplicación de Nominatim para la búsqueda de puntos de interés y estudiar un algoritmo de predicción.

Fecha de fin: 26/04/2019

*Tabla 10 Sprint 10*

Tarea	Estimación
<b>Manual de Instalación de Nominatim</b>	5
<b>Instalar Nominatim</b>	8
<b>Lectura de datos de Nominatim</b>	5
<b>Estudiar el algoritmo de predicción</b>	3

## Sprint 11

Probar algoritmos de *clustering* y el clasificador estudiado en el sprint anterior.

Fecha de fin: 17/05/2019

*Tabla 11 Sprint 11*

Tarea	Estimación
<b>Probar Clustering</b>	8
<b>Probar Clasificador de árbol</b>	8

## Sprint 12

Resolver problemas que se podían apreciar en el clasificador dado que había demasiadas clases para predecir y muy pocos datos.

Fecha de fin: 30/05/2019

*Tabla 12 Sprint 12*

Tarea	Estimación
<b>Crear arboles tipo</b>	5
<b>Agrupar clases por grupos</b>	13
<b>Resolver el problema de los edificios sin tipo</b>	13

## Sprint 13

Aprender cómo realizar aplicaciones web, más concretamente con el *framework* de Flask y crear un diseño base de la aplicación.

Fecha de fin: 03/06/2019

*Tabla 13 Sprint 13*

Tarea	Estimación
<b>Diseño Base de la aplicación web</b>	5
<b>Aprender Flask</b>	5

## Sprint 14

Avanzar a la memoria y los anexos y tratar de emplear un algoritmo de *clustering* utilizando matrices de adyacencia.

Fecha de fin: 11/06/2019

*Tabla 14 Sprint 14*

<b>Tarea</b>	<b>Estimación</b>
<b>Anexo B: B-1, B-2, B-3</b>	<b>8</b>
<b>Anexo C: Diagramas de casos de uso</b>	5
<b>Crear un algoritmo de clustering</b>	<b>8</b>
<b>Memoria: Conceptos teóricos</b>	5
<b>Memoria: Objetivos</b>	<b>3</b>
<b>Memoria: Introducción</b>	3

## Sprint 15

Finalizar la memoria la página web y los anexos.

Fecha de fin: 02/07/2019

*Tabla 15 Sprint 15*

<b>Tarea</b>	<b>Estimación</b>
<b>Terminar la memoria</b>	<b>8</b>
<b>Terminar página web</b>	13

En la siguiente tabla tenemos las sumas de los tiempos de cada sprint y el total, si sacamos la media de los tiempos nos da que cada sprint debería durar 18,2 fracciones de tiempo.

*Tabla 16 Sprints resultado de tiempos*

<b>Sprint 1</b>	38
<b>Sprint 2</b>	19
<b>Sprint 3</b>	13
<b>Sprint 4</b>	13
<b>Sprint 5</b>	13
<b>Sprint 6</b>	5
<b>Sprint 7</b>	18
<b>Sprint 8</b>	15
<b>Sprint 9</b>	8
<b>Sprint 10</b>	21
<b>Sprint 11</b>	16
<b>Sprint 12</b>	31
<b>Sprint 13</b>	10
<b>Sprint 14</b>	32
<b>Sprint 15</b>	21

<b>Total</b>	<b>273</b>
--------------	------------

### A.3. Estudio de viabilidad

#### Viabilidad económica

En este apartado de viabilidad económica se va a calcular los costes que tendría el desarrollo de esta aplicación para una empresa. Y los costes de explotación anuales de la aplicación plenamente operativa y con capacidad para trabajar con datos de todo el planeta. Los requisitos del sistema están basados en recomendaciones de Nominatim para manejar una base de datos de OSM completa.

#### Costes de desarrollo

Primero se calculan los costes que supone contratar a un desarrollador, para calcular esto se estima que un programador cobra 1500€ brutos y se estima el coste de la seguridad social en un 33%. También se tiene en cuenta que la duración del proyecto son 30 horas por crédito, siendo 12 los créditos del TFG. Esto hacen 360 horas que son aproximadamente dos meses a jornada completa.

Tabla 17 Costes de empleados

<b>Sueldo bruto al mes</b>	<b>1.500 €/mes</b>
<b>Seguridad social</b>	<b>495€/mes</b>
<b>Coste total de 2 meses</b>	<b>3.990 €</b>

A esto hay que añadir los costes derivados del equipo necesario para el desarrollo del software. Se calcula un plazo de amortización de 5 años para los equipos informáticos. Partimos de que un portátil promedio para este fin tiene un coste de 800€ con sistema operativo incluido.

Tabla 18 Costes de material

<b>Office 365 Empresa Premium</b>	<b>21,00€</b>
<b>Visual Studio Code</b>	<b>0,00€</b>
<b>Ordenador</b>	<b>26,66€</b>
<b>Total</b>	<b>47,66€</b>

Esto hace un coste total de desarrollo de **4.037,66€**

#### Costes de explotación

Para esto vamos a calcular el coste mensual que tiene un servidor Con 32GB de memoria RAM, 8 vCPU y una capacidad de 2TB de disco duro sólido. Los costes

son mensuales, están estimados para la plataforma Azure de Microsoft y pueden variar sensiblemente.

*Tabla 19 Costes mensuales de explotación*

<b>Servidor virtual</b>	<b>240,93€</b>
<b>2 TB disco duro SSD</b>	<b>198,62€</b>
<b>Ubuntu server 18.04</b>	<b>0€</b>
<b>Total</b>	<b>439,55€</b>

El coste anual de explotación se sitúa en **5.274,60€**

#### Viabilidad legal

En este apartado se estudia la viabilidad legal de la aplicación. Para esto se tiene en cuenta en primer lugar las licencias del software integrado en nuestra aplicación.

*Tabla 20 Tabla de licencias*

<b>Pandas</b>	<b>BSD</b>
<b>GeoPandas</b>	<b>BSD 3</b>
<b>SQLAlchemy</b>	<b>MIT</b>
<b>Flask</b>	<b>BSD</b>
<b>Prefixspan</b>	<b>MIT</b>
<b>BootStrap</b>	<b>MIT</b>
<b>Nominatim</b>	<b>GPL-2</b>

Todas ellas pueden usarse en una aplicación con licencia AGPL3 por lo que no supone ningún problema. Esta licencia es similar a GPL3, pero está pensada para aplicaciones Web.

El otro punto problemático puede ser el tratamiento de datos, pero al no contener datos personales ni que permitan identificar a individuos no hay ningún problema legal. Siempre y cuando los datos sean obtenidos de forma legal e informado a los usuarios de su uso.

---

# Especificación de Requisitos

---

## B.1. Introducción

En este apéndice se exponen los requisitos que debe tener el proyecto. Primero se especifican los objetivos generales de la aplicación software y posteriormente se pasa a catalogar todos los requisitos de la aplicación. Por último, se realiza una especificación detallada de los requisitos software de la aplicación.

## B.2. Objetivos generales

Los requisitos de funcionamiento que va a tener este proyecto son los siguientes.

- El objetivo principal que se persigue en este proyecto es crear una aplicación que permita analizar trayectorias. Con este fin se van a desarrollar e implementar las siguientes características.
  - Guardar los datos de la aplicación en una base de datos, estos datos se guardarán ya procesados.
    - Se precisa eliminar errores que pueda haber en las trayectorias debido a fallos aleatorios o de GPS.
    - Detectar cuales, de los puntos de la ruta, un usuario está parado y agrupar los puntos consecutivos en los que se determina que el usuario está parado.
    - Asignar identificadores de OSM a las paradas, estos identificadores corresponden a la dirección en la que determina que está parado.
  - Permitir al usuario decidir exactamente qué datos de la base de datos de la aplicación quiere cargar en el sistema para usar las distintas funcionalidades de la aplicación.
  - Predictor que permita predecir a qué categoría o tipo pertenece la siguiente parada de una trayectoria.
  - Una interfaz amigable para que el usuario pueda utilizar la aplicación.
  - Permitir al usuario que pueda exportar datos de la base de datos a un fichero CSV.

## B.3. Catálogo de requisitos

A continuación, se enumeran los requisitos funcionales y no funcionales de la aplicación.

### Requisitos funcionales

- **RF1 - Cargar datos CSV en la aplicación:** El usuario debe poder agregar nuevos datos procedentes de ficheros CSV.
  - **RF1.1 – Espera en carga de datos:** El usuario debe ser consciente de que los datos se están cargando en la aplicación y que debe esperar.

- **RF1.2 – Resultado de la carga de datos:** El usuario debe recibir información referente a los datos que se han cargado en la aplicación tras el proceso de carga.
  - **RF1.3 – Exportar resultados de carga:** El usuario podrá exportar los resultados que recibe una vez terminado el proceso de carga de datos.
- **RF2 – Predecir clases de las rutas:** El usuario deberá poder entrenar un clasificador con un conjunto de datos seleccionado, para predecir la clase a la que pertenecerá el siguiente punto de las rutas.
  - **RF2.1 – Selección de datos:** El usuario debe poder elegir que datos desea utilizaren cada momento.
  - **RDF2.2 – Validación cruzada:** El usuario visualizara los resultados de realizar una validación cruzada con el conjunto de datos cargado antes de entrenar el clasificador.
  - **RF2.3 – Datos insuficientes:** El usuario será informado en caso de haber datos insuficientes para entrenar un clasificador, en cuyo caso no podrá entrenar el clasificador con esos datos.
  - **RF2.4 – Reiniciar clasificador:** El usuario podrá reiniciar el clasificador así como todos los datos que están cargado en la aplicación para ser utilizados.
  - **RF2.5 – Múltiples predicciones:** El usuario podrá utilizar el clasificador múltiples veces para predecir Trayectorias semánticas
  - **RF2.6 – Exportar resultado de predicción:** El usuario podrá exportar los resultados de la predicción obtenidos.
- **RF3 – Exportar datos de la aplicación:** El usuario deberá poder exportar los datos contenidos en la aplicación mediante una sentencia de tipo SELECT.
  - **RF3.1 – Respuesta al usuario:** El usuario debe recibir un aviso si la sentencia SELECT no está bien formulada o un fichero CSV si la operación fue procesada correctamente.
- **RF4 – Datos de OSM:** El administrador podrá añadir nuevos datos de OSM a la aplicación de Nominatim.

#### Requisitos no funcionales

**RNF1 – Facilidad de uso:** El usuario debe recibir en todo momento información sobre los fallos en el uso de la aplicación y el estado de la aplicación

**RNF2 – Información adicional:** El usuario recibirá información adicional en el contexto de la aplicación o facilidades para no necesitar conocer la estructura de la base de datos para poder usarla.

#### Limitaciones Funcionales

**LF1 – Usuarios:** La aplicación no deberá ser utilizada por más de un usuario al mismo tiempo ya que compartirían la misma sesión.



#### B.4. Especificación de requisitos

En este apartado se van a especificar los casos de uso, que requisitos funcionales va a abarcar y que característica tiene cada caso de uso.

##### Actores

En los casos de uso que se exponen a continuación encontramos dos actores con roles completamente distintos.

**Usuario:** Este es el actor principal de la aplicación y es además el único que interactúa directamente con ella mediante la página web.

**Administrador:** Este es el encargado de que Nominatim tenga los datos de las partes del mundo que los usuarios puedan requerir y de que estén actualizadas.

##### Diagramas de casos de uso

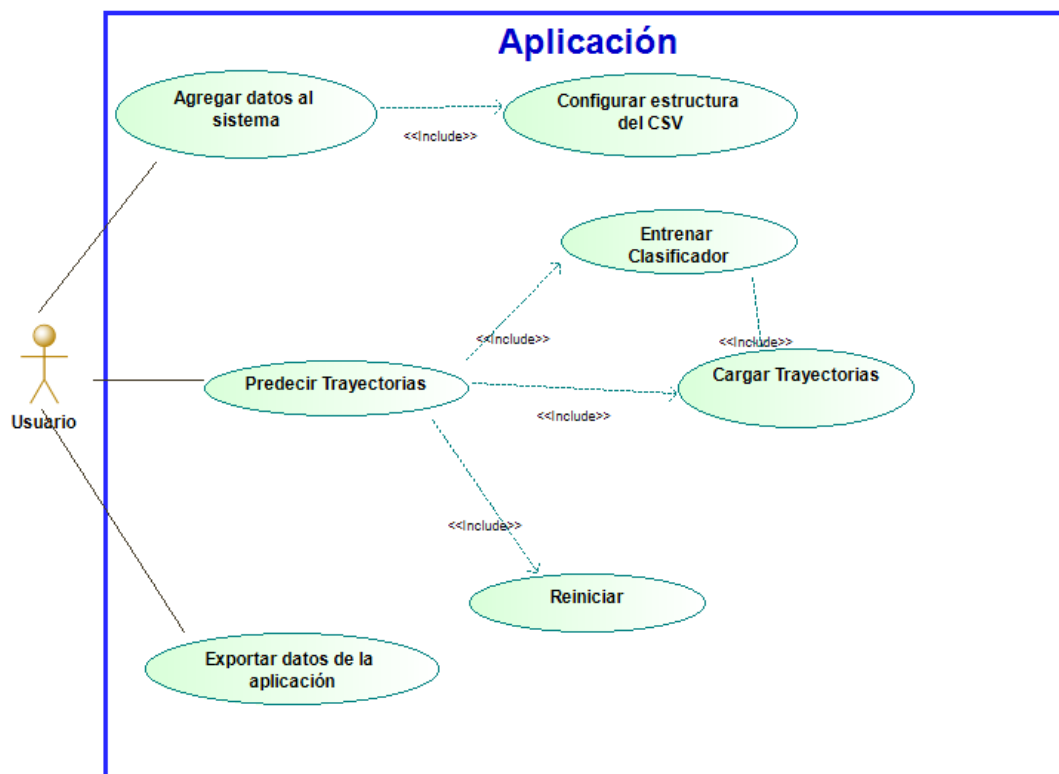


Ilustración 1 Casos de uso Usuario

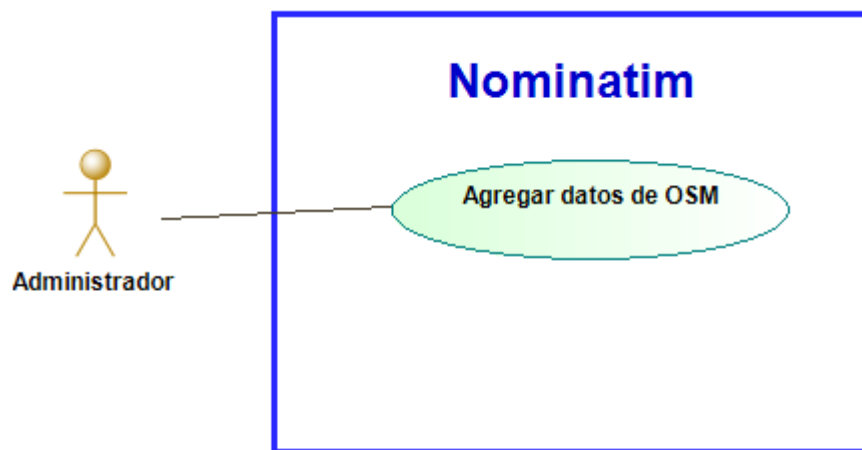


Ilustración 2 Casos de uso Administrador

## Especificación de los casos de uso

Tabla 21 CU1 – Agregar datos al sistema

<b>Nombre</b>	<b>CU1 – Agregar datos al sistema</b>	
<b>Descripción</b>	El usuario agrega datos al sistema	
<b>Requisitos funcionales</b>	RF1, RF1.1, RF1.2, RF1.3	
<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	En la página web estar en la ventana de carga, haber rellenado correctamente la configuración del CSV	
<b>Secuencia Normal</b>	Paso	Acción
	1	El usuario solicita cargar los datos
	2	El navegador carga los datos en el servidor
	3	El navegador pide la página cada 5 segundos al servidor, si termina el proceso de carga, varía la página y salta al paso 7
	4	El servidor lee los archivos con extensión especificada
	5	El servidor procesa los datos
	6	El servidor guarda los resultados en la base de datos
<b>Excepciones</b>	Paso	Acción
	1	Se produce un error al intentar cargar los datos
	2	El servidor envía una ventana de error al navegador
<b>Postcondiciones</b>	Se añaden datos a la base de datos	
<b>Frecuencia</b>	Baja	
<b>Importancia</b>	Crítica	

Tabla 22 CU1.1 – Configurar estructura del CSV

<b>Nombre</b>	<b>CU1.1 – Configurar estructura del CSV</b>
---------------	--

<b>Descripción</b>	El usuario configura el lector de CSV	
<b>Requisitos funcionales</b>	RF1	
<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	Estar en la ventana de carga de datos	
<b>Secuencia Normal</b>	Paso	Acción
	1	El usuario rellena los campos
	2	El usuario pulsara el botón de cargar
<b>Excepciones</b>	Paso	Acción
	1	Al pulsar agregar hay campos obligatorios sin rellenar o campos inválidos
	2	El navegador informa al usuario del primer campo vacío o invalido
<b>Postcondiciones</b>	Permite cargar los datos en el servidor	
<b>Frecuencia</b>	Baja	
<b>Importancia</b>	Critica	

Tabla 23 CU2 – Predecir Trayectorias

<b>Nombre</b>	<b>CU2 – Predecir Trayectorias</b>	
<b>Descripción</b>	Permite al usuario predecir la clase a la que pertenecerá la próxima ubicación en una trayectoria	
<b>Requisitos funcionales</b>	RF2, RF2.5, RF2.6	
<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	Tener el clasificador entrenado, tener datos cargados	
<b>Secuencia Normal</b>	Paso	Acción
	1	El usuario pulsa el botón de predecir
	2	El servidor pasa los datos actualmente cargados por el clasificador
	3	El navegador muestra en una tabla los resultados y permite exportarla
	4	El usuario puede exportar la tabla y/o volver a predecir o cargar datos
<b>Excepciones</b>	Paso	Acción
<b>Postcondiciones</b>		
<b>Frecuencia</b>	Alta	
<b>Importancia</b>	Critica	

Tabla 24 CU2.1 – Entrenar clasificador

<b>Nombre</b>	<b>CU2.1 – Entrenar clasificador</b>
<b>Descripción</b>	Entrena de forma automática un clasificador, realiza la validación cruzada
<b>Requisitos funcionales</b>	RF2, RF2.5

<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	Terminar de cargar los datos en el sistema, no hay clasificador entrenado	
<b>Secuencia Normal</b>	Paso	Acción
	1	El servidor realiza una validación cruzada
	2	Entrena un clasificador
	3	Presenta los resultados de la validación en el navegador
<b>Excepciones</b>	Paso	Acción
<b>Postcondiciones</b>	Obtenemos un clasificador entrenado	
<b>Frecuencia</b>	Alta	
<b>Importancia</b>	Crítica	

Tabla 25 CU2.2 - Reiniciar

<b>Nombre</b>	<b>CU2.2 - Reiniciar</b>	
<b>Descripción</b>	Devuelve a todos los componentes de predicción a su estado inicial	
<b>Requisitos funcionales</b>	RF2.4	
<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	Estar en la ventana de predicciones	
<b>Secuencia Normal</b>	Paso	Acción
	1	El usuario pulsa el botón de reiniciar
	2	El servidor devuelve las variables de predicción a su estado original
	3	Se refresca la página web
<b>Excepciones</b>	Paso	Acción
<b>Postcondiciones</b>	Desaparecen todo lo realizado en la ventana predicciones	
<b>Frecuencia</b>	Media	
<b>Importancia</b>	Alta	

Tabla 26 CU2.3 – Cargar Trayectorias

<b>Nombre</b>	<b>CU2.3 – Cargar Trayectorias</b>	
<b>Descripción</b>	Carga trayectorias en memoria	
<b>Requisitos funcionales</b>	RF2.1, RF2.3	
<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	Estar en la ventana de predicción	
<b>Secuencia Normal</b>	Paso	Acción
	1	El usuario puede rellenar, o no, los campos de la columna cargar datos
	2	El usuario presiona el botón de cargar
	3	Se envía una solicitud al servidor

	4	El servidor almacena los datos en memoria
<b>Excepciones</b>	Paso	Acción
	1	Si hay campos incompatibles con la consulta SELECT devuelve un error
<b>Postcondiciones</b>	Datos cargados en el sistema	
<b>Frecuencia</b>	Alta	
<b>Importancia</b>	Critica	

Tabla 27 CU3 – Exportar datos de la aplicación

<b>Nombre</b>	<b>CU3 – Exportar datos de la aplicación</b>	
<b>Descripción</b>	Permite a un usuario exportar datos almacenados en la base de datos	
<b>Requisitos funcionales</b>	RF3, RF3.1	
<b>Usuario</b>	Usuario	
<b>Precondiciones</b>	Estar en la ventana exportar, que el usuario tenga conocimientos de SQL básicos	
<b>Secuencia Normal</b>	Paso	Acción
	1	El usuario rellena la consulta SELECT
	2	El usuario presiona el botón para hacer la exportación
	3	El servidor detecta si la consulta pide datos geoespaciales, en caso afirmativo ir al paso 5
	4	El servidor hace una petición normal a la base de datos saltar al paso 7
	5	El servidor realiza una consulta con datos geoespaciales
	6	El servidor convierte la columna de datos geoespacial en una de longitud y otra de latitud
	7	Se devuelve un archivo CSV con los datos
<b>Excepciones</b>	Paso	Acción
	1	Si la consulta es invalida el servidor devuelve una notificación
<b>Postcondiciones</b>		
<b>Frecuencia</b>	Baja	
<b>Importancia</b>	Media	

Tabla 28 CU4 – Agregar datos de OSM

<b>Nombre</b>	<b>CU4 – Agregar datos de OSM</b>	
<b>Descripción</b>	Añadir datos a Nominatim	
<b>Requisitos funcionales</b>	RF4	
<b>Usuario</b>	Administrador	
<b>Precondiciones</b>	Estar logrado con el usuario de Nominatim en el servidor, tener internet	
<b>Secuencia Normal</b>	Paso	Acción

	1	El administrador debe descargar los nuevos datos de internet
	2	El administrador debe pasar estos datos al <i>setup</i> de Nominatim
<b>Excepciones</b>	Paso	Acción
<b>Postcondiciones</b>	Nuevas zonas para obtener puntos de interés	
<b>Frecuencia</b>	Baja	
<b>Importancia</b>	Crítica	

## Especificación de diseño

### C.1. Introducción

En este apartado se exponen los aspectos de diseño más importantes de la aplicación. En el apartado de diseño de datos se define como están estructurados los datos y como se almacena. En el diseño arquitectónico se explica cómo está estructurada la aplicación. Por último, en el diseño procedimental explican aquellas características más importantes de los módulos que conforman la aplicación.

### C.2. Diseño de datos

En este proyecto se utilizan bases de datos concretamente bases de datos geoespaciales. Primero vamos a analizar el diseño de la base de datos que aparece en la siguiente ilustración.

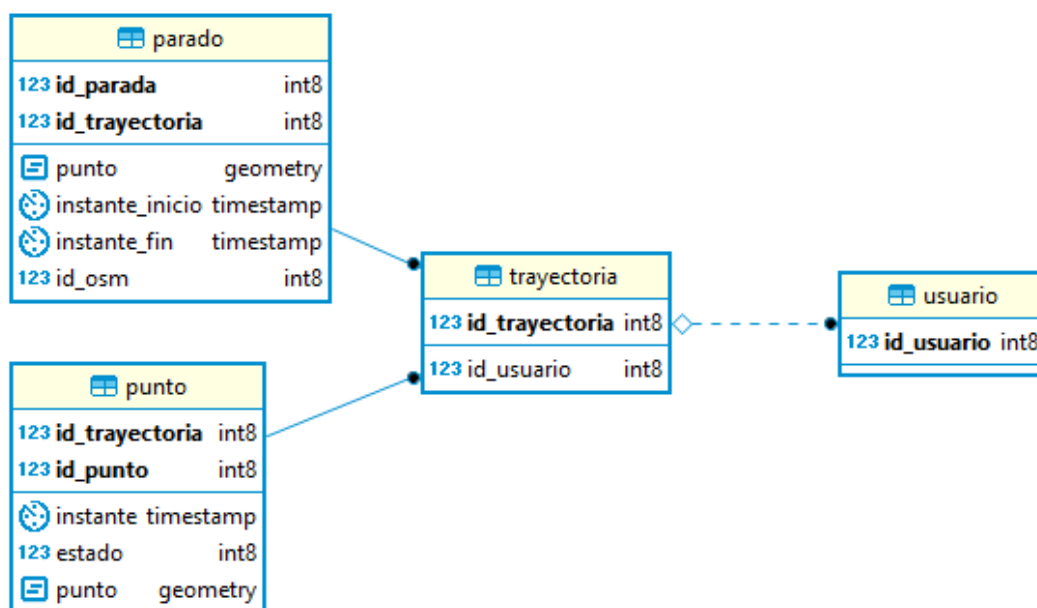


Ilustración 3 Base de Datos

Vamos a analizar el diagrama de derecha a izquierda y de arriba abajo. Lo primero que encontramos es la tabla usuario la cual solo tiene el id de usuario. La razón por la que se creó esta tabla en vez de integrarse en las trayectorias se debe a que se barajaba la posibilidad de guardar información sobre el usuario después de un proceso de análisis de sus trayectorias. Tiene una relación de "1 a N" con las trayectorias dado que un usuario tiene muchas trayectorias.

La tabla de trayectoria fundamentalmente relaciona a los usuarios las trayectorias tanto las formadas por paradas (que también denominamos conceptuales)

y los formadas por puntos (que denominamos trayectorias conceptuales). Con ambas tablas tiene una relación de “1 a N” dado que una trayectoria bruta está formada por “N” puntos y una trayectoria conceptual está formada por “N” paradas.

La tabla, parado está formado por una clave doble por un lado la de la trayectoria que se repite en todas las paradas de la misma trayectoria y por “id\_parada” que enumera a las paradas de una misma trayectoria. Esto nos permite identificarlas de forma única y además podemos mantener su correcto orden. En la tabla punto funciona de la misma manera. En la tabla parado se almacenan dos valores de tiempo que corresponden a el tiempo del inicio de la parada y a el tiempo del fin de la parada. El campo “id\_osm” se utiliza para buscar los datos del punto de interés relacionado con esa parada. Ya que ese campo es el del identificador en OSM del punto de interés relacionado con la parada.

Por último, ambas tablas tienen un campo punto de tipo geometría el cual contiene en cierto modo las coordenadas en las que se realizó la parada o en punto por el que paso el usuario. Realmente no son las coordenadas si no un valor que utiliza PostGIS para identificar al punto en sus tablas propias.

En la aplicación cada trayectoria se almacena en un objeto la cual contiene *DataFrames* con estos datos, más concretamente *GeoDataFrames* así como la *id* de usuario y la *id* de la trayectoria.

Las trayectorias semánticas en la aplicación son más un concepto que una realidad, ya que para no tener información redundante se obtiene a través de las trayectorias conceptuales. Se agrupan las paradas por id de OSM y se obtiene la información que buscamos.

### C.3. Diseño arquitectónico

Se describe como está distribuida la aplicación y como se relaciona sus componentes. En la ilustración siguiente se observa un plano general de cómo se agrupan los componentes en el patrón de diseño MVC. Posteriormente se verá más de cerca cada uno de sus componentes y que se encapsula en estos.



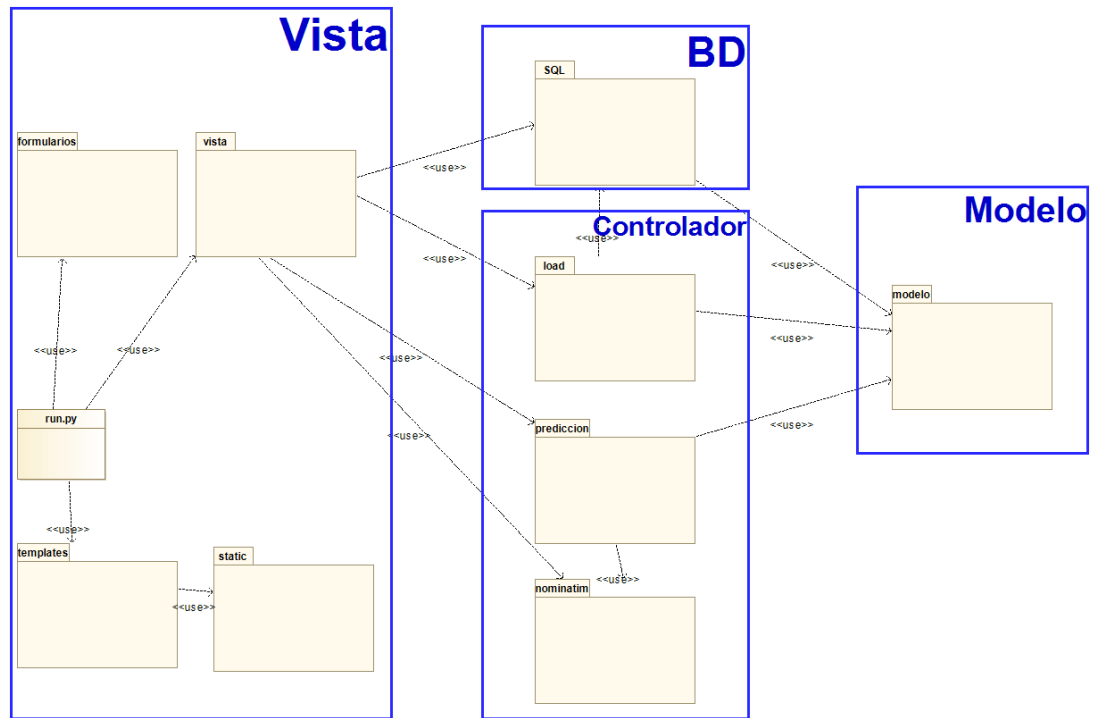


Ilustración 4 Paquetes MVC

### Modelo

El modelo en este diseño solo se accede de forma directa por el controlador y el módulo de bases de datos. Al tratarse de una aplicación que se centra en un modelo de datos concreto estos están agrupados en un solo modulo.

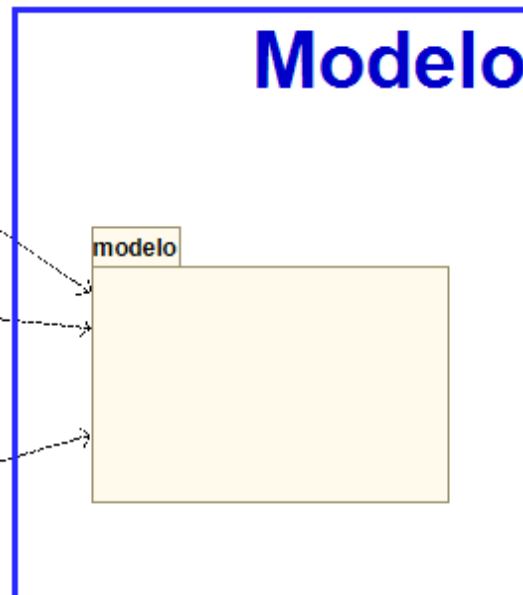


Ilustración 5 Paquetes Modelo

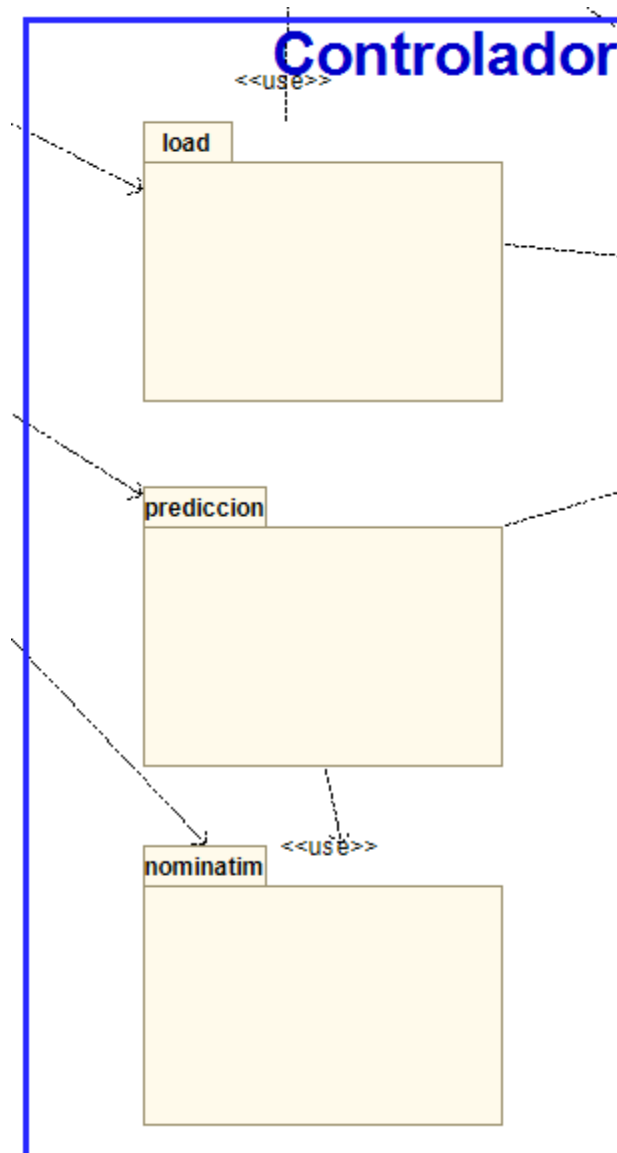
### Controlador

Este grupo está formado por tres componentes los cuales cada uno encapsula una parte de la lógica del programa.

**Load:** Se encarga de cargar los datos de los archivos y convertirlos en trayectorias. A este solo se accede desde la vista cuando un usuario quiere introducir nuevas trayectorias en el sistema. A su vez este accede a las bases de datos para saber que claves tiene que asignar a usuarios y trayectorias. También accede al modelo de datos dado que lo que hace es construir trayectorias brutas y conceptuales en base a este.

**Predicción:** Se encarga de toda la lógica del programa relacionada con minería de datos propiamente dicha. A esta se accede desde el módulo de vista, y esta a su vez accede a el modelo y al módulo de Nominatim.

**Nominatim:** Este módulo bien se podría haberse situado dentro de bases de datos, aunque no lo es en el sentido estricto, pero tiene una finalidad similar. Este módulo no necesita de ningún otro ya que solo se encarga de la comunicación entre la aplicación y el servidor de Nominatim. Este módulo recibe peticiones tanto de la vista como del módulo de predicciones.

*Ilustración 6 Paquetes Controlador*

### Vista

Este grupo fundamentalmente representa en este caso a la parte de web más concretamente a Flask. Flask nos obliga a tener para su propio funcionamiento “templates” y “static” los cuales no contiene código Python. “templates” contiene las paginas HTML y “static” contiene los archivos de estilos CSS y los archivos de Java Script.

Formularios: contiene los formularios que se utilizaran en la página web por parte de Flask. Están en código Python, pero solo se relaciona con “run.py” para que Jinja2 los pueda renderizar.

Vista: Contiene clases utilizadas por la vista para almacenar información que necesita y la clase que agrupa las funciones que se necesitan del modelo y de la base de datos. Es el único módulo de la vista que se comunica con el resto de la aplicación.

Run.py: Esta clase es la clase principal de Flask ya que es la que arranca el servidor de la página web y es donde se definen las solicitudes HTTP que aceptara el servidor y que respuesta se da a esas solicitudes.

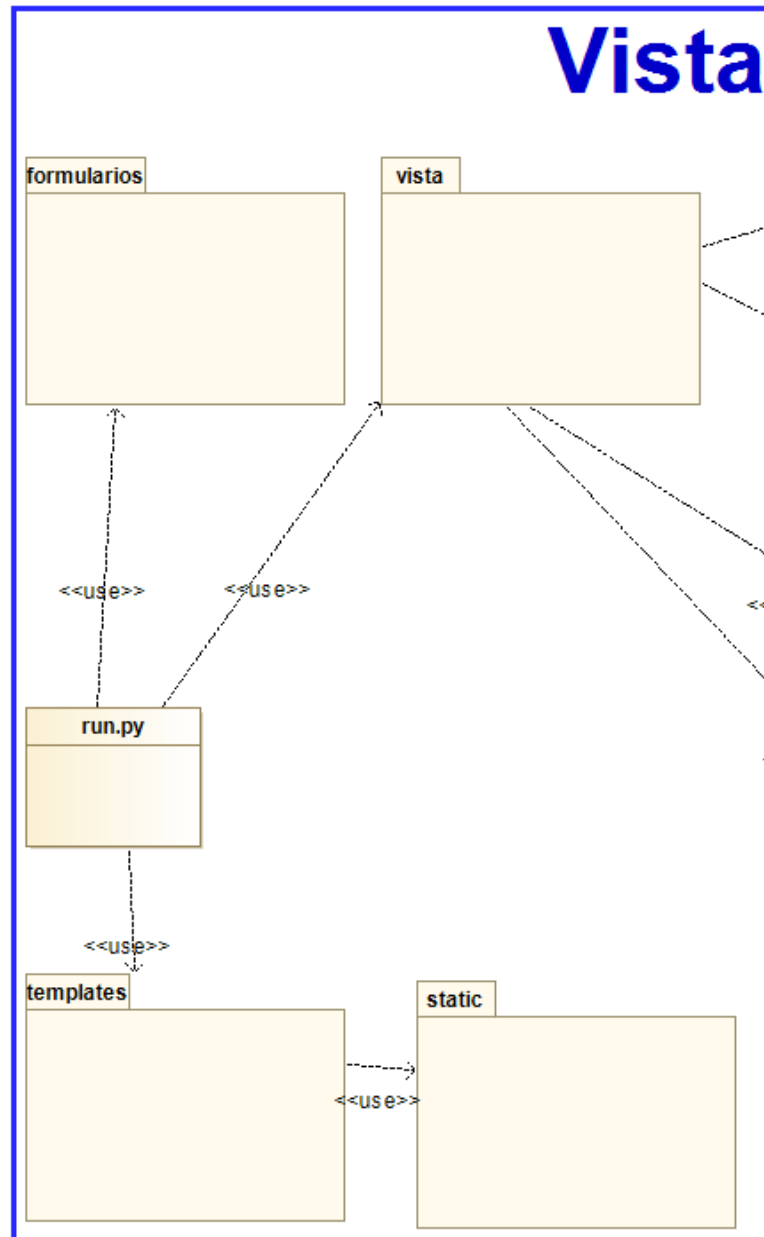


Ilustración 7 Paquetes Vista

### Base de Datos

El módulo de bases de datos se encarga de todo en tráfico de datos entre la base de datos y la aplicación. Está encargado de establecer conexiones, de convertir el modelo a registros de la base de datos y de convertir los registros de la base de datos en el modelo.

Este módulo es utilizado tanto por parte de la vista como por parte del módulo de load. Este a su vez depende del modelo para devolver los datos estructurados en base a este.

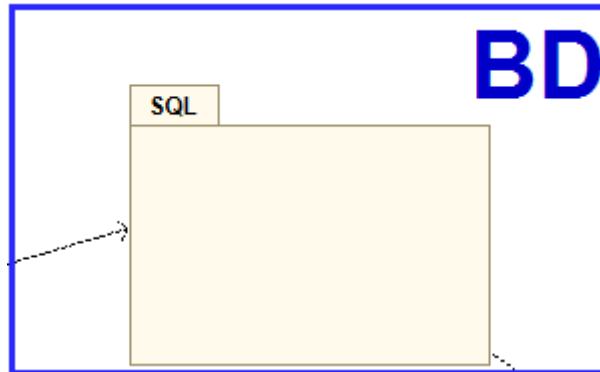


Ilustración 8 Paquetes Base de Datos

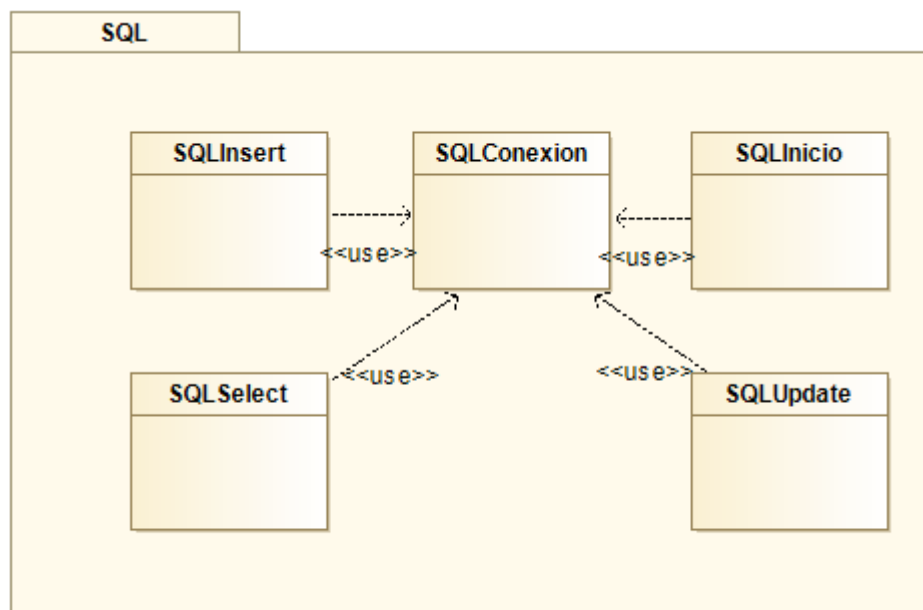
#### C.4. Diseño procedimental

En este apartado se tratan un poco más en profundidad como se relaciona algunos de los componentes de la aplicación. Como algunos de ellos son muy básicos y fáciles de entender no se van a detallar y se pondrá el foco en aquellos que puedan ser más difíciles de comprender.

Se debe destacar que esta no es una aplicación orientada a objetos, aunque si se utilizan. Esto implica que no todos los archivos de la aplicación contienen clases, si no que por el contrario algunos actúan como ficheros de configuración o como agrupaciones de funciones. Un buen ejemplo de esto es el paquete SQL el cual no tiene ninguna clase y los ficheros están compuestos por agrupaciones de funciones.

##### SQL

En la siguiente ilustración (que muestra un diagrama de clases, aunque no son clases) muestra cómo se relacionan y el funcionamiento es el mismo para todos los ficheros. El fichero “SQLConexion” guarda los datos de configuración del servidor y además tiene una función que proporciona una conexión con ese a quien la invoca. El único fichero que tiene un comportamiento distinto es “SQLInicio” que proporciona un contador actualizado al módulo “load” para asignar las ID a usuarios y trayectorias.

*Ilustración 9 Paquete SQL*

### Predicción

**Conversor:** Esta clase tiene distintas herramientas de conversión para convertir los datos de trayectorias a otros formatos que pueden entender clasificadores y algoritmos de *clustering*.

**Probador:** Es una clase construida con el objetivo de probar el clasificador incluye validación cruzada y cálculo de distintas medidas para evaluar la calidad del clasificador.

**ClasificadorPrediccion:** Es al mismo tiempo clase y paquete ya que alberga en el mismo fichero una clase nodo la cual utiliza para construir sus árboles.

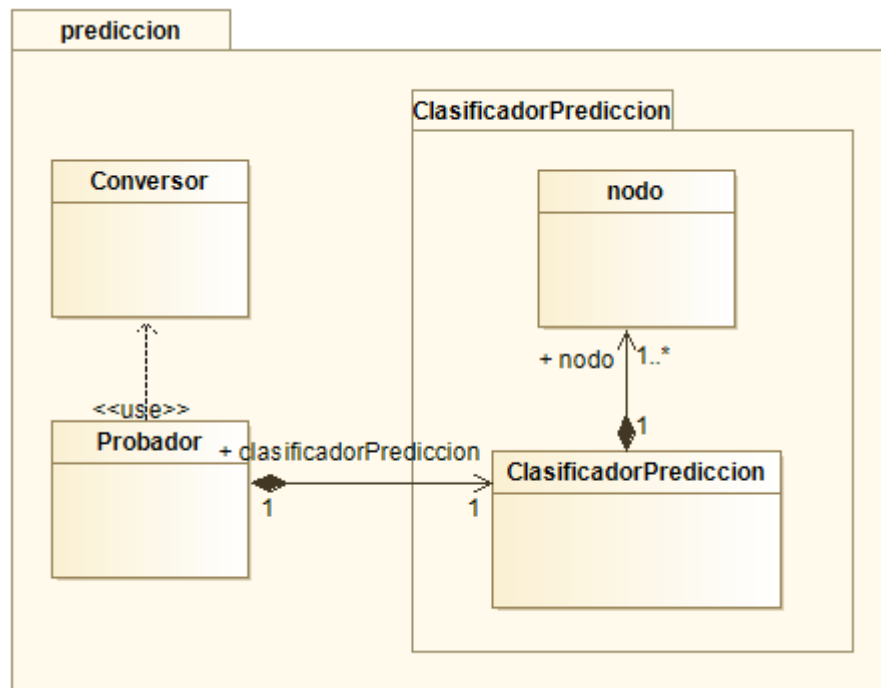
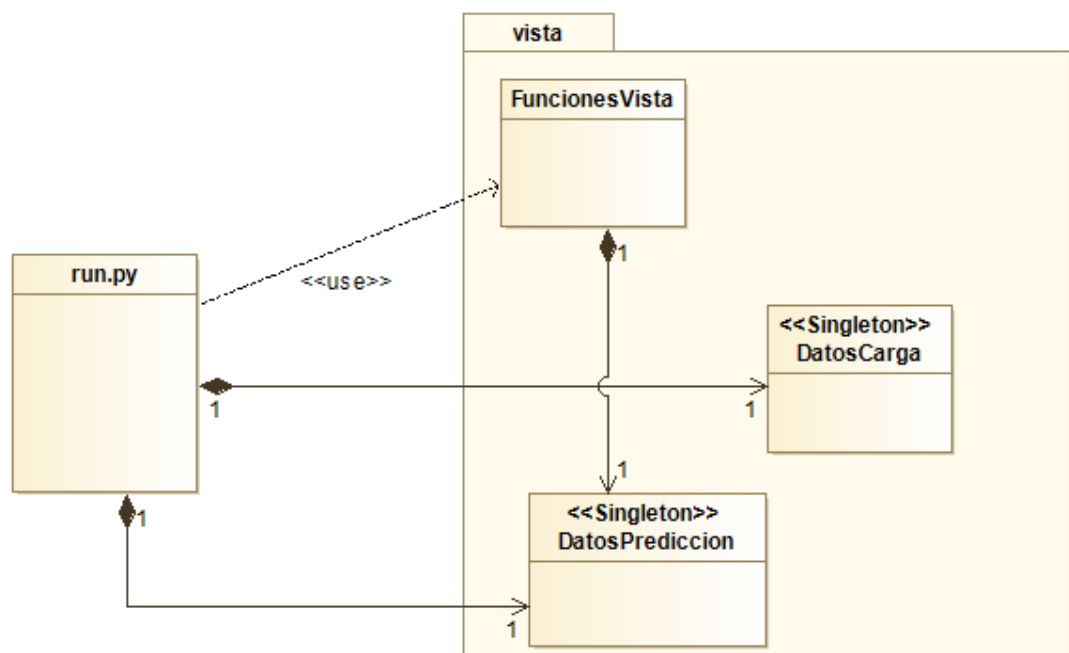


Ilustración 10 Paquete Predicción

### Vista

En la vista hay que destacar las comunicaciones entre “run” y “FuncionesVista” estas funciones tienen una serie de operaciones que con grandes volúmenes de datos pueden suponer un tiempo de ejecución excesivamente largo. Como el navegador no puede esperar mucho tiempo a que “run” responda lo que se hace es ejecutar procesos o hilos separados. Estos se almacenan en las dos clases *singleton* que aparecen en la ilustración. Lo que se hace “run” es lanzar la función de cargar como un proceso y lo almacena en la clase “DatosCargar” a si puede saber en todo momento si este proceso ha terminado, ya que pregunta en otras fases de ejecución. Esto es necesario debido a que run solo ejecuta alguna de sus funciones cuando el navegador hace alguna petición al servidor. Como “run” debe responder en un corto periodo de tiempo perdiendo el control no puede preguntar una y otra vez sin perder las variables de la función. Tampoco puede guardar variables globales porque no es una clase, por lo que una clase *singleton* puede solucionar este problema.

En el caso de “DatosPrediccion” se utiliza un hilo en vez de un proceso realizando una tarea similar con la diferencia que ahora este hilo también guarda datos en la clase “DatosPrediccion”.

*Ilustración 11 Paquete Vista y Run*



---

# Documentación técnica de programación

---

## D.1.Introducción

En este apartado se van a exponer distintas características técnicas del proyecto. Primero se expondrá la estructura de directorios del proyecto. Posteriormente se expondrán los pormenores relacionados con las necesidades para continuar desarrollando el software. A continuación, se explica cómo se ha de instalar el software y como se debe ejecutar. Por último, se expondrán las pruebas a las que ha sido sometida la aplicación.

## D.2.Estructura de directorios

EL proyecto tiene la siguiente estructura general partiendo de la raíz que es “GII17.0T\_Trayectorias\_Semanticas\_1819”.

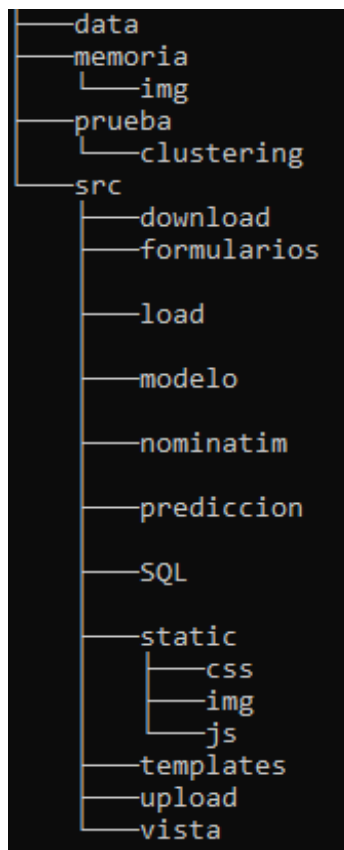


Ilustración 12 Estructura de directorios

1. Data: en esta carpeta encontramos datos como los utilizados en la aplicación. Concretamente se encuentran los datos que grave para pruebas.
2. Memoria: En esta se encuentran la memoria y los anexos del proyecto.

- 2.1. Img: contiene imágenes relacionadas con la memoria y los anexos concretamente se encuentran ahí las imágenes que se ponen a disposición del alumno.
- 3. Prueba: carpeta para almacenar pruebas que se realizan sobre aspectos del proyecto.
  - 3.1. Clustering: contiene pruebas echas sobre un conjunto de datos obtenido de la aplicación.
- 4. Src: en esta carpeta se almacena todo el código de la aplicación, así como carpetas necesarias para su funcionamiento.
  - 4.1. Download: carpeta requerida por la página web para almacenar los CSV antes de que el usuario los descargue.
  - 4.2. Formularios: carpeta que contiene los formularios en Python que posteriormente aparecen en la página web.
  - 4.3. Load: carpeta que contiene todos los ficheros Python relacionados con la carga de datos nuevos.
  - 4.4. Modelo: carpeta que contiene los ficheros Python que conforman el modelo de datos de la aplicación.
  - 4.5. Nominatim: carpeta que contiene los ficheros para interactuar con la aplicación de Nominatim.
  - 4.6. Predicción: contiene todos los archivos relacionados con minería de datos.
  - 4.7. SQL: Contiene los ficheros Python que permiten las operaciones con la base de datos de PostgreSQL.
  - 4.8. Static: contiene los directorios con los ficheros de la página web.
    - 4.8.1. Css: ficheros de estilos CSS para la página web.
    - 4.8.2. Img: imágenes que se vayan a utilizar en la página web.
    - 4.8.3. Js: archivos de java script que se utilizan en la página web.
  - 4.9. Templates: archivos HTML que se utilizaran en la página web de la aplicación.
  - 4.10. Upload: esta carpeta es usada por la aplicación para guardar temporalmente los archivos que suba el usuario.
  - 4.11. Vista: esta carpeta contiene ficheros Python con funciones y clases utilizadas por la vista de la aplicación.

### D.3.Manual del programador

En este apartado se exponen aquellos aspectos necesarios para la utilización, desarrollo y despliegue de la aplicación

#### Requisitos del sistema

Para poder desplegar esta aplicación se necesitan una serie de condiciones que se van a exponer a continuación. Es recomendable tener unos conocimientos básicos de informática ya que hay una serie de requisitos que se consideran triviales y no se detalla su instalación.

Para empezar la aplicación necesita un servidor Ubuntu Server 18.04 es recomendable utilizar esta ya que es en la que todos sus componentes se han probado y funcionan. Es recomendable que la máquina tenga al menos dos CPU y 8 GB de memoria RAM. También sería deseable por cuestiones de velocidad que dispusiese de discos duros de estado sólido ya que aumenta muy significativamente la velocidad de la aplicación. En cuanto a la capacidad de los discos dependerá de los datos que se vayan a cargar, con 50 GB se puede trabajar con un país y una gran cantidad de datos. Si queremos cargar la base de datos de OSM mundial en Nominatim al menos necesitaríamos 2TB. Por supuesto es necesario tener una conexión a internet al menos durante las fases de instalación.

Una vez tengamos una máquina de estas características necesitaremos en primer lugar instalar Python 3.6. para esto recomiendo la utilización de Conda o Miniconda ya que nos permiten seleccionar fácilmente esta versión e instalara correctamente todas las librerías a excepción de “*prefixspan*”. El resto de las librerías requeridas son las siguientes.

- flask
- flask-wtf
- wtforms
- geopandas
- requests
- geoalchemy2
- scikit-learn
- networkx

La librería que nos faltaría, “*prefixspan*” se puede utilizar el siguiente comando “*pip install prefixspan*” para instalarla.

También necesitaremos el sistema gestor de bases de datos PostgreSQL con PostGIS la extensión para bases de datos geoespaciales. Y la aplicación Nominatim la cual también necesita PostgreSQL. Nominatim además de esto necesita PHP y Apache Server, posteriormente detallaremos como instalar Nominatim y todos sus requisitos. Con esto ya tendríamos todos los requisitos.

#### Desarrollo de la aplicación

Para continuar con el desarrollo de la aplicación se necesitan los mismos requisitos que para hacerla funcionar. Dado que es una aplicación Python que es un lenguaje interpretado se puede trabajar directamente sobre sus ficheros. Si que es recomendable utilizar una herramienta IDE que nos facilite la edición del código. Mi recomendación es usar Visual Studio Code, este IDE permite la utilización de una gran cantidad de lenguajes. Esto puede resultar muy útil en un proyecto como este donde se mezclan múltiples lenguajes.

Por último, hay un punto muy importante que hay que tener en cuenta si lo que queremos es hacer tareas de desarrollo con la aplicación. En “*src/run.py*” al final del fichero encontraremos las siguientes líneas.

```
if __name__=="__main__":
    app.run(host='0.0.0.0',port=8001,debug=True)
```

*Ilustración 13 Debug*

Es muy importante que donde pone “debug=True” este así solo cuando la aplicación este ejecutándose para pruebas. Si se quiere volver a poner en explotación cambiar a “False”.

### Instalación

#### *Nominatim*

La instalación de Nominatim que vamos a explicar a continuación parte de un sistema operativo Ubuntu Server 18.04 completamente limpio. Es importante que antes de empezar con la instalación y configuración de Nominatim nos aseguremos que tenemos acceso a internet. Las instrucciones de instalación están sacadas de la [documentación oficial](#) de Nominatim.

Vamos a empezar actualizando los paquetes mediante el comando que viene a continuación.

**“sudo apt-get update -qq”**

Una vez tenemos todos los paquetes actualizados vamos a proceder a instalar todos los paquetes necesarios para que Nominatim pueda ser instalado y pueda funcionar correctamente. El siguiente comando contiene todos los paquetes necesarios para instalar Nominatim.

**“sudo apt-get install -y build-essential cmake g++ libboost-dev libboost-system-dev libboost-filesystem-dev libexpat1-dev zlib1g-dev libxml2-dev libbz2-dev libpq-dev libproj-dev postgresql-server-dev-10 postgresql-10-postgis-2.4 postgresql-contrib-10 postgresql-10-postgis-scripts apache2 php php-pgsql libapache2-mod-php php-intl git”**

Una vez tenemos todos los paquetes necesarios para que funcione Nominatim tenemos que crear un usuario dedicado. Con el siguiente comando podemos crear el usuario, lo que hacemos es indicarle donde se ubica su directorio home y que Shell va a utilizar.

**“sudo useradd -d /srv/Nominatim -s /bin/bash -m nominatim”**

A continuación, tenemos que dar permisos al usuario que acabamos de crear a esta carpeta.

**“sudo chmod a+x /srv/nominatim”**

Ahora que tenemos el usuario de Nominatim listo tenemos que configurar PostgreSQL. Dependiendo del volumen de datos que vallamos a manejar puede ser necesario hacer cambios en el siguiente fichero.

**“/etc/postgresql/10/main/postgresql.conf”**

Para la instalación que utilizamos en este proyecto en la cual solo cargamos una parte de los datos de OSM no sería necesario. En caso de que queramos modificar estos parámetros de configuración en el siguiente enlace podemos ver una serie de recomendaciones. Estas recomendaciones están pensadas para un servidor con 32GB de memoria RAM y en el que se va a cargar OSM completo. <http://nominatim.org/release-docs/latest/admin/Installation/#postgresql-tuning>

Si hemos hecho cambios antes de continuar debemos reiniciar el servicio de PostgreSQL mediante el siguiente comando.

**“systemctl restart postgresql”**

Para que Nominatim pueda utilizar PostgreSQL tenemos que crear en este un usuario con el mismo nombre. Este usuario tiene que tener permisos de edición dado que es el que va a cargar los datos.

**“sudo -u postgres createuser -s nominatim”**

También hay que crear otro usuario solo con permisos de lectura para la aplicación web.

**“sudo -u postgres createuser www-data”**

Ahora pasamos a configurar apache. Para configurar nomínate en apache lo podemos hacer mediante el siguiente comando.

**“sudo tee /etc/apache2/conf-available/Nominatim.conf <<  
EOFAPACHECONF”**

A continuación, nos deja meter los datos de configuración que son los siguientes.

```
<Directory "$USERHOME/Nominatim/build/website">  
Options FollowSymLinks MultiViews  
AddType text/html .php  
DirectoryIndex search.php  
Require all granted  
</Directory>
```

**Alias /nominatim \$USERHOME/Nominatim/build/website  
EOFAPACHECONF”**

La última línea cierra y guarda lo que hemos escrito. Esto también se puede hacer abriendo nano u otro editor de textos. Para activar la configuración utilizaremos el siguiente comando.

**“sudo a2enconf nominatim”**

Cuando ejecutamos este comando apache nos pide reiniciar. Lo podemos hacer con el siguiente comando.

**“sudo systemctl restart apache2”**

Una vez hemos hecho estas configuraciones podemos empezar con la instalación propiamente dicha de Nominatim. Para instalar Nominatim es importante que estemos en el sistema autenticados con el usuario nominatim. Para poder autenticarnos con el usuario nominatim antes tenemos que ponerle una contraseña con el siguiente comando.

**“sudo passwd nominatim”**

Ahora que ya tenemos una contraseña asignada podemos entrar mediante el siguiente comando con el usuario nominatim.

**“su nominatim”**

Ahora nos moveremos a su directorio home que es donde va a estar instalado Nominatim.

**“cd /srv/nominatim”**

Ahora descargamos Nominatim con el siguiente comando.

**“wget https://nominatim.org/release/Nominatim-3.2.0.tar.bz2”**

El siguiente paso es descomprimir Nominatim e instalarlo, para descomprimirlo podemos usar el comando.

**“tar -xjvf Nominatim-3.2.0.tar.bz2”**

Creamos un nuevo directorio donde instalaremos Nominatim

**“mkdir Nominatim”**

Ahora creamos la carpeta build.

**“mkdir Nominatim/build”**

Nos movemos a la carpeta build

**“cd Nominatim/build/”**

Ahora nos toca instalar Nominatim para esto utilizamos primero el siguiente comando.

**“cmake /srv/nominatim/Nominatim-3.2.0”**

Para instalarlo utilizamos el comando.

**“make”**

Con esto ya tenemos instalado Nominatim solo falta cargar los datos y terminar de configurar. Los datos se pueden descargar desde la página que aparece en el comando que vamos a utilizar a continuación.

**“wget https://download.geofabrik.de/asia/china-latest.osm.bz2”**

Ahora procedemos a descomprimir los datos para poder cargarlos en Nominatim.

**“bzip2 -d china-latest.osm.bz2”**

Para cargar estos datos en la aplicación utilizaremos el siguiente comando, pero debe tener en cuenta que esto puede tardar horas e incluso días si no se tiene un disco duro SSD.

```
“./utils/setup.php –osm-file china-latest.osm –all 2>&1 | tee setup.log”
```

Para terminar de instalar Nominatim debemos utilizar el siguiente comando para añadir la última parte de la configuración.

```
“tee settings/local.php << EOF  
<?php  
@define('CONST_Website_BaseURL', '/nominatim/');  
EOF”
```

Con esto queda concluida la instalación de Nominatim para probarlo podemos conectarnos al servidor de la siguiente forma en un navegador. Hay que sustituir las “x” por la IP del servidor.

<http://xxx.xxx.xxx.xxx/nominatim>

#### Aplicación

Lo primero que aremos es crear la base de datos para esto hay que ejecutar los siguientes códigos en la aplicación que utilicemos para gestionar la base de datos. No se deben ejecutar las comillas iniciales y finales y además si la herramienta que usamos para gestionar la base de datos es en línea de comando como “psql” será necesario escribirlo todo en una línea. Es recomendable hacer esto con el usuario postgres.

```
“CREATE DATABASE "DB_Trayectorias_Semanticas"  
WITH  
OWNER = postgres  
ENCODING = 'UTF8'  
CONNECTION LIMIT = -1;”
```

Ahora tenemos que añadir la extensión de PostGIS para ello utilizamos la siguiente sentencia.

```
“CREATE EXTENSION postgis;”
```

Por último, nos queda crear las tablas para esto hay que utilizar el siguiente comando.

```
“CREATE TABLE public.usuario  
(  
id_usuario bigint NOT NULL,  
PRIMARY KEY (id_usuario)  
)  
WITH (  
oids = FALSE  
);
```

```
ALTER TABLE public.usuario  
OWNER to postgres;
```

```
CREATE TABLE public.trayectoria  
(  
    id_usuario bigint NOT NULL,  
    id_trayectoria bigint NOT NULL,  
    PRIMARY KEY (id_trayectoria),  
    CONSTRAINT id_usuario FOREIGN KEY (id_usuario)  
        REFERENCES public.usuario (id_usuario) MATCH SIMPLE  
        ON UPDATE CASCADE  
        ON DELETE CASCADE  
)  
WITH (  
    oids = false  
);
```

```
ALTER TABLE public.trayectoria  
OWNER to postgres;
```

```
CREATE TABLE public.punto  
(  
    id_trayectoria bigint NOT NULL,  
    id_punto bigint NOT NULL,  
    instante timestamp(4) without time zone NOT NULL,  
    estado bigint NOT NULL,  
    punto geometry NOT NULL,  
    PRIMARY KEY (id_punto, id_trayectoria),  
    CONSTRAINT id_trayectoria FOREIGN KEY (id_trayectoria)  
        REFERENCES public.trayectoria (id_trayectoria) MATCH  
simple  
        ON UPDATE CASCADE  
        ON DELETE CASCADE  
)  
WITH (  
    oids = false  
);
```

```
ALTER TABLE public.punto  
OWNER to postgres;
```

```
CREATE TABLE public.parado  
(  
    punto geometry NOT NULL,
```



```

    instante_inicio timestamp without time zone NOT NULL,
    instante_fin timestamp without time zone NOT NULL,
    id_parada bigint NOT NULL,
    id_trayectoria bigint NOT NULL,
    id_osm bigint,
    PRIMARY KEY (id_trayectoria, id_parada),
    CONSTRAINT id_trayectoria FOREIGN KEY (id_trayectoria)
        REFERENCES public.trayectoria (id_trayectoria) MATCH
SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
WITH (
    OIDS = FALSE
);

ALTER TABLE public.parado
    OWNER to postgres;

```

Con esto ya esta la base de datos lista para funcionar ya solo queda descargar los ficheros de la aplicación y configurarla. Para descargar el proyecto podemos recurrir al siguiente comando.

**“git clone**

**[https://github.com/hca0004/GII17.0T\\_Trayectorias\\_Semanticas\\_1819.git](https://github.com/hca0004/GII17.0T_Trayectorias_Semanticas_1819.git)”**

Ahora dentro de la carpeta del proyecto que nos ha descargado debemos editar el fichero SQLConexion.py este se encuentra en la ruta “./src/SQL” dentro del proyecto. Es importante que otros usuarios del sistema no tengan acceso a este fichero ya que contiene la contraseña de la base de datos en texto plano. En la siguiente imagen tenemos un ejemplo de configuración estos campos, que se encuentran dentro de la función “conexionBDApp”.

```

usuario = "postgres"
contrasena="123456"
ip="localhost"
puerto="5432"
db="DB_Trayectorias_Semanticas"

```

*Ilustración 14 Configuración Aplicación Base de Datos*

También debemos comprobar cómo está configurada la conexión con Nominatim. Para esto nos dirigimos dentro del directorio de la aplicación a la ruta “./src/nominatim” y aquí editaremos el fichero “peticion.py”. este no hace falta tocarlo si está el servidor de Nominatim en el mismo servidor que la aplicación. En caso contrario debemos modificar la siguiente línea y modificar “localhost” por la IP del servidor.

```
10 ipNominatim="localhost"
```

*Ilustración 15 Configuración IP Nominatim*

Por último, en “./src” el fichero “run.py” en la última línea tiene la siguiente configuración.

```
app.run(host='0.0.0.0',port=8001,debug=False)
```

*Ilustración 16 Configuración IP Salida*

En esta línea se indica con `host="0.0.0.0"` que se publique en todas las direcciones IP del servidor y en `port=8001` el puerto en el que se va a publicar la aplicación web. Estos dos pueden ser modificados o se pueden dejar así.

Con esto la aplicación ya está instalada ahora si queremos ejecutarla se debe lanzar el fichero “run.py” con Python. Para esto recomiendo utilizar también el comando “nohub” para evitar que se cierre si cerramos nuestra sesión en el servidor. A continuación, tenemos un ejemplo en el cual habría que sustituir la ruta del archivo en función de donde tengamos la aplicación.

**“nohub python GII17.0T\_Trayectorias\_Semanticas\_1819/src/run.py &”**

#### D.4. Pruebas del sistema

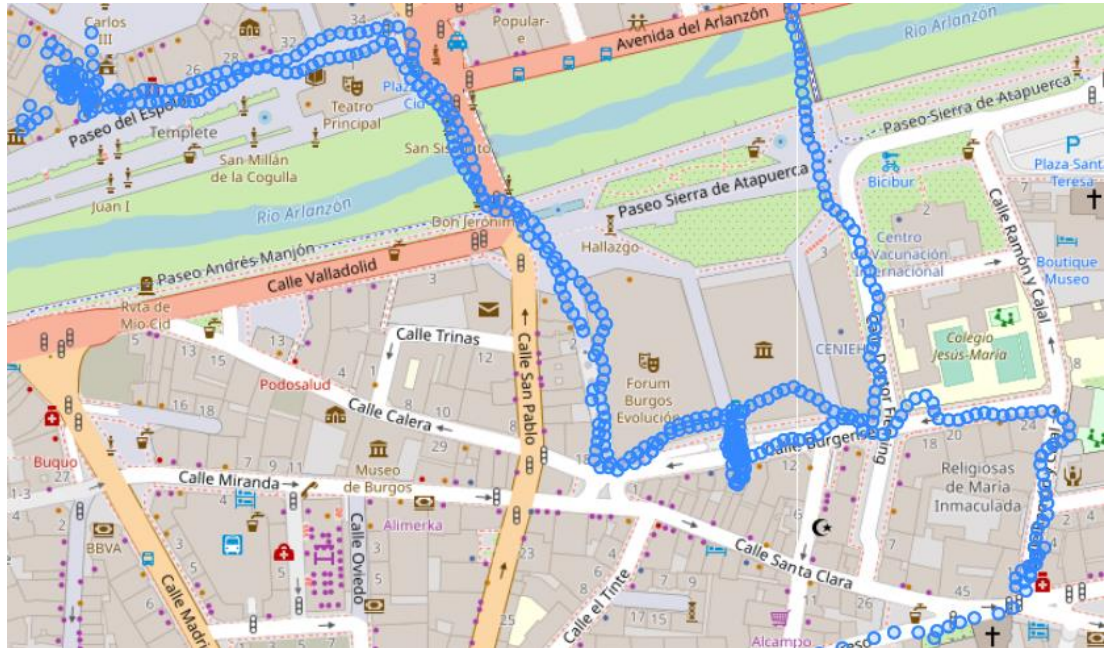
En este apartado se exponen las pruebas que se han realizado en el sistema. Sobre todo, nos centraremos en las pruebas realizadas sobre los algoritmos. En algunas clases como en “Trayectoria.py” se realizaron pruebas unitarias con doctest. Que nos permitía añadir las pruebas unitarias en los comentarios de documentación de las funciones. Posteriormente se vio que los test unitarios no cumplían nuestras necesidades reales y requerían una gran cantidad de tiempo y esfuerzo su implementación.

##### Pruebas de detección de paradas

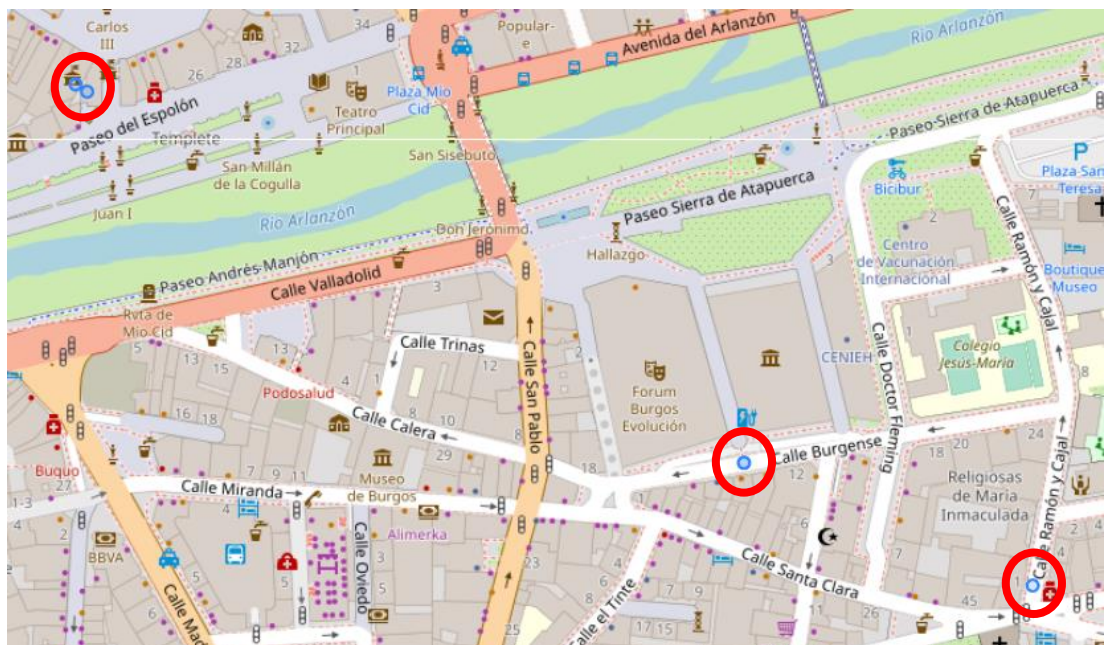
Pruebas sobre los algoritmos de detección de paradas. Ya que carecíamos de información para determinar si en los datos que teníamos realmente había paradas. Y como no podíamos calcular a mano los miles de líneas tratando de averiguar si la razón por la que detecta las paradas debía a los datos o a fallos del algoritmo optamos por obtener datos en los cuales si pudiésemos saber si realmente había paradas o no. Para esto tome datos sobre mí mismo en los que vistos sobre un plano podía determinar si estaba parado o no en función de la ubicación.

Para comparar las paradas utilizamos las propias funciones “pgAdmin 4” una herramienta web para trabajar con servidores de bases de datos Postgres. Con esta herramienta en la cual cuando realizamos una *SELECT* sobre los datos geoespaciales aparece una pestaña “Geometry Viewer”. En esta pestaña nos representa los datos geoespaciales.

En las siguientes ilustraciones tenemos un ejemplo de cómo probamos el algoritmo. La primera es una representación de los datos en bruto y la segunda de las paradas detectadas. En la segunda imagen podemos ver que hay tres zonas con paradas de izquierda a derecha, la primera es en el ayuntamiento de burgos donde he estado realizando las practicas. La segunda es donde vivo en la calle del burgense N° 8 y la tercera corresponde con la ubicación en la que tenía aparcado el coche.



*Ilustración 17 Trayectoria en Bruto*



*Ilustración 18 Trayectoria Conceptual*

Durante estas pruebas se observó que también pueden producirse paradas en zonas que no son necesariamente paradas, como en los semáforos. Por lo cual para evitar estas paradas que no consideramos como tal lo que se hace es quedarnos con aquella con una duración de “x” en lugar de usar todas.

#### Pruebas clasificador

Para probar el clasificador lo primero que se hizo fue probar que para el ejemplo del artículo del que esta sacado el algoritmo da la misma predicción y la misma ruta. Hecho esto pasamos a evaluar el algoritmo para esto sirve la clase probador la cual permite hacer muchos entrenamientos con validación y guardando los resultados. Al final puedes pintar los resultados en graficas como las siguientes.

FMeasure es una métrica que mide la tasa de aciertos y de fallos teniendo en cuenta que proporción de los datos se han utilizado para para clasificar.

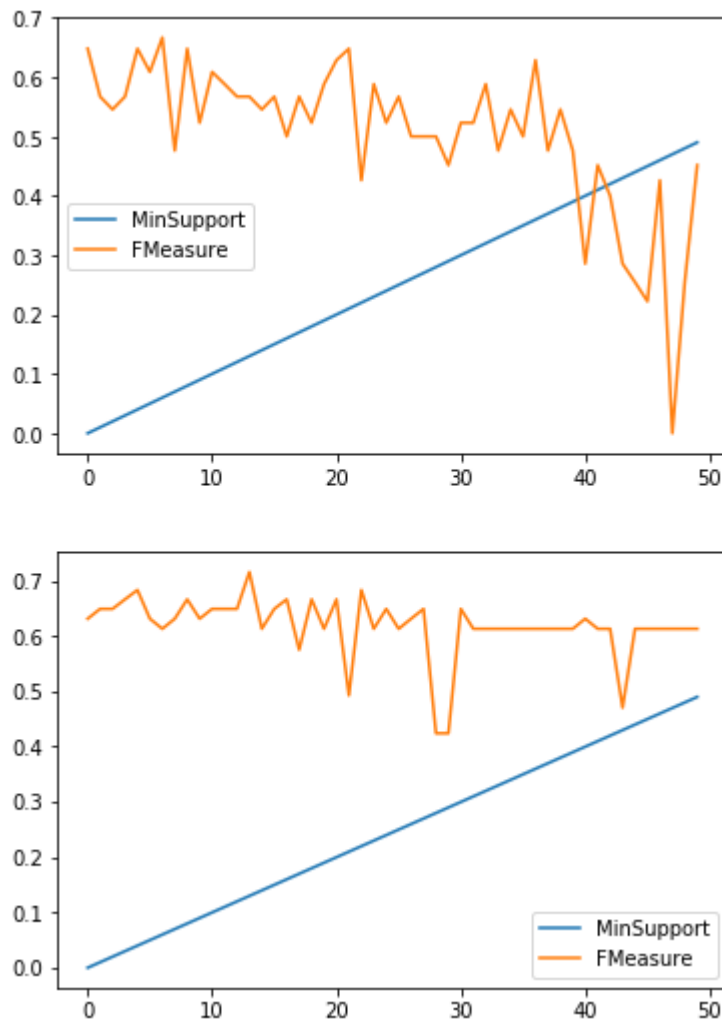


Ilustración 19 Gráficos FMeasure Predicciones

Estos resultados podemos ver no solo si acierta mucho o poco algo que resulta bastante útil, si no la calidad de lo que estamos tratando de predecir. En el primer caso vemos que va variando el número de aciertos a medida que se reduce el soporte mínimo (el soporte mínimo es el porcentaje de las rutas en las que está presente una determinada secuencia) en el segundo no. Esto nos hace prever que en segundo caso o bien las rutas son muy completas y tiene información de todo lo que hacen o siempre está prediciendo un número muy reducido de cosas. En estos casos nos solemos encontrar con que lo que predice son cosas triviales.

Si desciende bastante a medida que aumentamos el soporte necesario es porque está prediciendo clases que aparecen poco lo que es un muy buen resultado porque podremos predecir más clase y mejor.

#### Pruebas clustering

En estas pruebas no pudimos sacar datos concluyentes sobre los algoritmos. Con PCA representando los datos (a los que previamente quitamos las columnas sin datos) en dos dimensiones obtuvimos el siguiente resultado.

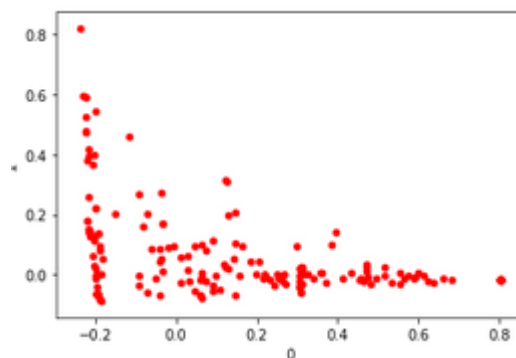
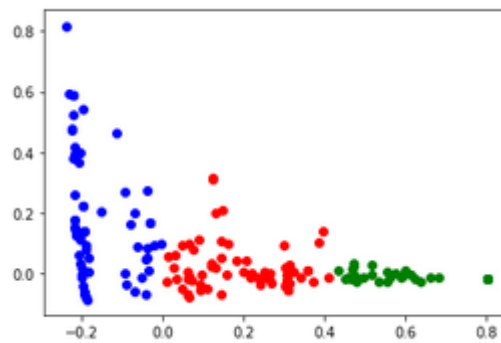
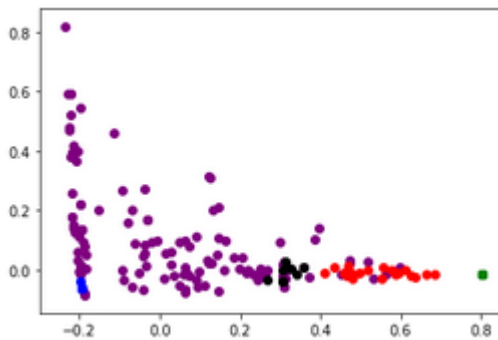
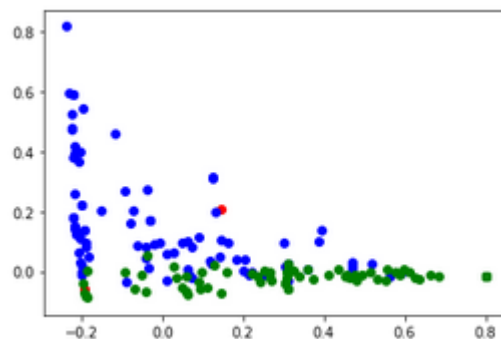


Ilustración 20 PCA 2D

En el eje X está representada el componente principal mientras que en el eje Y está representada la segunda componente estas tienen un peso de 0.60 la principal y 0.10 la secundaria. Con esta representación probamos a utilizar distintos algoritmos (KMeans, DBSCAN y GaussianMixture) variando sus configuraciones y obtuvimos unos resultados los cuales no coincidían con los esperados. Esto teniendo en cuenta que los datos utilizados eran de los usuarios entre semana y los fines de semana y no pudimos separarlos en función de esto.

*Ilustración 21 KMeans 3 Clústeres**Ilustración 22 DBSCAN**Ilustración 23 GaussianMixture*

Analizando un poco más afondo y obteniendo los centros de los clústeres en KMeans quisimos saber a qué datos de usuario corresponderían. Obtuvimos que la componente principal estaba principalmente marcada por el número de veces que un usuario se desplaza de una calle a otra. Esto parece indicarnos que la calidad de los datos es muy variable ya que la razón por las que identifica en estos lugares como paradas suele deberse a que no hay otros puntos de interés próximos.

Esto también explica porque obteníamos los resultados que obtuvimos en las predicciones en las cuales para algunos usuarios no variaba casi nada la tasa de aciertos con el soporte. Ya que si el 90% de los puntos son calles y predigo siempre que van a ser calles tendré una tasa de acierto del 90%.



No ha habido tiempo para tratar de averiguar las razones exactas de este resultado, pero parece indicar que podría ser una de las razones por las que el algoritmo de predicción tiene unos resultados tan dispares entre usuarios.

```

CLUSTER 0 Azul
va de  tourism a highway el 0.746 % de las veces
va de  highway a tourism el 0.605 % de las veces
va de  highway a highway el 1.146 % de las veces
va de  highway a building el 1.779 % de las veces
va de  highway a amenity el 0.851 % de las veces
va de  building a highway el 1.875 % de las veces
va de  building a building el 4.115 % de las veces
va de  building a amenity el 2.317 % de las veces
va de  amenity a highway el 0.947 % de las veces
va de  amenity a building el 2.087 % de las veces
va de  amenity a amenity el 3.384 % de las veces
CLUSTER 1 Rojo
va de  tourism a highway el 4.821 % de las veces
va de  highway a tourism el 4.6 % de las veces
va de  highway a highway el 38.892 % de las veces
va de  highway a building el 3.008 % de las veces
va de  highway a amenity el 6.138 % de las veces
va de  building a highway el 3.369 % de las veces
va de  building a building el 3.626 % de las veces
va de  building a amenity el 2.272 % de las veces
va de  amenity a highway el 6.916 % de las veces
va de  amenity a building el 1.572 % de las veces
va de  amenity a amenity el 4.284 % de las veces
CLUSTER 2 Verde
va de  tourism a highway el 0.589 % de las veces
va de  highway a tourism el 0.517 % de las veces
va de  highway a highway el 86.098 % de las veces
va de  highway a building el 1.129 % de las veces
va de  highway a amenity el 2.168 % de las veces
va de  building a highway el 2.04 % de las veces
va de  building a building el 0.07 % de las veces
va de  building a amenity el 0.282 % de las veces
va de  amenity a highway el 2.83 % de las veces
va de  amenity a building el 0.379 % de las veces
va de  amenity a amenity el 0.857 % de las veces

```

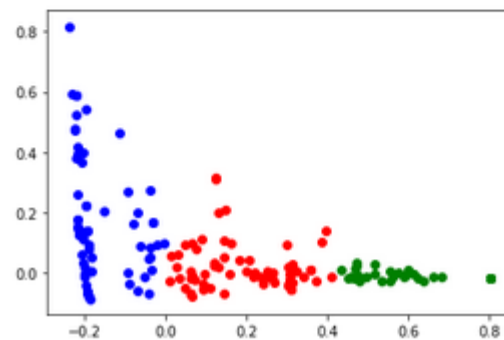


Ilustración 24 Información del Clúster

---

# Documentación de usuario

---

## E.1. Introducción

En este apartado se va a exponer toda la información que van a necesitar los usuarios de la aplicación que definimos en los casos de uso.

## E.2. Requisitos de usuarios

En el caso de los usuarios normales lo único que requieren es de un navegador. La aplicación es compatible con los principales navegadores Firefox, Chrome, Edge y Opera.

Para acceder a la página web, el administrador del servidor donde está montada la aplicación deberá comunicarle la ruta web de acceso a esta Ej. <http://10.0.0.10:8001>

Los requisitos para los datos que puede el usuario cargar en la aplicación son archivos de tipo CSV que contenga al menos las siguientes columnas latitud, longitud y instante de tiempo en el que fue tomado. En el directorio “data” de la aplicación hay ejemplos de archivos CSV. Los archivos se cargan en la aplicación por directorios y se considera que cada directorio pertenece a un usuario distinto.

Por último, el usuario puede requerir de unos conocimientos básicos de consultas SQL para poder exportar los datos de la aplicación.

## E.3. Instalación

Esta aplicación no requiere de instalación por parte del usuario solo acceso a la red en la que este publicada la página web de la aplicación y la dirección de esta. La instalación solo necesita hacerla el administrador del sistema.

Dentro de la instalación podríamos considerar como tal la actualización y agregación de datos geoespaciales de OSM en Nominatim.

Los datos se pueden descargar desde la página <https://download.geofabrik.de>

Reitero que para instalarlos como ya se explicó en el apartado de instalación en el manual del programado se hace de la siguiente forma.

Ejemplo de descarga:

```
“wget https://download.geofabrik.de/asia/china-latest.osm.bz2”
```

Ejemplo de descompresión:

```
“bzip2 -d china-latest.osm.bz2”
```

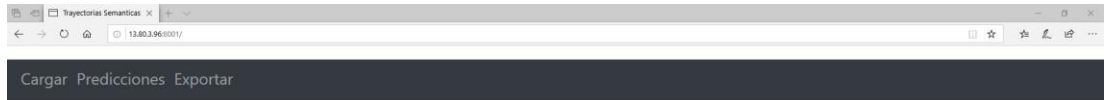
Ejemplo de cómo se instalan:

```
“/srv/nominatim/Nominatim/build/utls/setup.php –osm-file china-latest.osm –all 2>&1 | tee setup.log”
```



#### E.4. Manual del usuario

En este apartado se expone como ha de usarse la aplicación por parte del usuario. Lo primero es acceder a la dirección en la que se aloja la aplicación. Nos aparecerá una ventana vacía como la que aparece en la siguiente ilustración



*Ilustración 25 Index*

En esta ventana tenemos tres opciones para trabajar con la aplicación. La primera es Cargar que nos permite añadir nuevos datos a la aplicación. La segunda es Predecir que nos permite utilizar el algoritmo de predicción integrado en la aplicación. Y la tercera es Exportar la cual nos permite extraer datos de la aplicación.

##### Cargar

Para cargar los datos primero hay que estudiar cómo están estructurados en los archivos CSV. En la ilustración inferior tenemos el ejemplo de los ficheros que se van a cargar.

```
ele,time,_lat,_lon
849.9,2019-02-12T19:58:09Z,42.33853287,-3.69739709
871.4,2019-02-12T19:58:28Z,42.33846773,-3.69733858
875.6,2019-02-12T20:07:46Z,42.33848697,-3.69734723
890.8,2019-02-12T20:36:25Z,42.33853066,-3.69730271
892.4,2019-02-12T20:48:54Z,42.33851092,-3.69736499
895.3,2019-02-12T21:02:12Z,42.33850821,-3.69729767
```

*Ilustración 26 Ejemplo CSV*

Podemos ver que tiene cuatro columnas de las cuales nos interesan las tres últimas. Para saber que columnas hay que indicarle a la aplicación empezamos a contarlas desde 0. La complicación reside en especificar el formato a la fecha, en este [enlace](#) podemos ver todos los detalles pero vamos a explicar los que a priori podríamos necesitar.

Tabla 29 Tokens Fechas

Token	Componente de fecha	Formato
%d	Día del mes	01, 02, ..., 31
%m	Numero de mes	01, 02, ..., 12
%Y	Año	0000, 0001, ..., 2019, ..., 9999
%H	Hora	00,01, ..., 23
%M	Minuto	00, 01, ..., 59
%S	Segundos	00, 01, ..., 59

Con esta tabla ya podemos construir el formato de la fecha, vamos a ver el ejemplo que aparece en el CSV de ejemplo. Tenemos “2019-02-12T19:58:09Z” con esto lo único que hay que hacer es cambiar el número que representa a cada dato por el token de la tabla. El resto de los caracteres se dejan tal y como aparecen, el resultado es “%Y-%m-%dT%H:%M:%SZ”. Una vez analizado el CSV podemos volver a la ventana en la que aparece el siguiente formulario en la parte central.

### Cargar archivos de datos

Columna de la longitud\*

Columna de la latitud\*

Columna de tiempo 1\*

Formato del tiempo 1\*

Columna de tiempo 2

Formato del tiempo 2

Fila de inicio de los datos\*

Extensión de los archivos\*

Los campos con "\*" son obligatorios

Ilustración 27 Formulario Cargar

Para empezar, nos aparece un recuadro el cual nos dejara seleccionar el directorio que queremos cargar en la aplicación. Es de este del que sacara los archivos con la extensión que especificamos. En los tres siguientes campos hay que poner la columna y en el cuarto el formato de la fecha que se ha explicado anteriormente. Al final nos pide que especifiquemos en que línea queremos que empiece a leer (empezando desde 0) y que extensión tiene los archivos. Los campos marcados con un asterisco son obligatorios.

Si hay algún campo con formato invalido o vacío en el caso de los obligatorios nos aparecerá un mensaje de error a final del formulario como el de la siguiente ilustración.

Se debe introducir una al menos una columna válida para el tiempo

Ilustración 28 Cargar mensaje error

Si todo está bien nos aparecerá una ventana de carga esta ventana puede tardar en aparecer si estamos cargado muchos datos en el servidor.



Ilustración 29 Cargar Espera

Al terminar aparecerá una tabla con un resumen de los datos por cada usuario que se añade a la base de datos.

Resultado			
Exportar			
id_usuario	numero_trayectorias	numero_paradas	numero_puntos
211	5	15	2750

Ilustración 30 Carga Tabla Resultado

Esta tabla se puede exportar como archivo CSV pulsando sobre el botón exportar. Si la tabla aparece vacía es porque se han introducido mal los datos para su lectura.

#### Predicciones

Esta ventana tiene tres columnas la primer es para cargar datos, también estrenara un clasificador si no hay ninguno entrenado. La segunda muestra información sobre el clasificador y tiene la opción de predecir y de reiniciar los datos de predicción. En la tercer se mostrará una tabla con el resultado de las predicciones.

A continuación, se muestra el aspecto que tiene la primera columna.

## Cargar Datos

Usuario

Seleccione los días de la semana, si no se marca ninguno se utilizarán todos

☐ L ☐ M ☐ X ☐ J ☐ V ☐ S ☐ D

```
SELECT public.parado.punto, public.parado.instante_inicio,  
public.parado.instante_fin, public.parado.id_parada,  
public.parado.id_trayectoria, public.trayectoria.id_usuario,  
public.parado.id_osm  
FROM public.parado
```

```
INNER JOIN public.trayectoria ON  
public.parado.id_trayectoria =  
public.trayectoria.id_trayectoria
```

```
ORDER BY public.parado.id_trayectoria,  
public.parado.id_parada;
```

Cargar

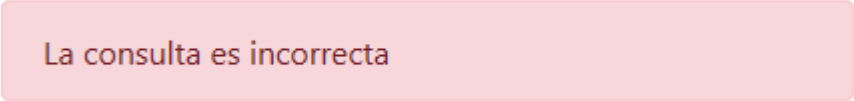
*Ilustración 31 Predicción Cargar*

En esta primera columna podemos construir una SELECT, pero con un formato concreto. El primer campo nos sirve para elegir con que usuario queremos hacer predicciones. Esta suele ser la opción más recomendable y rápida, aunque no es obligatorio solo es obligatorio que sea un número.

Después se permite seleccionar por días de la semana los datos que se va a cargar, por defecto son todos los días.

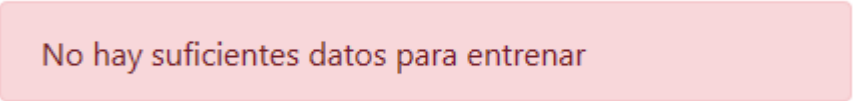
Debajo de días de la semana vemos que aparecen unas consultas SELECT esta la podremos completar teniendo en cuenta algunas partes de la SELECT que no permiten modificaciones

Cuando demos a cargar si la consulta no es válida saldrá el siguiente error.

A red rectangular box with rounded corners containing the text "La consulta es incorrecta" in a dark red font.

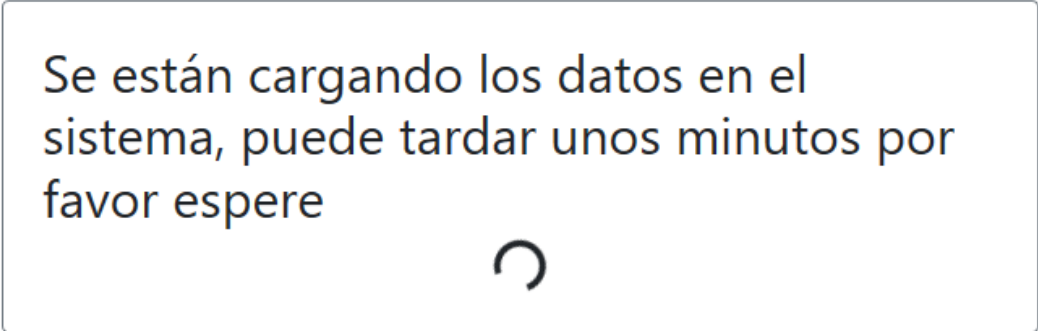
*Ilustración 32 Predicción error SQL*

También nos puede pasar que ese usuario no tenga datos suficientes para entrenar, esto solo pasa cuando no hay clasificador entrenado y sale el siguiente mensaje.

A red rectangular box with rounded corners containing the text "No hay suficientes datos para entrenar" in a dark red font.

*Ilustración 33 Predicción error faltan datos*

Si todo está bien nos aparecerá la siguiente ventana mientras carga

A white rectangular box with rounded corners and a thin purple border. It contains the text "Se están cargando los datos en el sistema, puede tardar unos minutos por favor espere" in a dark blue font. Below the text is a dark grey circular loading spinner.

*Ilustración 34 Predicción espera*

La segunda columna muestra información sobre el clasificador.

### Información de entrenamiento

Numero de trayectorias 5

Aciertos 0

Fallos 5

FMeasure 0.0

Precisión 0.0

Recall 1.0

El clasificador esta entrenado y hay datos para predecir

Predecir

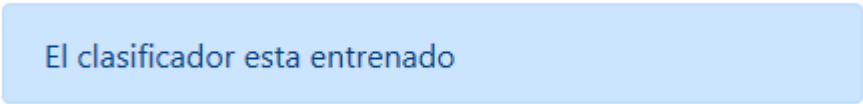
Reiniciar

*Ilustración 35 Predicción información*

La información que muestra es la siguiente, numero de trayectorias utilizadas para entrenar el clasificador. Numero de aciertos en la validación cruzada, numero de fallos, FMeasure que es una medida para estimarlos fallos producidos en función del número de ejemplos que se utilizan para entrenar y el número de ejemplos utilizados para clasificar. La precisión que tasa de aciertos ha tenido y el recall que cuantifica la relación entre datos de entrenamiento y los usados para predecir. Al ser una validación cruzada siempre nos da un valor de uno.

Al final tenemos el botón para predecir, que solo se habilita si además de los datos de entrenamiento se han vuelto a cargar datos esta vez para predecir. Y otro botón para reiniciar todos los parámetros de la ventana.

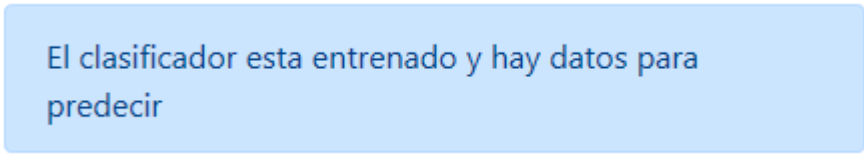
En esta columna pueden aparecer los siguientes mensajes que nos informan del estado del clasificador



El clasificador esta entrenado

*Ilustración 36 Información estado 1*

Si aparece este segundo podremos predecir.



El clasificador esta entrenado y hay datos para predecir

*Ilustración 37 Información estado 2*

En la tercera columna aparecerá una tabla con las predicciones después de dar a predecir, si no, estará en blanco.



### Resultado de la predicción

## Resultado

Exportar

<b>Id Usuario</b>	<b>Id Trayectoria</b>	<b>Trayectoria Categoría (3 ultimos)</b>	<b>Prediccion</b>
122	20531	highway highway highway	highway
122	20536	highway highway highway	highway
122	20572	highway highway highway	highway
122	20686	highway highway highway	highway
122	20701	building amenity highway	highway

*Ilustración 38 Tabla de predicciones*

Esta tabla muestra la ID del usuario, de la trayectoria, la categoría de las ultimas paradas de la trayectoria y la clase que se ha predicho. También podemos exportarnos esta tabla como un CSV.

Exportar

Tiene dos columnas la primera permite introducir consulta SELECT y nos la devolverá en forma de CSV hay que tener en cuenta que la aplicación convierte las

columnas geospaciales en latitud y longitud por lo que no se recomienda renombrar una columna con el mismo nombre de estas, que es “punto”.

Introduzca una consulta SELECT de los datos que desea exportar

Exportar

Ilustración 39 Exportar SELECT

Si la sentencia no es válida aparecerá el siguiente mensaje en caso contrario devuelve un CSV.

La consulta no es valida

Ilustración 40 Exporta consulta invalidar

La segunda columna no es mas que una imagen con el diagrama de la base de datos para facilitar la realización de consultas.

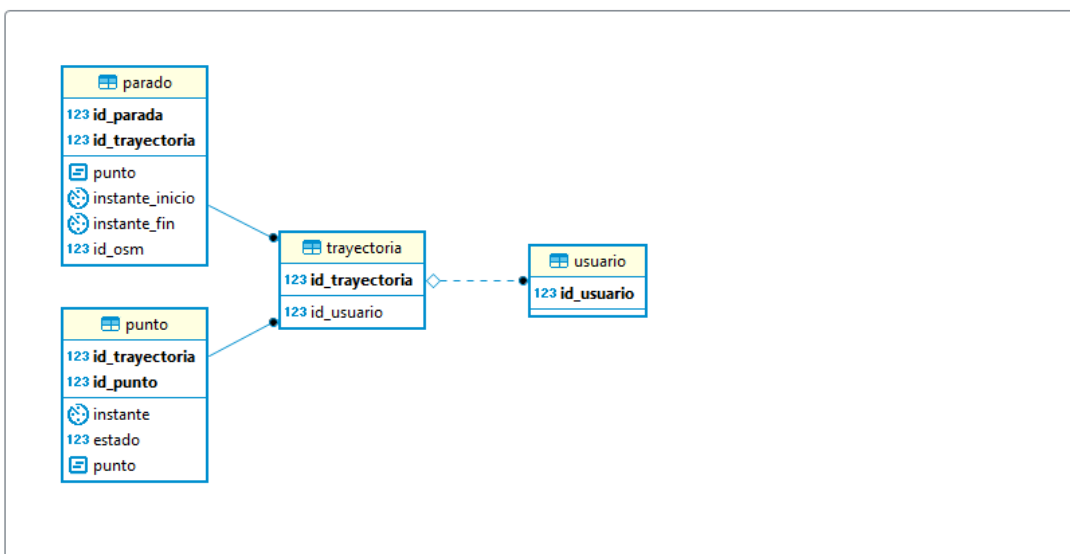


Ilustración 41 Exportar Diagrama