INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Ambienta2MX”

2014 – B073

Que para cumplir con la opción de titulación curricular en la carrera de:

“Ingeniería en Sistemas Computacionales”

Presentan

Eduardo Gamaliel Jiménez García
Juan Alberto Reséndiz Arteaga

Directores

M. en C. ***Mario Augusto Ramírez Morales*** **M. en E.** ***Carlos Silva Sánchez***

Diciembre 2015

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

No. de Registro: 2014-B073

Diciembre de 2015

Documento técnico

“Ambienta2MX”

Presentan

Eduardo Gamaliel Jiménez García[1]

Juan Alberto Reséndiz Arteaga[2]

Directores

M. en C. Ramírez Morales Mario Augusto

M. en C. Silva Sánchez Carlos

RESUMEN

Resumen – "Ambienta2MX" es una plataforma de análisis, estandarización y muestreo de información ambiental como variables de temperatura, humedad, presión e índices de contaminación. El sistema se compone del módulo de consulta, registro masivo y estandarización de datos relacionados con el medio ambiente. Ésta plataforma permitirá conceptualizar la información de variables ambientales e índices de contaminación almacenada por el INEGI y otras instituciones públicas o privadas. Dicha información puede ser utilizada para fines educativos, productivos y para la sociedad en general. Los usuarios finales podrán hacer uso de Ambienta2MX a través de una página de internet; la interacción se realizará a través de servicios web tipo REST con un estándar de datos propuesto por el equipo de trabajo

Palabras clave – Desarrollo web, Sistemas Distribuidos, Aplicaciones para las comunicaciones en red.

1 egjimenezg@gmail.com

2 jresendiz27@gmail.com

ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE FORMACIÓN INTEGRAL
E INSTITUCIONAL

COMISIÓN ACADÉMICA DE TRABAJO TERMINAL

México D.F., Noviembre de 2015

DR. FLAVIO ARTURO SÁNCHEZ GARFIAS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJO TERMINAL
P R E S E N T E

Por medio del presente, se informa que los alumnos que integran el trabajo terminal 2014-B073 titulado “Ambienta2MX” concluyeron satisfactoriamente su trabajo.

Los discos (DVDs) fueron revisados ampliamente por sus servidores y corregidos, cubriendo el alcance y el objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE:

M. en E. *Silva Sánchez Carlos*

M. en C. *Ramírez Morales Mario Augusto*

Advertencia

“Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan.”

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000 Extensión 52000

Índice general

Capítulo 1

Introducción

Ambienta2MX pretende ser una plataforma única en su tipo a nivel nacional, proporcionando la infraestructura lógica, modelo de datos, diagramas y esquemas lógicos necesarios para proveer un servicio de consulta de datos climatologicos e índices de contaminación a través de un portal web y/o servicios expuestos, para fines privados, públicos o sociales en general.

Han existido diversos diversos sistemas que proveen información semejante o han intentado dar solución a la problemática expuesta no solo considerando datos ambientales cómo la temperatura sino geográficos, geodésicos, topográficos, etcetera; sin embargo, no han tenido el impacto o el apoyo necesario para crecer y proveer la infraestructura lógica y/o física para satisfacer ese problema.

Para llegar a la solución que será descrita a lo largo de éste documento fue necesario considerar un grupo específico de soluciones existentes, que van desde sistemas totalmente orientados a la estandarización, hasta soluciones que toman sólo parte del problema completo que pretende atacar Ambienta2MX.

Problemas de este tipo han sido atacados en otros países, teniendo un impacto favorable en la consulta de información de cierta área en específico, casos como el anterior serán mencionados próximamente.

Capítulo 2

Antecedentes

2.1. Sistemas de información geográfica

Un Sistema de Información Geográfica (GIS, por sus siglas en inglés) es un sistema computacional de captura, almacenamiento, chequeo y exposición de datos relacionados a la superficie de la Tierra. Los SIG pueden mostrar información de varios tipos en un mismo lienzo (usualmente mapas) lo que facilita el análisis de la información recolectada (véase Figura 2.1).[5]

Figura 2.1: Sistemas de Información Geográfica. [18]

Los SIGs son una herramienta usada por organizaciones, escuelas, instituciones gubernamentales y negocios. Éstos sistemas pueden ser orientados a datos globales o a una región en específico del globo terraqueo. Entre los datos almacenados más importantes se encuentra la georeferenciación (Longitud, Latitud, Altitud), algunos datos relativos a la zona, por ejemplo, códigos postales.

En México, la institución encargadada del manejo de datos estadíticos y geográficos es el INEGI. El Instituto Nacional de Estadística, Geografía e Informática (INEGI) es la fusión de varias instituciones gubernamentales que funcionaban de forma independiente hasta el año 1983; éstas se encargaban de el manejo de datos estadísticos, comerciales, económicos y financieros.

Actualmente el INEGI, sede se encuentra ubicada en Aguascalientes, Aguascalientes, es la institución encargada de la captación, procesamiento y difusión de información relativa al territorio nacional. [6]

Los datos geoespaciales (Longitud, Latitud, Altitud) que brinda el INEGI se encuentran bajo el estandar ITRF 2008 en época 2010. Sin embargo, aún existen sistemas que se encuentran trabajando bajo el formato ITRF92 época 1988 y NAD27 (formato usado hasta 1998), para ello, se brindará soporte a formatos previos, dando prioridad al último estandar existente.

2.2. Sistemas de Monitoreo del medio Ambiente.

2.2.1. Proyecto en la ciudad de Salamanca

Uno de los proyectos nacionales que realizo un monitoreo usando un PCFM fue realizado en la ciudad de Salamanca- una de las ciudades más contaminadas en México. Este proyecto es titulado "Análisis de la contaminación del aire usando un Algoritmo PCFM aplicado a una base de datos real.

La intención del estudio es analizar la relación entre contaminación y las variables ambientales. En el análisis de este estudio se involucraron datos desde Enero a Diciembre del 2007. Algunas de las variables que se incluyeron fueron S02, y la concentración de partículas menores a 10m y variables meteorológicas. Velocidad del viendo, dirección del viento, temperatura y humedad relativa. A partir de este estudio se instalaron algunos centros de monitoreo en Salamanca.

2.2.2. Inventario Nacional de Emisiones de Gases de Efecto Invernadero (INEGEI)

En este proyecto se elabora un informe que comprende las estimaciones de las emisiones por fuente y sumidero. Esto se realiza de acuerdo a lo conforme establecido en los ar-

tículos 4 y 12 de la Convención marco de las naciones unidas sobre el cambio climático. Este proyecto es una publicación que nos ayuda a saber que deficiencias tenemos, y que efectos pudiera llegar a tener el exceso de algunos contaminantes, sin embargo no es una plataforma para accede a estos datos tan importantes.

2.2.3. Mapa Digital - INEGI

Este proyecto es un sistema de información geográfica que tiene como objetivo facilitar el estudio de los objetos geográficos a través del conocimiento de su ubicación espacio-temporal, así como de atributos asociados (véase Figura 2.2); tales servicios bridan al usuario final la posibilidad de:

- Mostrar en forma gráfica la dimensión de la información contenida por medio de acercamientos, selección de capas de información, localizaciones, mediciones, etc.
- Analizar e interpretar los contenidos geográficos y estadísticos mediante operaciones matemáticas, mapas temáticos, gráficos estadísticos, análisis espacial y estadísticos básicos.
- Integrar información a través de la incorporación de datos vectoriales y raster provenientes de archivos locales, conexiones a servicios WMS y base de datos geospaciales de PostGis.

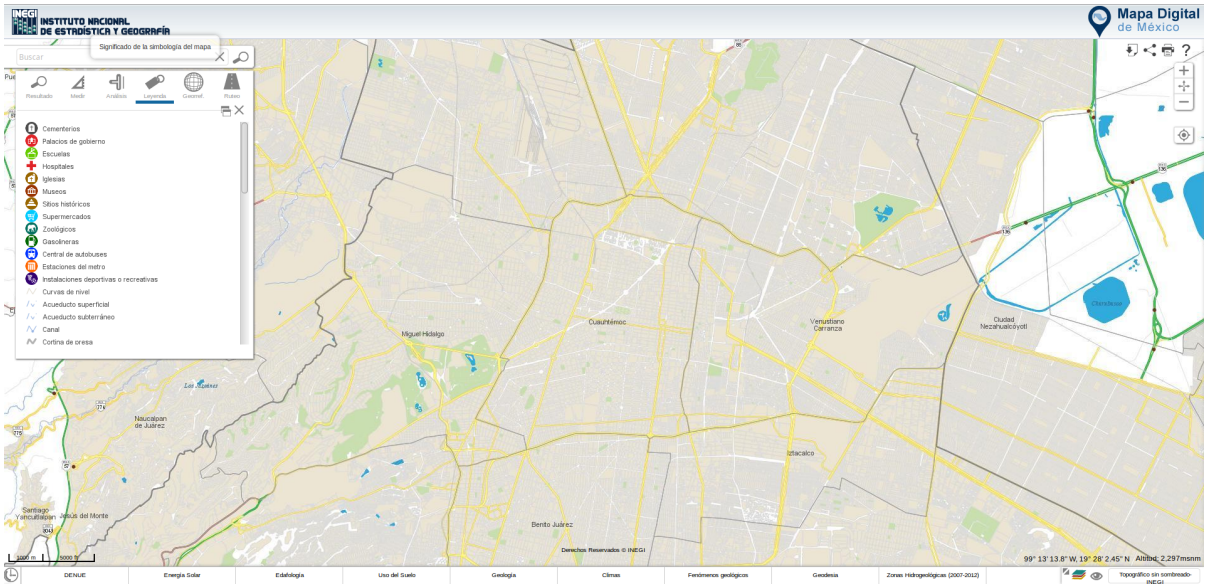


Figura 2.2: Mapa Digital - INEGI.

Toda la información que muestra el mapa digital del INEGI es obtenida por medio de servicios tipo REST, se puede hacer uso de éstos usando Web Scrapping, sin embargo, se encuentra totalmente ligado a la aplicación, trayendo consigo problemas en caso de que se deseara extraer el contenido que ofrecen.

El INEGI y el mapa digital cuentan con algunos puntos de acceso a la información usando como medio principal las peticiones HTTP. Estos servicios suelen ser aislados y requieren de una interacción directa con el humano, es decir, es necesario que éste tenga una interacción mediante escritura o clicks en los formularios de éstos.

2.2.4. Base de Datos Estadísticos BADESNIARN

Proyecto de la SEMARNAT que presenta información integrada, revisada y validada con cada una de las fuentes. Además esta estructurada para adecuarse a las necesidades de cada usuario . El usuario en su consulta encontrará el último dato revisado con la fuente, así como la serie histórica disponible en cada caso. Finalmente la plataforma tecnológica detrás le permitirá obtener un archivo electrónica en varios formatos con la info que muestra en pantalla.

Figura 2.3: Base de datos estadísticos - SEMARNAT.

Tiene un módulo de consultas temática en donde están muy bien delimitados los temas relacionados con el ambiente, y uno dinámica en donde se asocia cada metadato o archivo con palabras clave para que el usuario pueda encontrar el tema de su interés (véase Figura 2.3). [17]

2.2.5. Forecast IO

Es un atlas del clima que se puede visualizar a través de una página web. En el se pueden consultar datos climatológicos – tales como la temperatura actual, probabilidades de precipitación considrando como modo de visualización base mapas de alta definición (véase Figura 2.4).

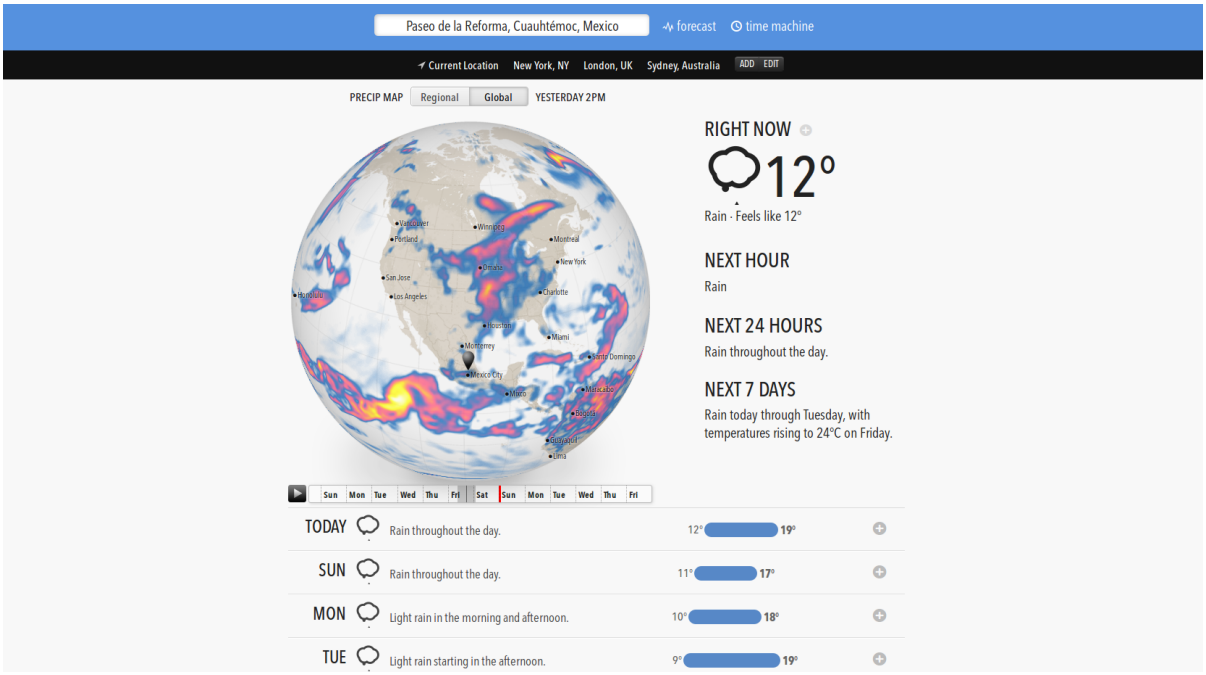


Figura 2.4: Forecast.io

Este proyecto integra la información de diversas fuentes y también provee un servicio para obtener datos a través de peticiones REST. Una de sus más grandes desventajas es la limitante de peticiones hacia su servicio, ya que después de cierta cantidad se aplican cargos a una tarjeta definida al dar de alta la cuenta de desarrollador en la plataforma. [14]

2.3. Sistemas de estandarización y unificación de datos.

2.3.1. Proyecto INSPIRE.

El Proyecto INSPIRE comenzó a ser desarrollado el 15 de Mayo 2007 y será completamente implementado para el año 2019. Su principal objetivo es es la creación de una infraestructura única de información geoespacial a lo largo de la unión europea (véase Figura 2.5).

La infraestructura de datos espaciales asistirá y trabajará a lo largo de todo el territorio Europeo y parte de Asia(Territorio de perteneciente a Rusia), los datos serán diversos considerando temás comunes y técnicos.

El proyecto INSPIRE trabaja bajo algunos principios, por ejemplo:

- Los datos deben ser recabados sólo una vez y mantenerse actualizados de forma efectiva.
- Es posible combinar información espacial de varias fuentes a través del territorio Europeo y compartir sus aplicaciones.
- La información geográfica almacenada en todos los niveles será transparente y compartida.
- Busqueda fácil de de información geográfica que puede ser usada para fines particulares y de uso general.

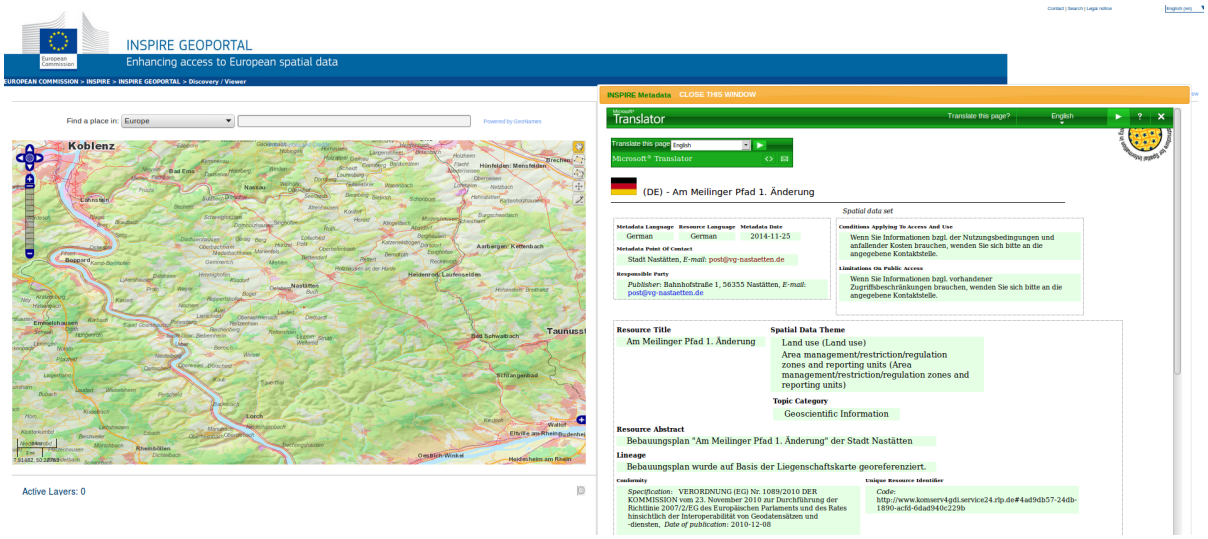


Figura 2.5: Visualización de datos y medatados en INSPIRE

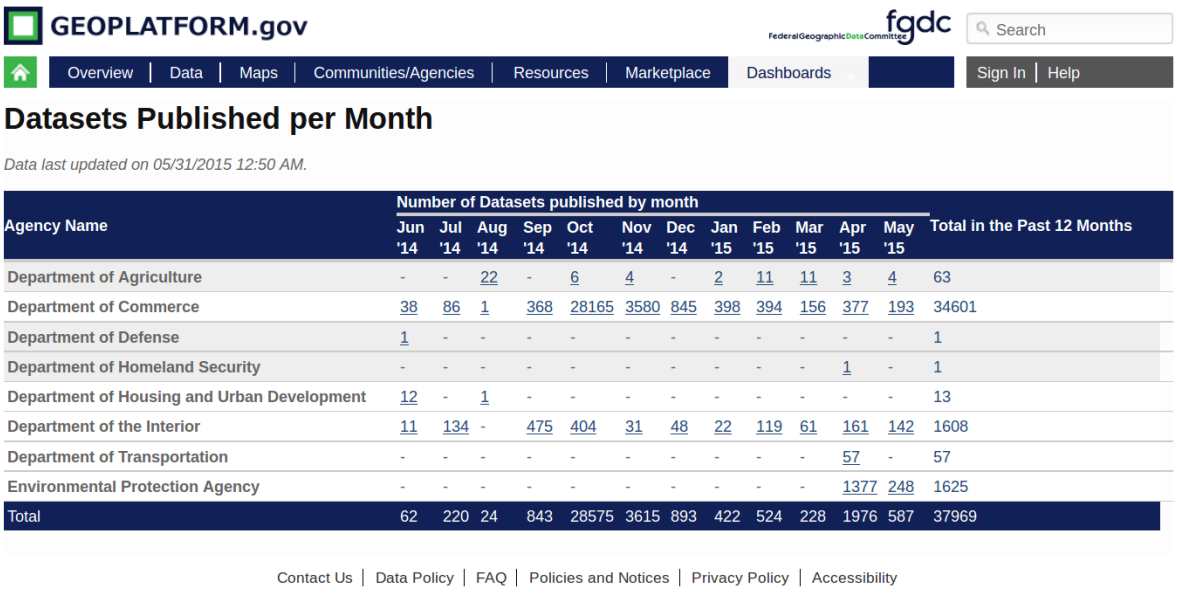
Actualmente parte de los modulos desarrollados se encuentran en etapas de pruebas. Cuenta con un editor de metadatos que consiedera la fecha, el nombre de la organización, correo electrónico, palabras clave, ubicación (Latitud, Longitud, Altitud), entre otros. Los metadatos pueden ser también validados utilizando las mismas herramientas que provee la plataforma.

El proyecto INSPIRE cuenta con un proceso de participación de "StakeHolders" para mejorar y analizar los detalles específicos del sistema INSPIRE y todos sus modulos.

2.3.2. Geoplatform

Plataforma propuesta por el gobierno de Estados Unidos, principalmente por el Comité Federal de Datos Geográficos (The Federal Geographic Data Committee) [19]. Actualmente funciona como una PaaS (Platform as a Service), cuyos principales objetivos son:

- Unificar y brindar información geoespacial confiable a través de datos y servicios.
- Soporte para toma de decisiones.
- Aplicación para la solución de problemas que pueden ser desarrollados sólo una vez y usados varias veces a través de distintas instituciones federales y otras organizaciones.
- Infraestructura compartida para almacenar datos y aplicaciones.
- Punto focal donde instituciones gubernamentales, académicas, privadas y públicas pueden visualizar información relativa a su región.



Agency Name	Number of Datasets published by month												Total in the Past 12 Months
	Jun '14	Jul '14	Aug '14	Sep '14	Oct '14	Nov '14	Dec '14	Jan '15	Feb '15	Mar '15	Apr '15	May '15	
Department of Agriculture	-	-	22	-	6	4	-	2	11	11	3	4	63
Department of Commerce	38	86	1	368	28165	3580	845	398	394	156	377	193	34601
Department of Defense	1	-	-	-	-	-	-	-	-	-	-	-	1
Department of Homeland Security	-	-	-	-	-	-	-	-	-	-	1	-	1
Department of Housing and Urban Development	12	-	1	-	-	-	-	-	-	-	-	-	13
Department of the Interior	11	134	-	475	404	31	48	22	119	61	161	142	1608
Department of Transportation	-	-	-	-	-	-	-	-	-	-	57	-	57
Environmental Protection Agency	-	-	-	-	-	-	-	-	-	-	1377	248	1625
Total	62	220	24	843	28575	3615	893	422	524	228	1976	587	37969

Figura 2.6: Datasets publicados por Geoplatform

Actualmente usan un estándar de datos abierto conocido como "Common Core Metadata Schema" bajo su versión 1. [20]

La implementación de esta plataforma fue motivada por los principios y espirito de "Open Government"[21], cuya principal visión es la de enfatizar la comunicación, contabilidad y transparencia entre gobierno-ciudadano, dentro del territorio estadounidense. Todos los datos, aplicaciones y servicios que provee Geoplatform se encuentran bajo licencias libres.

La plataforma fue desarrollada por miembros de la el comité federal de datos geográficos (FGDC por sus siglas en inglés) mediante la coolaboración de escuelas y usuarios expertos en el área. La audiencia que toma como fuente incluye agencias federales, estatales y locales, además de sector privado, educativo y al público en general (véase Figura 2.6). [19]

2.3.3. GeoMap México

GeoMap es una Plataforma Geoespacial Integral, que aprovecha las tecnologías informáticas actuales, implementando metodologías y estándares internacionales. Contiene todas las funcionalidades de un Sistema de Información Geográfica (SIG) y ofrece todos los servicios necesarios para la creación de aplicaciones geoespaciales (véase Figura 2.7). Entre sus principales funcionalidades se encuentran:

- Multiplataforma (Versión Escritorio, Web y para Dispositivos Móviles).
- Módulo de Mapas Dinámicos.
- Interacción con diferentes Servidores de Mapas.
- Conexión a distintas Bases de Datos Geoespaciales.
- Importación e Integración de información geográfica de diferentes fuentes y formatos.
- Visualización de Imágenes de Satélite, Ortofotos y de UAV (Vehículo Aéreo No Tripulado).
- Módulo de Administración y Autenticación de Usuarios.

La plataforma Informática responde a la necesidad cada vez más frecuente de utilizar mapas y servicios geográficos en aplicaciones Web y para la Nube. Incluye las funcionalidades de integración y administración de información cartográfica, visualización y procesamiento de información geográfica y tabular, conexión a bases de datos geoespaciales e implementación de análisis espacial avanzado. La plataforma se especializa en temas como la geocodificación y geolocalización, áreas de influencia, zonas de cobertura, ubicación de puntos de interés, visualización, búsqueda y análisis de datos geográficos.

2.4. Características climáticas y de contaminación en México.

2.4.1. ¿Cuándo se iniciaron los problemas de contaminación del aire?

Desde siempre la humanidad ha emitido contaminantes al aire, pero esto se incrementó de manera dramática a partir de la Revolución Industrial iniciada en el Reino Unido a finales del siglo XVII. En esa época, el trabajo manual fue remplazado por maquinaria,

Figura 2.7: Información publicada por GeoMap

básicamente por la introducción de tecnologías que empleaban el vapor y que hacían posible tener altos niveles de producción. El problema fue que con estos avances industriales se incrementó el uso de combustibles, tal como el petróleo y el carbón mineral, ambos indispensables para el funcionamiento de la nueva maquinaria.

Desde entonces el problema de contaminación del aire se ha convertido en una constante en muchas ciudades industriales de todo el mundo, lo que ha ocasionado problemas de salud a su población.

Algunos de los casos más dramáticos y graves son la famosa niebla tóxica londinense de 1952, el deterioro de los bosques europeos por la lluvia ácida en los años cincuenta y sesenta del siglo XX, y la grave situación de la calidad del aire en la Ciudad de México, Tokio y Sao Paulo durante las últimas décadas del siglo anterior.

2.4.2. ¿Cuáles son los contaminantes y qué efectos tienen?

Los contaminantes pueden ser emitidos de forma natural o por actividades relacionadas con el ser humano. Los fenómenos naturales que se producen en la superficie o en el interior de la tierra –como el caso de erupciones volcánicas, que produce emisiones de gases, vapores, polvos y aerosoloes-, también contribuyen a la contaminación del aire.

Los principales contaminantes relacionados con la calidad del aire son el bióxido de azufre (SO2), el monóxido de carbono (CO), los óxidos de nitrógeno (NOx), las partículas suspendidas, el plomo y el ozono (véase Figura 2.8).

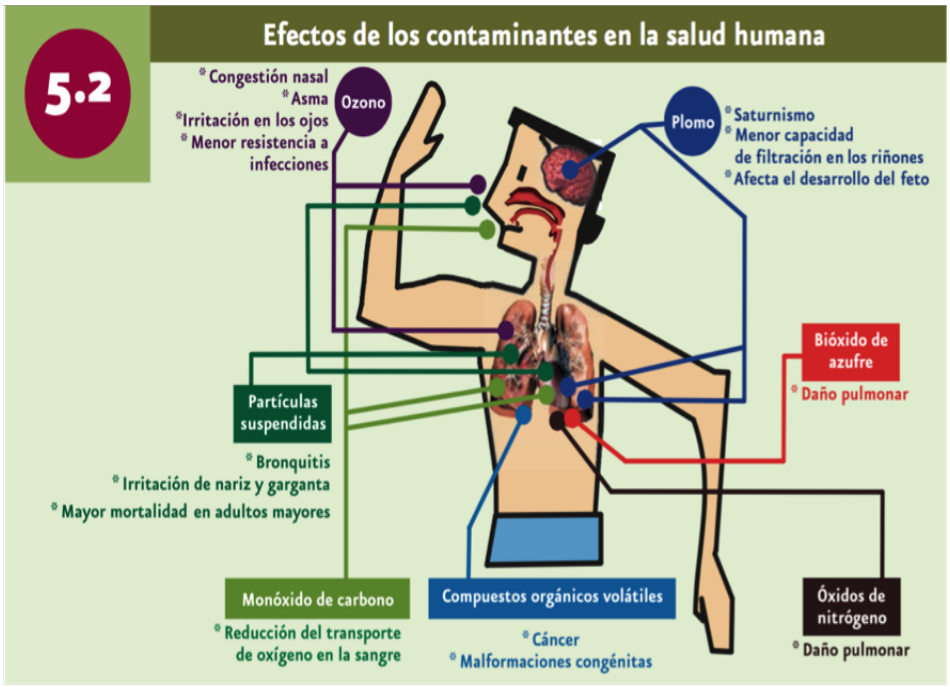


Figura 2.8: Efectos de los contaminantes en el cuerpo.

Las plantas, animales y otros organismos también resienten los efectos de contaminantes como el ozono. Principalmente con la formación de la lluvia ácida, dicha lluvia es ocasionada con la presencia de ciertos ácidos en la atomósfera que se precipitan a la tierra con la lluvia. El dióxido de azufre y los óxidos de nitrógeno, resultado de la quema de combustibles fósiles causan lluvia ácida, ya que al combinarse con agua, oxígeno y otros compuestos químicos forman ácidos como el ácido sulfúrico y el nítrico.

Las plantas se ven afectadas por esta lluvia ya que los ácidos pueden obstruir y acidificar los diminutos poros de las hojas por los que las plantas toman el aire que necesitan para realizar la fotosíntesis, además la lluvia ácida degrada los suelos, lo cual afecta las raíces y la nutrición de las plantas.

En el parque nacional Izta-Popo, Zoquiapan y en el Parque Nacional Desierto de los Leones, la lluvia ácida a dañado la vegetación. Estos daños involucran la pérdida de hojas y ramas, crecimiento lento y vulneravilidad a ataques de plagas y enfermedades (véase Figura 2.9).

Por otro lado los ríos, lagos y lagunas también pueden hacerse más ácidos por efecto de la lluvia ácida, lo cual pone en serio riesgo a las especies de plantas y animales que los habitan. Algunos ejemplos de estos daños se encuentran en los lagos del norte de Europa, en los que se ha reportado inclusio que han quedado sin ninguna forma de vida luego de la contaminación por lluvia ácida.

Figura 2.9: Impacto de la lluvia ácida en los bosques de México.

Figura 2.10: Daños en edificaciones por lluvia ácida.

Por otro lado también los monumentos y edificios sufren deterioros por la lluvia ácida, ya que los ácidos funcionan como agente corrosivo. El laboratorio de restauración del Instituto de Investigaciones Antropológicas de la UNAM indica que en losúltimos 25 años el deterioro de los monumentos y edificios históricos de la ciudad de México se ha acelerado de manera impresionante por el incremento de los niveles de contaminación (véase

Figura 2.10).

2.4.3. Quiénes generan los contaminantes atmosféricos?

En México al igual que en otros países se han desarrollado inventarios de emisiones que proporcionan información sobre la cantidad de contaminantes que se liberan al aire. En el año de 1999 de acuerdo al inventario de emisiones a nivel nacional se produjeron 40.5 millones de toneladas de las cuales el 58 % correspondieron a fuentes naturales- es decir, el suelo, la vegetación y las actividades volcánicas- y 42 % a la contaminación de origen humano.

A pesar de que aparentemente las fuentes naturales sean las mayores productoras de contaminación, son las fuentes antropogénicas las que están cerca de la población y las que influyen en mayor medida en la calidad del aire que se respira.

Dentro de las fuentes antropogénicas, los vehículos automotores son los mayores productores de contaminantes, después la quema de gas LP y al final las emisiones de plantas generadoras de electricidad.

2.4.4. ¿Qué hemos hecho para resolver el problema?

México lleva tiempo tomando acciones para resolver estos problemas, en 1988 implementó el Sistema Nacional del Inventario de Emisiones de Fuentes Fijas, así como un proyecto para cuantificar las emisiones del Valle de México. A partir del monitoreo de la calidad del aire ha diseñado algunas mejoras como eliminar el plomo en la gasolina, reducción del contenido de azufre.

Actualmente también existe una red de monitoreo atmosférico que abarca 52 ciudades y zonas metropolitanas que mide los niveles de contaminación presentes en el país. La concentración de los contaminantes en el aire se obtiene mediante la toma de muestras de aire que se analizan y procesan.

A partir de estas mediciones nacionales se han detectado cuales son las localidades con mayor índices de contaminación así como los contaminantes que emiten principalmente. Uno de los esfuerzos más notables son los programas para mejorar la calidad del aire (Proaires) que buscan revertir las tendencias de deterioro, ya que incoroporan medidas para el control y abatimiento de las emisiones de los contaminantes. También existen centrales eólicas para generar electricidad a partir de la energía del viento, una en la Venta, Oaxaca y la otra en Guerrero Negro, Baja California Sur.

Capítulo 3

Metodología

3.1. Desarrollo ágil de software

El desarrollo ágil propone una alternativa al desarrollo de software tradicional. Los enfoques del desarrollo ágil son principalmente usados en el desarrollo de software para ayudar a las compañias a responder fácilmente al cambio. [22]

3.1.1. Scrum

Scrum es un marco de gestión para el desarrollo incremental de un producto que proporciona una estructura de roles, reuniones, reglas y artefactos.

Scrum utiliza iteraciones de longitud fija denominados Sprints, que son típicamente de dos semanas o 30 días de duración. Los equipos de Scrum intentan generar un incremento de producto potencialmente entregable (debidamente probado) en cada iteración. [23]

3.1.2. Programación Extrema (XP)

Es un enfoque disciplinado para entregar software de alta calidad rápida y continuamente. Promueve una alta participación del cliente, retroalimentación rápida, pruebas continuas, planeación continua y entrega de software funcional en intervalos muy frecuentes que van de 1 a 3 semanas. [24]

3.2. Técnicas de desarrollo ágil

3.2.1. Desarrollo guiado por pruebas (Test Driven Development)

El desarrollo guiado por pruebas es una técnica avanzada qua hace uso de pruebas unitarias para guiar el diseño del software y forzar el desacoplamiento de dependencias.

El resultado de usar esta práctica es una comprensiva suite de pruebas que pueden ser ejecutadas en cualquier momento para proporcionar información de que el software está aún funcionando. [25]

3.2.2. Refactor

Es una técnica disciplinada para la reestructuración de un cuerpo de código existente, alterando su estructura interna sin cambiar su comportamiento exterior.

Cada transformación es pequeña, pero una secuencia de transformaciones pueden producir una importante reestructuración. [26]

3.2.3. Integración continua

Es una práctica de desarrollo que requiere a los desarrolladores la integración de código en un repositorio compartido varias veces al día.

Cada registro de entrada es entonces verificado por un proceso de construcción automatizado permitiendo a los equipos detectar problemas a tiempo.

La integración continua trae muchos beneficios, entre ellos la detección de errores rápidamente y reducir los problemas de integración lo que permite entrega de software más rápidamente. [27]

3.3. Justificación

Para el desarrollo del sistema pondrémos en práctica las técnicas mencionadas anteriormente que son características del marco de trabajo Scrum y de la metodología Extreme Programming.

El proceso de desarrollo del proyecto se llevará a cabo de la siguiente manera:

- Planteamiento de las historias de usuario.
- Defínción del plazo de tiempo para cada sprint y el conjunto de funcionalidades relacionadas a cada historia de usuario que serán desarrolladas dentro del mismo.
- Implementación de pruebas unitarias en conjunto con el desarrollo de cada módulo de la aplicación.
- Implementación de pruebas de integración para verificar la comunicación entre dos o más modulos.
- Despliegue de la aplicación en un ambiente productivo una vez que todas las pruebas hayan pasado.
- Implementación de pruebas funcionales para verificar que el flujo descrito en las historias de usuario se lleva a cabo correctamente.

3.4. Modelo por prototipos

Es un modelo para el desarrollo de sistemas en el cual un prototipo es construido, probado y finalmente reconstruido las veces que sea necesario hasta que un prototipo aceptable es finalmente alcanzado del cual el sistema completo o producto puede ser totalmente desarrollado.

Este modelo funciona bien en escenarios donde no se conocen por completo los requerimientos.

3.4.1. ¿Qué es un prototipo de software?

Es un modelo de software con una funcionalidad limitada que permite al usuario evaluar los propósitos del desarrollador y probarlos antes de su implementación.

También ayuda a entender los requerimientos específicos del usuario y que no pudieron haber sido considerados por el desarrollador durante el diseño del sistema.

3.4.2. Implementación del modelo de prototipos

- Los nuevos requerimientos del sistema se definen con el mayor detalle posible.
- Un diseño preeliminar es creado para el nuevo sistema.
- Un primer prototipo del nuevo sistema es construido para el diseño preeliminar que representa una aproximación a las características del producto final.
- El usuario evalua el primer prototipo, notando sus fortalezas y debilidades, qué necesita agregarse y que debería removerse. El usuario recoge las observaciones de los usuarios.
- El primer prototipo es modificado, basado en los comentarios hechos por el usuario, y un segundo prototipo del nuevo sistema es construido.
- El segundo prototipo es evaluado del mismo modo que el primero.
- Los pasos anteriores son iterados tantas veces como sea necesario, hasta que los usuarios consideren que el prototipo representa al producto final deseado.
- El sistema final es construido, basado en el prototipo final.
- El sistema final es evaluado y probado a fondo. Existe una rutina de mantenimiento continuo para prevenir errores a gran escala.

3.4.3. Integración del modelo orientado a prototipos, el framework Scrum y la metodología Extreme Programming

Al desarrollar el proyecto utilizando el modelo orientado a prototipos nos fue posible incorporar varias características del framework Scrum y la metodología XP.

Nos dimos cuenta de que el planteamiento de las historias de usuario al inicio del desarrollo de un prototipo nos permita describir rápidamente los requerimientos principales y definir criterios de aceptación para considerar cuando una funcionalidad o flujo de la aplicación estaba completo.

Otra de las característica que tomamos del framework Scrum fueron los Sprints; los cuales implementamos para enfocarnos en el desarrollo de cierta funcionalidad que agrupamos en un periodo de tiempo. La duración de cada Sprint fue de dos semanas.

De igual modo, hicimos uso de algunas técnicas características de la metodología XP como el desarrollo guiado por pruebas al construir nuestro primer prototipo; en el cual, primero se definen un conjunto de pruebas de unidad y en base a ellas se va desarrollando la funcionalidad para que las pruebas pasen sin problemas. Al hacer esto se asegura que la funcionalidad de un módulo pueda ser extendida fácilmente.

Capítulo 4

Tecnologías

Para el desarrollo del proyecto se decidió utilizar un lenguaje de programación dinámico que corre sobre la JVM (Groovy) y el conjunto de herramientas que existen en su ecosistema. Además, para la parte de la aplicación web será necesario integrar tecnologías JavaScript que nos ayuden en la interacción con la aplicación que expone datos para mostrar la información que se requiera.

A continuación se describen las ventajas de las tecnologías que se han decidido utilizar para el desarrollo del proyecto

Nombre	Ventajas
	▪ Implementa funciones espaciales para encontrar información relevante de ubicaciones específicas. ▪ Útil para el procesamiento de grandes cantidades de información.
	▪ Lenguaje expresivo que incrementa la productividad. Se reduce el azucar sintactico en comparación con lenguajes como C++ y Java y la curva de aprendizaje es muy pequeña si ya se conoce Java. ▪ Se pueden integrar fácilmente todas las bibliotecas de Java ya que corre sobre la JVM. ▪ Closures
	▪ Sigue un enfoque de construcción por convención. ▪ A diferencia de otras herramientas para la construcción de proyectos como Maven o Ant que utilizan XML, Gradle hace uso de un poderoso lenguaje específico de dominio (Groovy) para la definición de las tareas que deben ejecutarse. ▪ Cuenta con un administrador de dependencias que se encarga de descargarlas y las deja disponibles para su uso en la aplicación.
	▪ Genera la estructura lógica para aplicar el patrón de arquitectura Model Vista Controlador, ya que se encuentra orientado a microservicios usando el framework Spring MVC. ▪ Buen soporte de pruebas unitarias, de integración y funcionales. ▪ Capa de Mapeo Objeto Relacional que trabaja sobre Hibernate para las operaciones transaccionales.
	▪ Gran escalabilidad ▪ Canal distribuido de mensajes como medio de comunicación. ▪ Herramienta políglota.

	▪ Permite la automatización y ejecución de tareas para la construcción de un proyecto JavaScript.
 YEOMAN	▪ Define la estructura de directorios y archivos para un proyecto JavaScript. ▪ Cuenta con varios generadores para crear el código repetitivo que se necesita para iniciar un proyecto. ▪ Define tareas para que el desarrollador se concentre sólo en realizar la funcionalidad de la aplicación.

Cuadro 4.1: Descripción de las tecnologías

Capítulo 5

Modelo de Datos

5.1. Justificación

Se hará uso de un modelo de base de datos orientado a documentos. También se contará con un esquema no relacional, es decir, se carece de una normalización definida, haciendo uso de MongoDB, que se encargará de persistir y manejar las estructuras de datos en documentos JSON. Esta base de datos pertenece a la categoria NoSQL.

Las características de una base de datos NoSQL son las siguientes:

- Modelo de datos flexible.
- Buen rendimiento en clusters.
- Sin esquemas.

Este tipo de bases resultan útiles debido a que se se pretende procesar grandes volúmenes de información para mostrar un historial de la información climática que se vaya persistiendo.

Las tecnologías para grandes volumenes de información son relativamente nuevas; estas surgieron debido a la necesidad que tenían empresas grandes como Google y Amazon.

La información fue convertida de un modelo relacional a un modelo basado en documentos, jerárquico o basado en columnas usando proceso de denormalización, ésto con la finalidad de dar un orden a la información considerando simplemente consultas de cierto tipo evitando así operaciones típicas del álgebra relacional cómo el Producto Cartesiano, que implicaba el uso de grandes recursos.

Las tecnologías de tipo NoSQL se orientan a consultas y no a transacción. Lo que brinda un fácil acceso a la información, una estructura de los datos orientada a su posterior análisis, respuestas en tiempo real y una gran capacidad de escalar y replicar. [10]

Empresas cómo Foursquare, Google, Amazon, Uber o Twitter, han optado por complementar la información que persisten de forma relacional con modelos orientados a documentos usando tecnologías como Hbase, MongoDB, BigTable, DynamoDB, por citar algunos ejemplos. [11] [12]

5.2. Descripción

Considerando la problematica y solución que plantea el equipo de Ambienta2MX, se ha optado por orientar el sistema a consultas usando documentos como modelo de datos base.

La información que será guardada y posteriormente consultada por otros módulos del ecosistema Ambienta2MX seguirá un esquema propuesto por el equipo de trabajo, éste recopila la estructura de sistemas que proveen información climática cómo lo son weather.gov, forecast.io, Weather Underground, proyecto INSPIRE (Unión Europea), Servicio Meteorológico Nacional, entre otros. [13] [14] [15]

Para la persistencia de la información se han definido los siguientes documentos:

- Places
- Pollution
- Weather

Los documentos serán almacenados en MongoDB considerando los datos definidos en la especificación JSON Data Interchange Format (en su versión 2013). [16]

El equipo de Ambienta2MX ha considerado el uso de el formato antes mencionado debido a su facilidad de lectura e integración con otro tipo de plataformas y lenguajes de programación, actualmente es el formato que rige el manejo de API de tipo REST.

Dentro de los modelos existe información relativa a los proveedores o fuentes de información, ésta información será utilizada para fines de consulta y contar con el control del origen de los datos.

5.2.1. Places

```
{
  "location": { // Using GeoJson Spec
    "type": "Point",
    "coordinates": [Number]
  },
  "sexagesimal_coordinates": [Number], // Original coordinates from
     INEGI source
  "itrf_coordinates": [Number], // Converted coordinates to itrf2008
     format.
  "nad27_coordinates": [Number], // Previous coordinates
  "height": Number, // height of the place
  "town": String, // name of the town
  "state": String, // name of the state
  "city": String, // name of the city
  "fullName": String, // Place full name,(Search)
  "zipCode": String, // Optional information
  "extraInfo": [String], // Extra information
  "provider": [Object], // Information about the providers (INEGI,
     Google, Geohack, etc) as provider:{information:content,information
     :url, ...}
  "lastUpdated": Date, // Grails generated information
  "dateCreated": Date, // Grails generated information
}
```

5.2.2. Pollution

```
{
  "location": { // Using GeoJson Spec
    "type": "Point",
    "coordinates": [Number],
  },
  "height": Number, // height of the place
  "airQuality": Number, // Air Quality (Percentage)
  "ozone": Number, // Ozone index (Percentage)
  "sulphurDioxide": Number, // Sulphur Dioxide index (Percentage)
  "nitrogenDioxide": Number, // Nitrogen Dioxide index (Percentage)
  "carbonMonoxide": Number, // Carbon Monoxide index (Percentage)
  "UV": Number, // Ultraviolet index
  "provider": [Object], // Providers list as provider:{information:
     content,information:url, ...}
  "sampleDate" : Date, // Sample date
  "lastUpdated": Date, // Grails generated information
  "dateCreated": Date, // Grails generated information
  "fullName": String // For text searching purposes
}
```

5.2.3. Weather

```
{
  "location":{ // Using GeoJson Spec
    "type":"Point",
    "coordinates":[Number],
  },
  "height":Number, // height of the place
  "description": String, // Description about the weather
  "rainfallIntensity": Number, // Rain Intensity (Percentage)
  "rainfallProbability": Number, // Rain probability (Percentage)
  "temperature": Number, // Celsius (desired)
  "apparentTemperature": Number, // Apparent Temperature (Some sources
     could provide this information)
  "dewPoint": Number, // Dew Point
  "humidity": Number, // Humidity index (Percentage)
  "windSpeed": Number, // Wind speed (Km/h)
  "windDirection": Number, // Wind direction, considering eight
     cardinal points
  "windBearing": Number, // Wind direction (Consdiering 360 degrees)
  "visibility": Number, // Visibility (Percentage)
  "cloudCover": Number, // Clouds on the sky (Percentage)
  "pressure": Number, // Pressure (mmHg)
  "provider": [Object],  // Providers list as provider:{information:
     content,information:url, ...}
  "sampleDate": Date,  // Sample date
  "lastUpDated": Date, // Extra information
  "dateCreated": Date, // Extra information
  "fullName": String // For text searching purposes
}
```

Capítulo 6

Definición temática de Ambienta2MX

6.1. Objetivo general

Considerando el problema de información y estructura de datos que tiene México en la actualidad, además de seguir la tendencia y aprovechar la brecha que ha disminuido el uso de datos abiertos, el equipo de Ambienta2MX decidió afrontar la tarea de desarrollar una herramienta que permita la conceptualización de la información de variables ambientales e indices de calidad y contaminación del aire que el INEGI y otras instituciones públicas o privadas almacenan y/o exponen para fines educativos, informativos o de uso partícular.

6.2. Justificación

Existe la necesidad de una fuente de información que presente datos climáticos y de contaminación del aire que ofrecen diversas instituciones como el Instituto Nacional de Estadística y Geografía (INEGI), el Servicio Meteorólogico Nacional y la Comisión Nacional del Agua de manera estándarizada. Esto con la finalidad de ordenar y categorizar un conjunto de metadatos.

La aplicación del esqueleto de Ambienta2MX, puede ser tomada como base para la expansión hacia otro tipo de variables y servicios, quedando a disposición de los futuros interesados analizar los datos que se almacenarán en las bases de datos de tipo MX (mencionadas más adelante) además de llevar a cabo el proceso de búsqueda y obtención de los datos deseados por éstos.

6.2.1. Alcances en Trabajo Terminal 2

Considerando la metodología de trabajo, además del alcance del proyecto, se decidió delimitar la información climatológica y de variables de contaminación al Distrito Federal cómo caso de estudio, principalmente contando con datos de la zona norte de la capital del país.

Como muestra visual del caso de estudio, se proveerán mapas que demostrarán la aplicación de los datos estandarizados y obtenidos de diversas fuentes, cabe destacar, que la información puede tener diversas interpretaciones dependiendo el uso que se le vaya a dar, queda a disposición del usuario final la interacción y aplicación del contenido climatológico recopilado por la plataforma Ambienta2MX.

Capítulo 7

Descripción de Ambienta2MX

7.1. ¿Qué y para qué es Ambienta2MX?

Ambienta2MX es el nombre de la plataforma que pretende formar parte de una macro solución orientada a la estandarización de datos geoespaciales que el INEGI y otras instituciones públicas tienen en su haber.

Actualmente no existe un estándar de datos geográficos a nivel nacional. Han existido aproximaciones mediante concursos que instituciones públicas como el INEGI ha publicado, o simplemente han existido propuestas que han brindado una solución incompleta a la unión y manejo de información geográfica, geodésica, hidrográfica, climática, topográfica, etc.

Ambienta2MX toma parte de todo el problema y propone una infraestructura lógica para afrontar la estandarización de variables ambientales y algunos índices de contaminación. Esta información actualmente se encuentra en formatos muy rudimentarios como textos planos sin algún protocolo o definición para su interpretación.

Sistemas semejantes, por ejemplo, el Servicio Meteorológico Nacional carece de algún recurso del cual se puedan realizar consultas que no sea mediante su portal web, esto trae problemas directos de compatibilidad con otros sitemas.

Un caso semejante se tiene con la información que la Conagua maneja en sus centrales meteorológicas a lo largo del país, los datos que brindan se actualizan de forma periodica y el único medio de acceso es a través de una página de internet que devuelve archivos en formato de texto u hojas de cálculo.

Los impedimentos antes mencionados conllevan a situaciones tan triviales como la consulta de datos para algúna región o punto específico del territorio nacional, al existir diversas fuentes no es posible tener un compendio del cual tomar la información que más sea conveniente.

Si a este problema se le añade que los datos carecen de un estandar, se puede visualizar el punto en que intentar manipular o tratar los datos se vuelve una tarea complicada y en exceso tediosa.

Considerando dichos problemas Ambienta2MX, propone un estandar de datos climáticos tomando como referencia diversas fuentes y adaptando los tipos de datos a tecnologías y tendencias actuales, brindando así una mayor portabilidad y simplicidad en la consulta de información.

7.2. Diagrama de Ambienta2MX

Ambienta2MX constará de varios módulos que trabajarán de forma conjunta para satisfacer la necesidad de tener un estandar y un repositorio de datos climáticos a nivel nacional.

Al brindar un sistema modularizado, se genera de forma directa un impacto en el proceso de análisis, desarolló e intengración. Éste tipo de modelo describe de una forma sencilla los componentes necesarios para solventar la demanda a la que se encontrará sometida la plataforma.

A continuación se muestra el diagrama a bloques de Ambienta2MX (véase Figura 7.1), todos los módulos, recursos y bases de datos serán descritos de forma posterior.

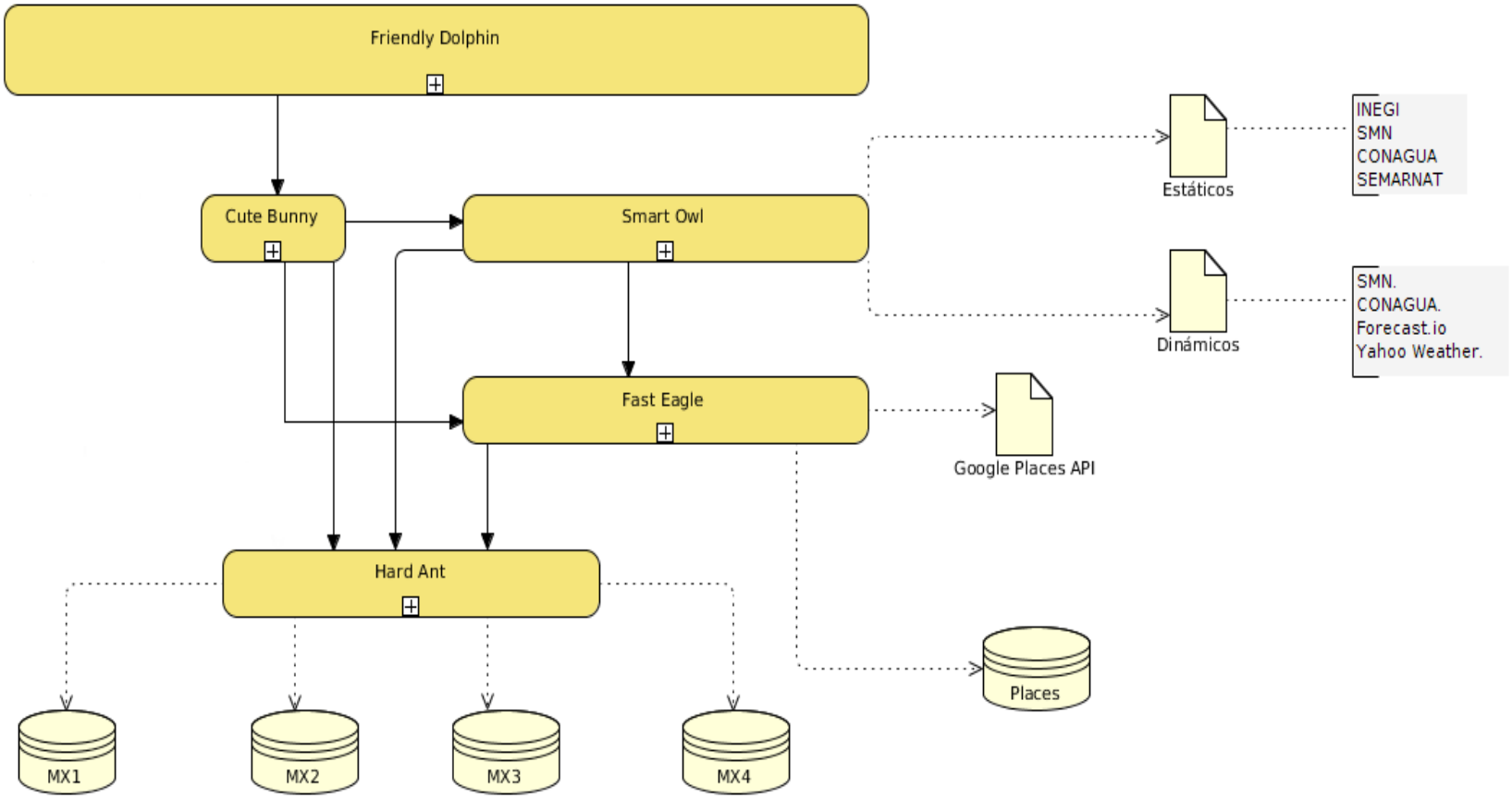


Figura 7.1: Módulos y estructura de Ambienta2MX

Cómo se apreciar en el diagrama, además de los gestores de bases de datos, Ambienta2MX se encuentra dividido en siete módulos básicos:

- Friendy Dolphin.
- Cute Bunny.
- Smart Owl.
- Fast Eagle.
- Hard Ant.

En el mismo diagrama se pueden observar las fuentes que proporcionarán la información ya sea a un nivel estático, por ejemplo, carta climática anual de algún municipio del territorio nacional; o bien, recursos que se actualizan de forma periodica como son los datos que provee el Servicio Meteorológico Nacional.

Se condieran cinco bases de datos, *MX1, MX2, MX3, MX4, Places*. Todas las bases del tipo MX contarán con la información de variables ambientales así también de los índices de contaminación de las zonas que conforman al territorio nacional.

Para el caso de *Places*, la base será usada como un macro índice cartográfico del territorio nacional, es decir, esta base será la referencia a nivel latitud, longitud y altitud para ubicar los datos que requieran ser procesados.

Todas las bases se encontrarán funcionando bajo un modelo de base de datos documental teniendo una alimentación bajo demanda, es decir, el contenido gestionado irá aumentando conforme las éstos vayan siendo solicitados.

Capítulo 8

Módulos de Ambienta2MX

8.1. Friendly Dolphin

8.1.1. Definición y objetivos

Éste módulo es el encargado de brindar la información procesada al usuario a través de una página de internet (véase Figura 8.1). Es el módo visual que los usuarios finales tendrán para poder interactuar con el ecosistema Ambienta2MX.

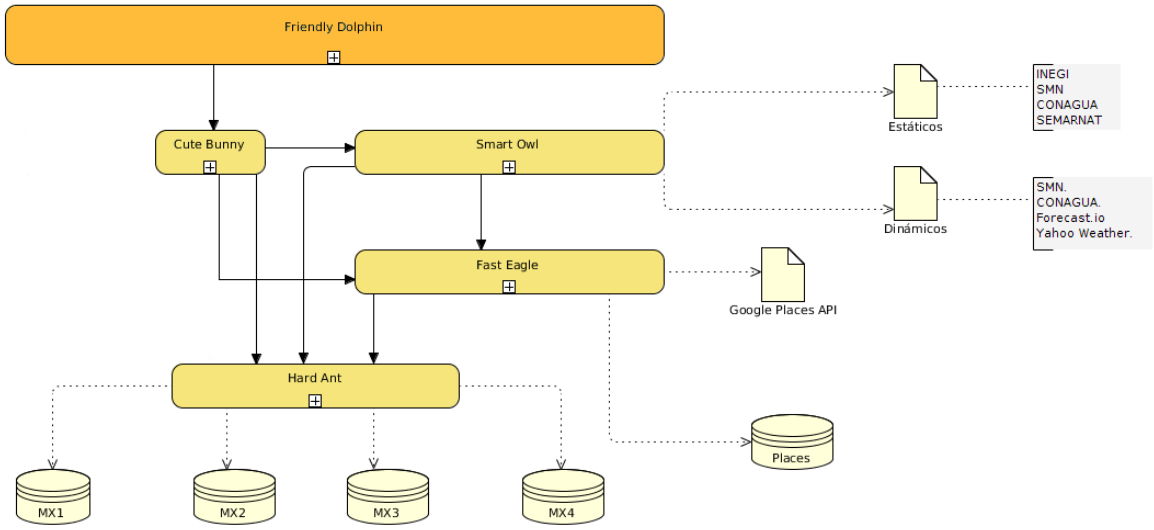


Figura 8.1: Friendly Dolphin, Módulo de Ambienta2MX.

Se presenta cómo un módulo web que consumirá la información procesada y almacenada por las cuatro bases (MX1,MX2,MX3,MX4) y la base de soporte (Places).

La principal función es la de consulta y visualización de datos. Es la capa más expuesta y visual de Ambienta2MX ya que es la que tendrá interacción directa con usuarios no técnicos, sin embargo, contará con los procesos necesarios para poder extraer información de las demás plataformas en formatos convencionales cómo JSON o CSV para uso posterior del usuario.

8.1.2. Alcances

Interactúa de forma directa con el bloque *Cute Bunny*, que forma parte de la segunda capa de exposición de datos de Ambienta2MX. Se comunica con los demás módulos mediante servicios de tipo REST que funcionan bajo el patrón de convención sobre configuración[8], brindando así una gran compatibilidad con éstos además de disminuir el tiempo de desarrollo debido a que no es necesario generar código único y se opta por la reutilización de éste además de apoyarse con el uso de bibliotecas que siguen el mismo método de trabajo.

Friendly Dolphin sólo puede ser visto como una herramienta de consulta, no podrá ser visto cómo un utensilio de análisis de datos climatológicos, es por ello que muestra la información en mapas, gráficas y detalles de la consulta.

8.1.3. Restricciones

Éste módulo se ve limitado por la API de Google Maps para la visualización de la información, ya que las consultas resultan limitadas en su versión gratuita, sin embargo, la cantidad es suficiente para demostrar la aplicación de los datos en una herramienta de visualización. Las restricciones estan definidas a 25,000 solicitudes por día y limitada a un segundo por petición o usuario.

La vista también cuenta con cierta limitantes, sólo podrá ser visualizada en navegadores con Internet Explorer 11+, Mozilla Firefox 20+ y Google Chrome 20+

8.1.4. Arquitectura

Friendly Dolpin contará con varios procesos y módulos a ser desarrollados (véase Figura 8.2). Éste módulo se desarrollará usando tecnologías cómo HTML, Javascript y CSS, además de contar con un ciclo continuo de desarrollo usando herramientas de apoyo cómo Yeoman, Gulp para el maquetado y gestión de tareas comunes en projectos de tipo web.

Se hará uso del servidor interno que ofrece Gulp junto con las tareas y gestión de bibliotecas de terceros. En cuanto al desarrollo de los estilos necesarios para las vistas se implementará Bootstrap cómo maquetado CSS y finalmente el manejo de vistas, peticiciones y lógica dentro del navegador de los clientes se implementará un patrón de tipo SPA (Single Page Application)[36] desarrollado por el equipo de trabajo.

8.1.5. Factibilidad

Se decidió cambiar de framework en las vistas, se quitó la implementación de EmberJs del proyecto y se optó por simplemente la ideología que éste tiene, considerando un patrón SPA[36] mínimo ya que el framework antes mencionado contaba con demasiadas características que no iban a ser implementadas sin embargo el framework hace uso de estas.

Se optó por seguir esa ideología brindando la flexibilidad necesaria que consideró el equipo de trabajo para cumplir con el objetivo de tener una página dinámica que convive con los demás módulos de Ambienta2MX.

A continuación se muestran las características principales de cada tecnología:

“EmberJs”

- Manajo de un patrón SPA.
- Trabajo y desarrollo utilizando Convenció sobre configuración.
- Control de rutas, controladores, modelos, vistas, pruebas y dependencias externas.
- Implementación de plantillas (templates).
- Uso de AJAX como forma de interacción con servicios externos.

“Servicio Generado por el Equipo de Ambienta2MX”

- Uso de AJAX como forma de interacción con servicios externos.
- Manejo básico de templates para vistas.
- Control de rutas y datos adquiridos del servidor.

Cómo puede verse, el framework EmberJs contiene más caracteristicas que las necesarias para el desarrollo del proyecto, es por eso que decidió adaptarse esa misma ideología haciendo uso de un framework mínimo creado por el equipo de Ambienta2MX totalmente adecuado a las necesidades que tiene en éste caso el módulo Friendly Dolphin.

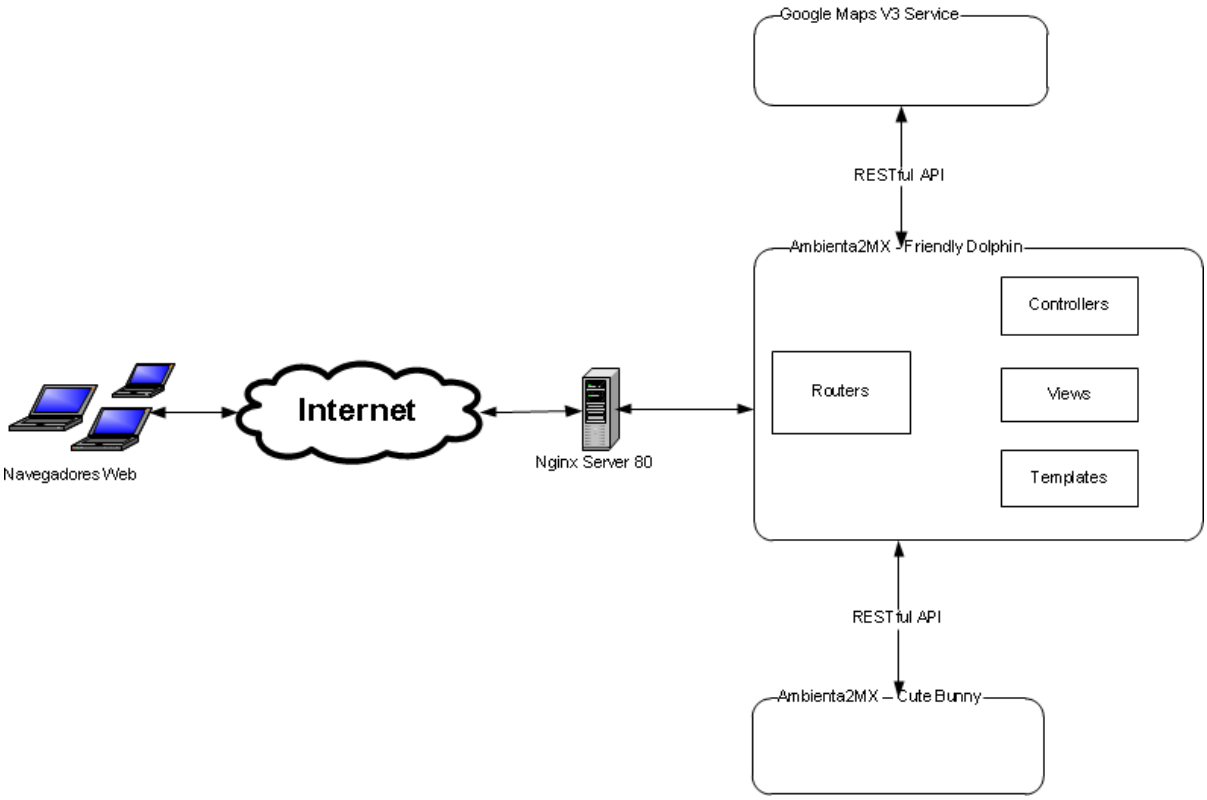


Figura 8.2: Diagrama General de Friendly Dolphin

8.1.6. Pruebas y Capturas de pantalla

Para este módulo se realizaron pruebas funcionales indicando el flujo básico de información que un usuario común seguiría. Las pantallas básicas se muestran a continuación y la información de las pruebas pueden ser visualizadas en el anexo en este documento.

Cómo se puede visualizar en la imagen anterior los datos de clima de ciertas regiones del país, en este caso en el área de Baja California, utilizando un concepto llamado HeatMap (Mapa de Calor) que nos permite la transposición de un mapa de Google y los datos climatólogicos recolectados por el sistema de Ambienta2MX. Ésta vista puede ser vista por medio del formulario de busqueda ubicado en la parte superior de las vistas que fueron generadas (véase Figura 8.4).

También puede ser visualizada la información en forma de gráfica de barras, considerando en este caso el área de estudio que es el Distrito Federal, principalmente la parte norte de la capital.

El mapa de calor (HeatMap) se traslapa con la vista satelital que nos brinda el servicio de Google Maps. En esta captura se puede apreciar de mejor manera el espectro de temperatura en el área en cuestión, mostrando de un color rojo las áreas con el más alto índice de temperatura y disminuyendo a un color verde en las zonas donde la sensación termica fue menor.

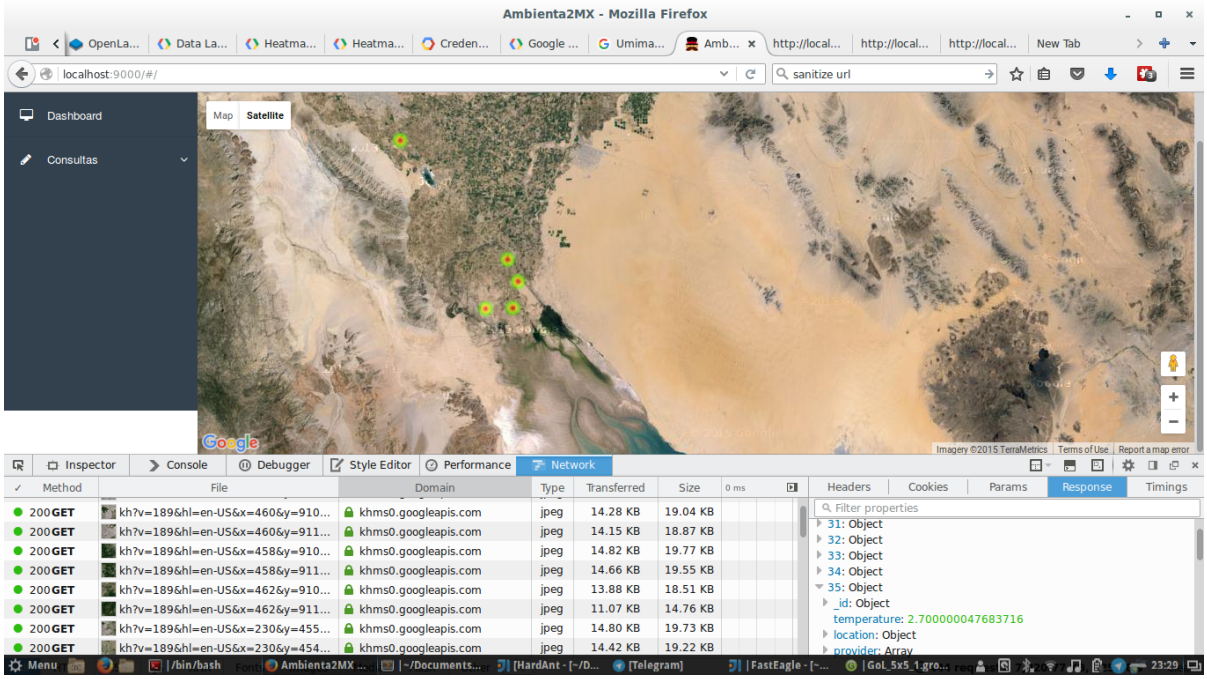


Figura 8.3: Mapas de Calor, Friendly Dolphin

Cómo puede verse en la siguiente imagen, también se cuenta con una consulta que toma como base un "Marker" cólocado por el usuario e índicando el radio de búsqueda en metros. Con ello, el servicio recorrerá las bases de tipo MX para poder encontrar los

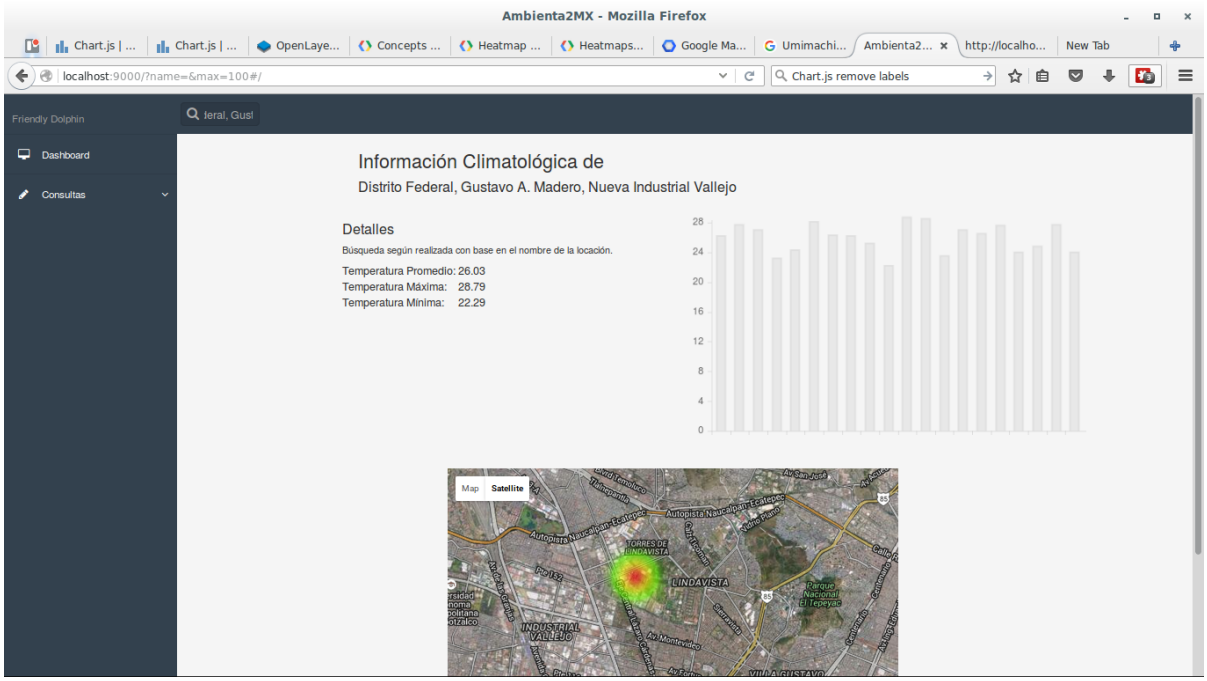


Figura 8.4: Tablas e información de clima, Friendly Dolphin

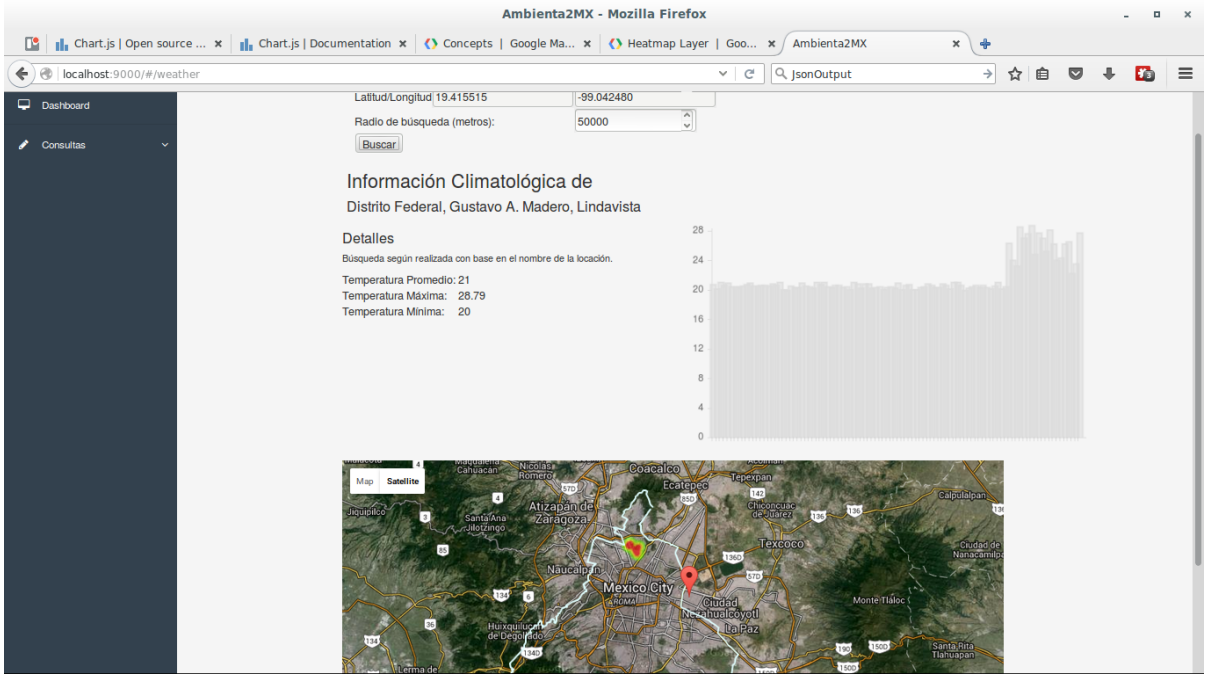


Figura 8.5: Tablas e información de clima, Friendly Dolphin

datos climatológicos o bien de contaminación que cumplen con el criterio de proximidad tomando la información Latitud/Longitud existente en el mapa.

Éste tipo de búsqueda resulta util cuando se desea analizar los datos de un área delimitada por un círculo. El ejemplo muestra los datos a 50 kilométros de Ciudad Nezahualcóyotl (Estado de México), coincidiendo con datos de prueba (ubicados en la delegación Gustavo A. Madero), demostrando la funcionalidad de la búsqueda.

La información que hace referencia a variables de contaminación ambiental cuenta con pantallas análogas a las mostradas.

Las pruebas funcionales fueron ejecutadas utilizando el framework iMacros, a continuación se muestra la rutina que indica la consulta de un lugar y se muestra en la pantalla.

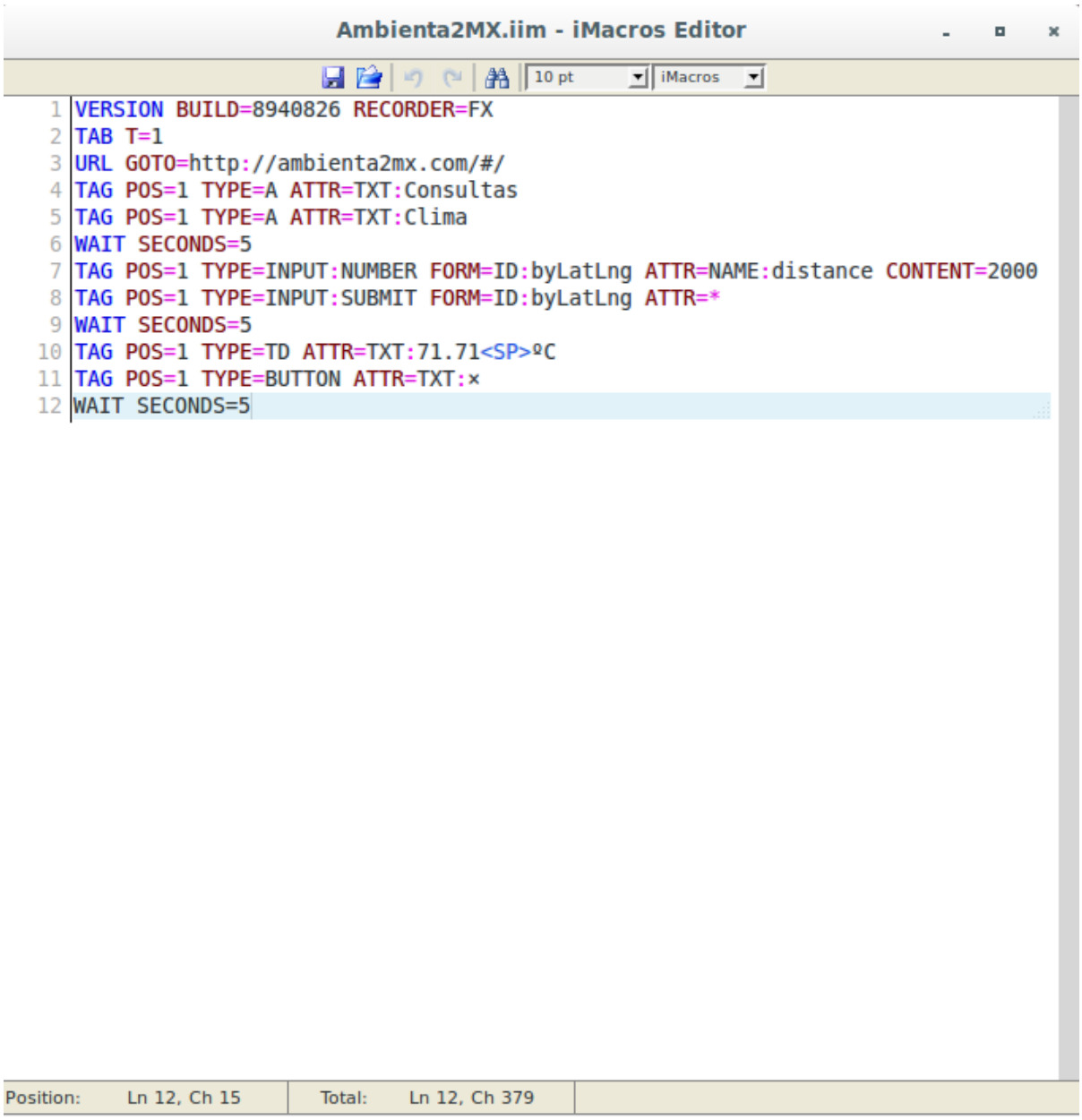


Figura 8.6: Código de prueba funcional

8.2. Cute Bunny

8.2.1. Definición y objetivos

El módulo *Cute Bunny* es parte medular para la comunicación con otros sistemas. Cute Bunny es el encargado de proporcionar los respectivos servicios de tipo REST a otros sistemas o módulos que deseen consultar la información almacenada en las respectivas bases de tipo MX (véase Figura 8.5).

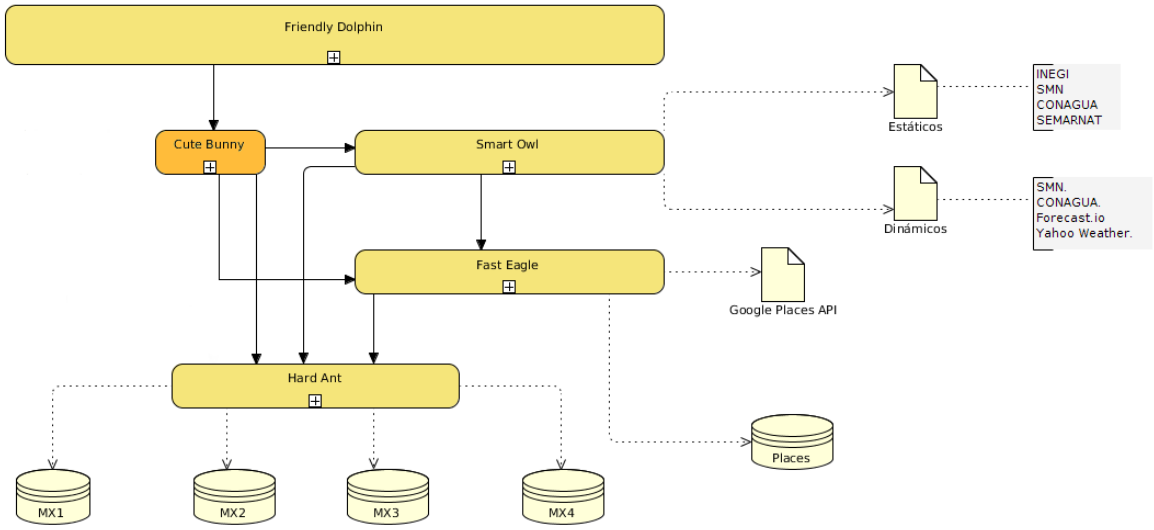


Figura 8.7: Cute Bunny, Módulo de Ambienta2MX.

Cute Bunny tiene como objetivo unir y establecer la comunicación inicial entre los módulos de Ambienta2MX y la vista del usuario final, proporcionada por el módulo Friendly Dolphin. A continuación se muestra su ubicación con el diagrama general de Ambienta2MX.

Cute Bunny interactúa con todos módulos de Ambienta2MX, Smart Owl, Hard Ant y Fast Eagle. La interacción con estos surge debido a que Hard Ant es el encargado de gestionar el acceso a las bases de datos de tipo MX y Smart Owl brindará y dará solución a las busquedas que no se encuentren en las bases de tipo MX, es decir, tratará de encontrar la información que Cute Bunny le solicitó para guardarla en algunas de las bases y posteriormente regresar el resultado al solicitante, en el caso de Fast Eagle, para la búsqueda de locaciones en la base de datos Places.

8.2.2. Alcances

Considerando el objetivo primordial, qué es el de comunicar y unir, su único alcance es el de mantener la conexión de forma transparente entre los módulos y permitir la

interacción como si todos se encontraran dentro de una misma máquina o servicio único; esto con la finalidad de brindar el acceso a los datos mediante una API de tipo REST libre del contexto y alcance de cada módulo.

Considerando la aplicación de un servidor como Nginx, se puede permitir expandir la funcionalidad sin mostrar al usuario la existencia de diversos sistemas trabajando bajo la misma dirección, es una de sus más grandes bondades.

8.2.3. Restricciones

Una de las principales limitantes de éste módulo se ve reflejada en el negocio, debido a que carece de una capa lógica o de procesos entonces se terminó por delegar la información y la lógica a los otros proyectos involucrados.

Para éste módulo se decidió hacer uso de un servicio tipo proxy que sólo se encarga de mantener la transparencia como una API única, carece totalmente de un sentido en el negocio y puede ser cambiado por algún otro tipo de servidor web.

8.2.4. Arquitectura

A continuación se mostrará el diagrama por bloques que define la estructura de Cute Bunny.

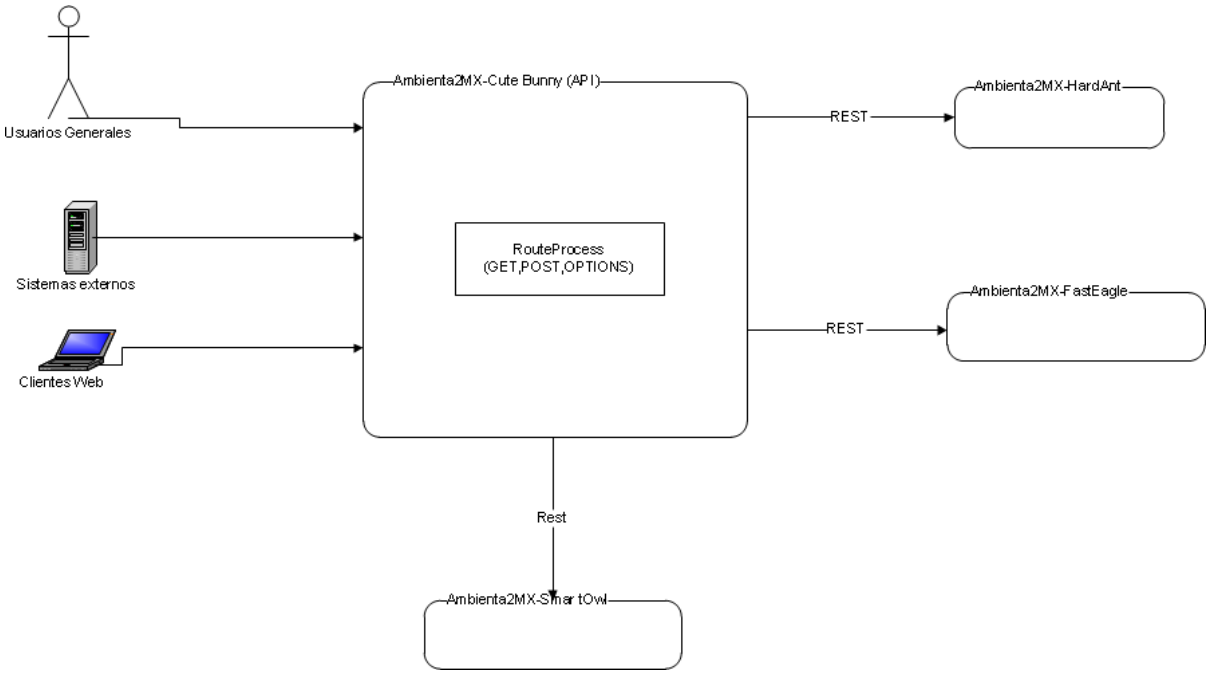


Figura 8.8: Diagrama General de Cute Bunny

Como puede ser visualizado, funge como el pegamento de todos los módulos para que éstos puedan ser accedidos desde la vista (Friendly Dolphin), o bien, que servicios de externos puedan alimentarse de la información climatológica obtenida por el sistema Ambienta2MX (véase Figura 8.6).

8.2.5. Factibilidad

Originalmente se tenía pensado realizar la implementación y unión de todos los módulos del negocio de Ambienta2MX utilizando la tecnología Grails, sin embargo, se dió un cambio drástico debido a que el framework realmente no sería utilizado al 100 %, lo cual nos permitió visualizar el problema desde un punto más de DevOps [38], optando por la aplicación del servidor nginx como pieza para la unión del sistema.

A continuación se muestran las características principales de cada tecnología:

"Grails"

- Web Framework basado en un patrón MVC.
- Trabajo bajo una convención sobre configuración.
- Funcionamiento bajo la máquina virtual de Java (JVM)

"Nginx"

- Servidor HTTP orientado a microservicios.
- Escrito en C, soporta caché de peticiones y balance de carga.
- Soporte para la actualización al protocolo HTTP v2.

Cómo puede visualizarse, no existe punto de comparación ente ambas tecnologías, este cambio se realizó debido al enfoque se tenía con la aplicación y definición del módulo Cute Bunny desde una instancia inicial. "Grails" se encuentra definido como parte de un servidor de aplicación (Tomcat o Jetty), mientras Nginx es un Web Server (Semejante a Apache2).

La información del servidor puede ser visualizada con más detalle en el Anexo 6: Instalación y configuració de Nginx.

8.3. Smart Owl

8.3.1. Definición

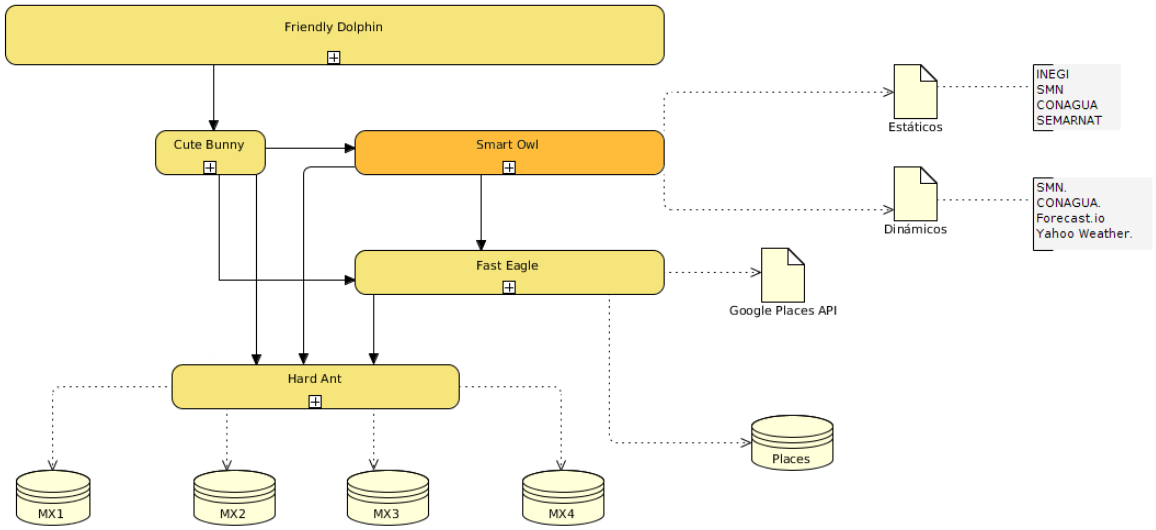


Figura 8.9: Smart Owl, Modulo de Ambienta2MX.

La función principal del Smart Owl es obtener información de diversas fuentes que proveen datos climáticos y de contaminación y exponer esta información de manera estandarizada con una estructura definida en formato JSON (véase Figura 8.7).

Todas la información que se encontrará en las bases de datos de tipo MX será obtenida a través de Smart Owl, muchas de las fuentes no cuentan con los datos climatológicos completos, principalmente las gubernamentales como: CONAGUA, INEGI o SMN, por citar algunas.

Considerando esa problematica, Smart Owl busca y trata de resolver la información de los campos faltantes tomando como base distintas fuentes de datos, algunas establecidas y otras de tipo gubernamental.

También se tomará información de fuentes que tipo dinámica,es decir, cuya información suele ser actualizada en entre periodos de una o dos horas. Estas fuentes suelen contar con RESTFul API's para consumo de forma programática.

El modelo de datos final podrá ser entonces persistido después de que haya sido resuelto completa o parcialmente la petición.

Se llevará el control de los metadatos considerando su origen, su fecha y algunos tags relacionados los que proveen la información.

8.3.2. Alcances

Una vez que se definieron las fuentes principales para la obtención de datos se escribió una prueba que verifica la obtención de la información de cada una de ellas.

La primera fuente de la que se extrajo información fueron los archivos que publica cada 10 minutos la página de la CONAGUA en el sitio http://smn.cna.gob.mx/emas/.

La técnica de Web Scraping fue utilizada para la extracción de los archivos que generan las diferentes estaciones ubicadas en los estados de las república (véase Figura 8.8).

Para ello se busco la url de cada archivo generado en todas las estaciones del país con ayuda de la biblioteca tagsoup.

Después de obtener las urls se persistieron en una base de datos no relacional de tipo llave-valor, en donde la llave es el valor de la latitud y longitud de la estación que genera la información

Este proceso se ejecuta una sola vez en la aplicación, ya que cuando se despliega se verifica que ya existan las urls de los archivos en la base de datos

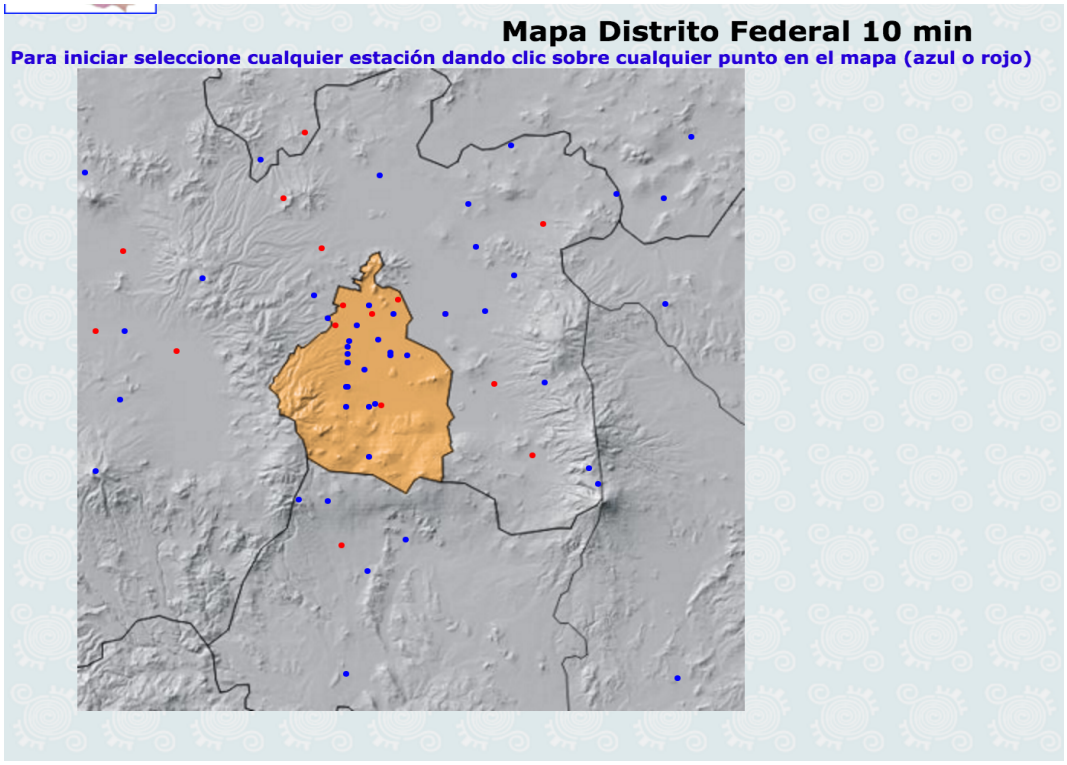


Figura 8.10: Estaciones con información climática del Distrito Federal

Ya que se tienen la información de los archivos disponible, se toma busca en la base la url del que tenga la latitud y longitud más cercana a los parámetros de consulta de la API para descargarlo y posteriormente iniciar el proceso extracción de las variables

La segunda fuente con la que se intenta complementar el modelo es Weather Underground.

Este sitio expone un servicio que recibe un código de ciudad para obtener la información climática (véase Figura 8.9).

Finalmente, si el modelo de datos aún no está completo, se consulta a la API de Forecast.io para buscar los datos faltantes. Esta es la última opción de búsqueda ya que la API tiene un número limitado de consultas por día.


```
{
  - conds: {
      - ICIUDADD120: {
            epoch: 1447935840,
            ageh: 0,
            agem: 3,
            ages: 42,
            type: "PWS",
            id: "ICIUDADD120",
            lat: "19.47519112",
            lon: "-99.11769867",
            adm1: "Ciudad De M xico",
            adm2: "",
            country: "MX",
            neighborhood: "Calle Diamante",
            dateutc: "2015-11-19 12:01:55",
            winddir: "-9999",
            windspeedmph: "-9999.0",
            windgustmph: "-999.0",
            humidity: "74",
            tempf: "61.7",
            rainin: "-999.00",
            dailyrainin: "-999.00",
            baromin: "30.38",
            dewptf: "53.4",
            weather: "",
            clouds: "",
            windchillf: "-999",
            heatindexf: "62",
            softwaretype: "Netatmo",
            elev: "7345",
            maxtemp: "66.4",
            maxtemp_time: "12:09AM",
            mintemp: "61.7",
            mintemp_time: "5:51AM",
            maxdewpoint: "54.7",
            mindewpoint: "52.1",
            maxpressure: "30.38",
            minpressure: "30.34",
            maxwindspeed: "-9999",
            maxwindgust: "-999",
            maxrain: "-999.00",
            maxheatindex: "66",
```


Figura 8.11: Consulta al servicio de WeatherUnderground

8.3.3. Restricciones

En algunas ocasiones no será posible encontrar toda la información climática o de contaminación.

El funcionamiento del módulo depende de las fuentes de información disponibles, por lo que si alguna falla o no está disponible no será posible mostrar correctamente la información.

Si el formato de los archivos de la CONAGUA cambia también ocurrirán errores en la aplicación, sin embargo hay pocas probabilidades de que esto ocurra.

8.3.4. Arquitectura

El módulo se realizó pensando en una arquitectura orientada a Microservicios.

Smart Owl es un microservicio que se despliega con un Tomcat embebido y expone la información siguiendo las convenciones de una aplicación REST.

El siguiente diagrama de clases muestra el modelo que define el estándar de los datos.

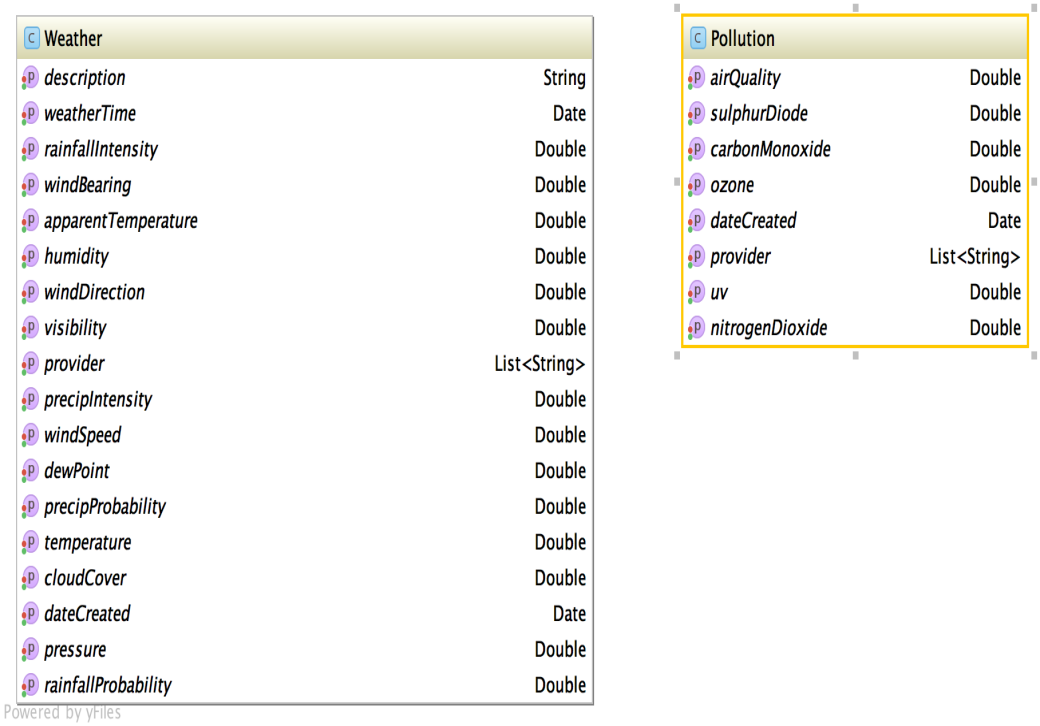
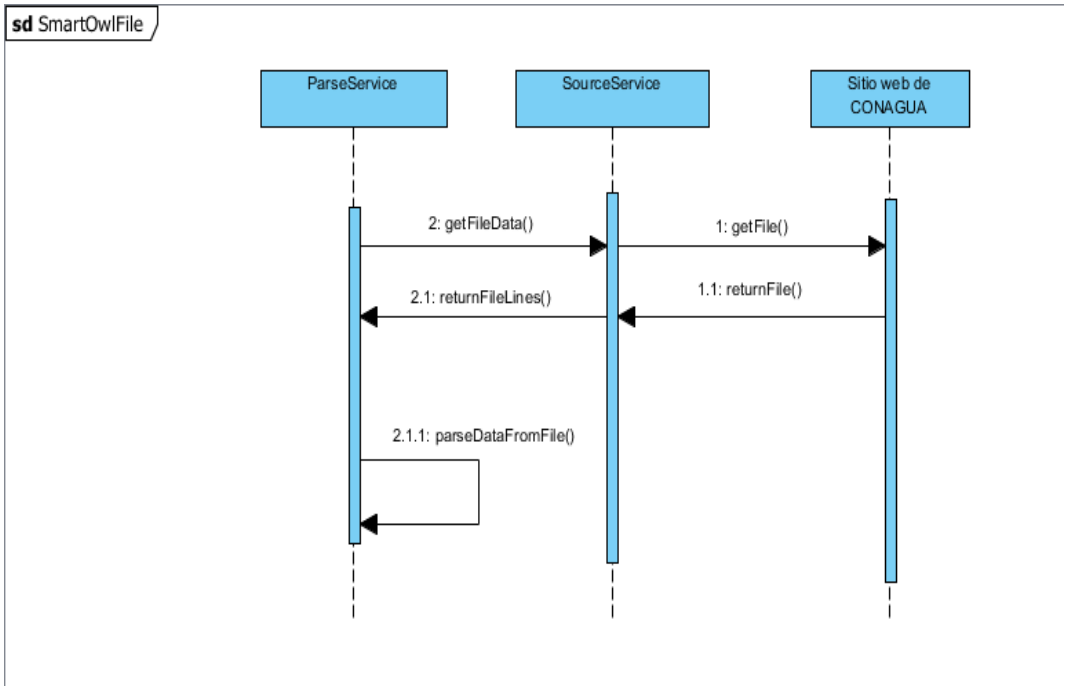
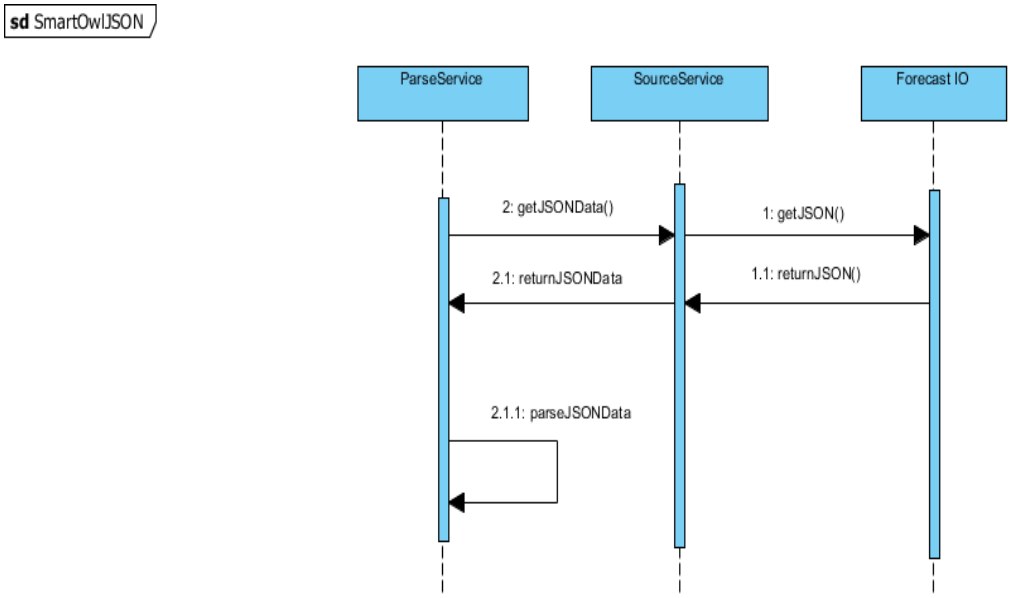


Figura 8.12: Diagrama de clases

A continuación se muestran los diagramas de secuencia planteados para el funcionamiento del módulo mencionado.





8.3.5. Estudio de Factibilidad

El desarrollo de la aplicación es posible ya que existen fuentes gratuitas que proveen la información necesaria para completar el modelo de datos que quiere exponerse y estandarizarse.

Las fuentes para la construcción de la API son las siguientes:

- Archivos de texto de texto que provee la CONAGUA (Comisión Nacional del Agua) con información climática
- Información de la página de Weather Underground
- API de Forecast.io

Hubo complejidad en la obtención de los datos ya que las fuentes presentan estructuras muy variadas, sin embargo, con ayuda de las pruebas unitarias fue posible implementar la funcionalidad para llenar el modelo de datos propuesto de manerá sencilla.

8.3.6. Implementación

Para el desarrollo del módulo se hizo uso del lenguaje de programación Groovy.

Smart Owl es un microservicio que se despliega con ayuda de SpringBoot y expone dos urls: /weather y /pollution que reciben los parámetros latitude y longitude para la búsqueda de la información.

Se escribió un conjunto de pruebas unitarias para la funcionalidad de la obtención de datos y la estandarización de la información con la ayuda de Spock Framework.

Gradle fue útli para las tareas de testing, administración de dependencias y la implementación del framework SpringBoot para el despliegue de la aplicación.

8.3.7. Pruebas

Package mx.ipn.ambienta2mx.smartOwl

all > mx.ipn.ambienta2mx.smartOwl

7	0	0	8.311s
tests	failures	ignored	duration

100%
successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
DistanceServiceSpec	1	0	0	0.135s	100%
ParseServiceSpec	2	0	0	3.155s	100%
SourceServiceSpec	4	0	0	5.021s	100%

Package mx.ipn.ambienta2mx.smartOwl

all > mx.ipn.ambienta2mx.smartOwl

7	0	0	8.311s
tests	failures	ignored	duration

100%
successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
DistanceServiceSpec	1	0	0	0.135s	100%
ParseServiceSpec	2	0	0	3.155s	100%
SourceServiceSpec	4	0	0	5.021s	100%

Package mx.ipn.ambienta2mx.smartOwl

all > mx.ipn.ambienta2mx.smartOwl

7	0	0	8.311s
tests	failures	ignored	duration

100%
successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
DistanceServiceSpec	1	0	0	0.135s	100%
ParseServiceSpec	2	0	0	3.155s	100%
SourceServiceSpec	4	0	0	5.021s	100%

Package mx.ipn.ambienta2mx.smartOwl

all > mx.ipn.ambienta2mx.smartOwl

7	0	0	8.311s
tests	failures	ignored	duration

100%
successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
DistanceServiceSpec	1	0	0	0.135s	100%
ParseServiceSpec	2	0	0	3.155s	100%
SourceServiceSpec	4	0	0	5.021s	100%

8.4. Fast Eagle

8.4.1. Definición y objetivos

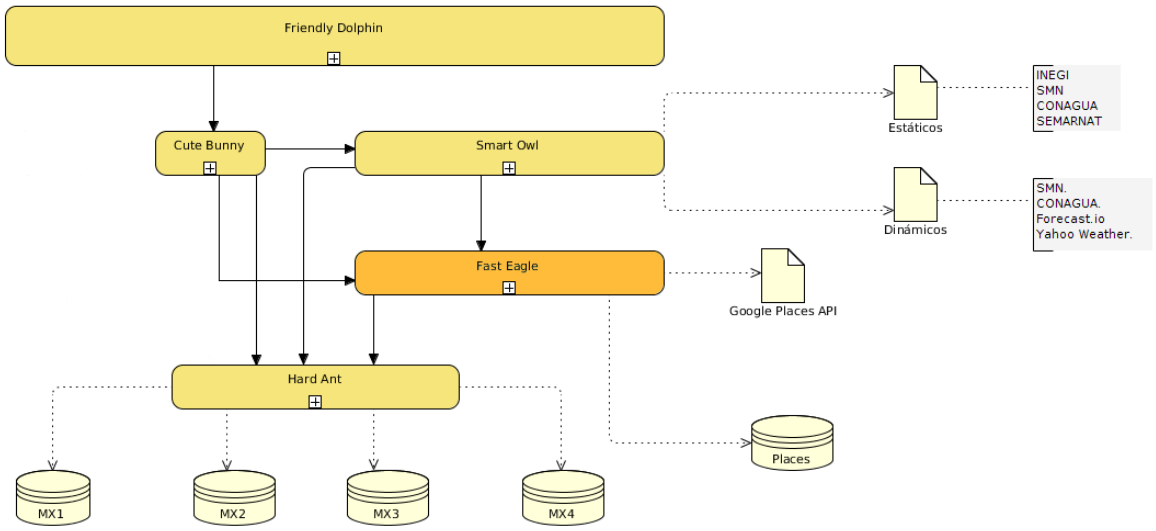


Figura 8.13: Fast Eagle, Módulo de Ambienta2MX.

El módulo Fast Eagle forma parte de la arquitectura final de Ambienta2MX. El propósito principal de este módulo es brindar la información cartográfica de México por medio de un servicio expuesto, considerando latitud, longitud o nombre de la localidad deseada (véase Figura 8.11).

Fast Eagle se encargará de exposición de datos brindados por el (Instituto Nacional de Estadística y Geografía) INEGI considerando los su politica de datos públicos que tiene, tomando como base la cartografía del territorio nacional.

Actualmente la información que proporciona el INEGI se encuentra en archivos de tipo CSV o bien, terceros se han encargado de estandarizar la información de forma relacional considerando MySQL como gestor principal de información.

Sin embargo, por la necesidad de búsquedas geográficas, se generó una migración de la información a un modelo de datos orientado a documentos, siguiendo la especificación GeoJSON en su versión del año 2008. Sólo basta con realizar un proceso de exportación al gestor de de documentos Mongo, y generar los índices geográficos. [35]

La información que proporciona el INEGI carece de campos esenciales para la estandarización de la información cartográfica considerando el formato propuesto por el equipo de trabajo, para dar solución a ese contratiempo se hará uso de servicios externos que ya cuentan con información definida, es decir, que su información ha pasado bajo un cierto proceso de limpieza y regulación, por ejemplo, los servicios de Google Places API.

8.4.2. Alcances

Fast Eagle, tendrá como alcance principal el brindar la información de las localidades del territorio nacional utilizando un servicio a demanda, en contraste con otro tipo de recolección como lo es el Crowd Sourcing[37], las bases dependientes de sistemas externos tendrán siempre una unión, aunque sea mínima,con éstos debido a que la información suele actualizarse de forma periódica.

Éste módulo sólo podrá consultar la información utilizando índices georeferenciados y coincidencia parcial con cadenas, es decir, sólo dos tipos de servicios serán expuestos.

Sin embargo, la unión con estos sitemas tiende a ser mínima conforme la comunidad conmienza a hacer uso del módulo o sistema en cuestión.

8.4.3. Restricciones

Una de las principales limitantes del odulo es la dependencia de terceros conforme el proyecto comienze a obtener información, es decir, al ser utilizado como una base a demanda, la información siempre tendrá que ser procesada por un tercero, posteriormente Íla dependencia será menor conforme los puntos sean más en la base de datos de tipo Places.

Al considerar una base orientada a documentos, cómo lo es Mongo, el problema principal es la gestión de la inforación y la posible compatibilidad con otros sistemas, por ejempo, el Stack Geográfico conocido como OSGeo[38], no soporta consultas de forma directa con este tipo de bases.

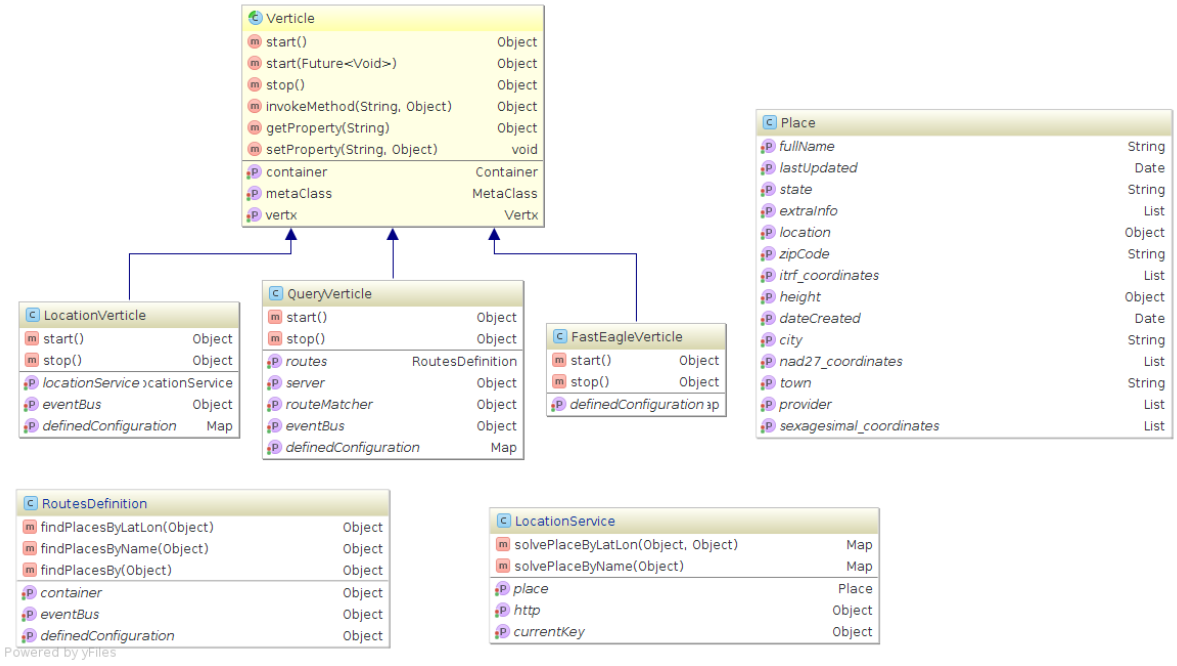
8.4.4. Arquitectura

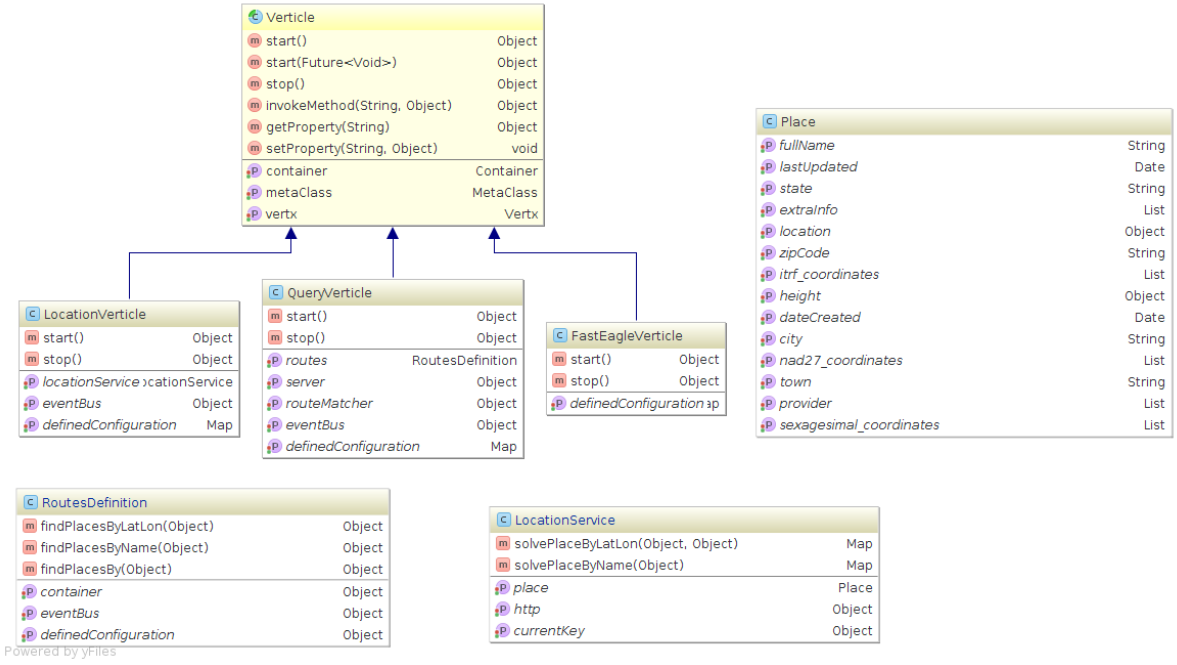
A continuación se muestra con más detalle la arquitectura y diagramas que componen al módulo Fast Eagle.

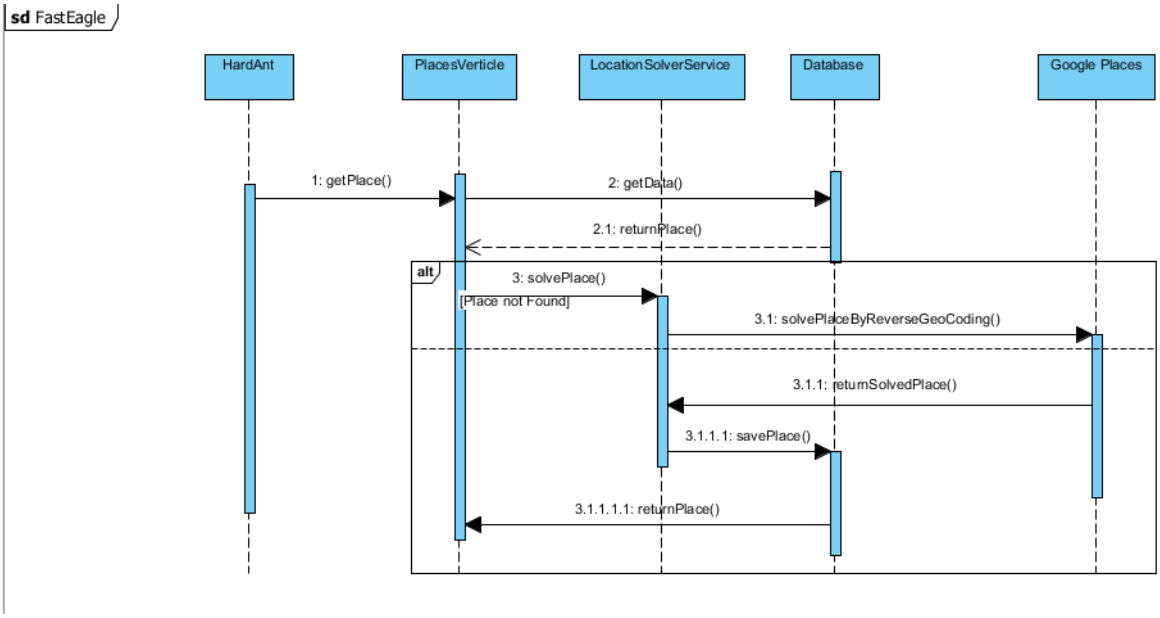
El objetivo principal de este módulo es brindar a los demás componentes del sistema información cartográfica de una localidad de país, considerando como valores de entrada, latitud y longitud o bien el nombre del lugar.

En caso de que no exista la información deseada por el usuario o algún otro módulo del sistema en la cartografía (base de datos llamada Places), Fast Eagle tratará de resolver la información en fuentes externas, persistiendo el modelo resuelto y regresando esa información al solicitante.

Fast Eagle cuenta con varios procesos desarrollados, la integración de cada proceso y su respectiva integración da solución a un problema de estandarización, resolución y consulta de datos geográficos vía Latitud, Longitud y Ubicación.


Verticle
start() Object
start(Future<Void>) Object
stop() Object
invokeMethod(String, Object) Object
getProperty(String) Object
setProperty(String, Object) void
container Container
metaClass MetaClass
vertx Vertx
LocationVerticle
start() Object
stop() Object
locationService cationService
eventBus Object
definedConfiguration Map
QueryVerticle
start() Object
stop() Object
routes RoutesDefinition
server Object
routeMatcher Object
eventBus Object
definedConfiguration Map
FastEagleVerticle
start() Object
stop() Object
definedConfiguration ap
Place
fullName String
lastUpdated Date
state String
extraInfo List
location Object
zipCode String
itrf_coordinates List
height Object
dateCreated Date
city String
nad27_coordinates List
town String
provider List
sexagesimal_coordinates List
RoutesDefinition
findPlacesByLatLon(Object) Object
findPlacesByName(Object) Object
findPlacesBy(Object) Object
container Object
eventBus Object
definedConfiguration Object
LocationService
solvePlaceByLatLon(Object, Object) Map
solvePlaceByName(Object) Map
place Place
http Object
currentKey Object


Verticle
start() Object
start(Future<Void>) Object
stop() Object
invokeMethod(String, Object) Object
getProperty(String) Object
setProperty(String, Object) void
container Container
metaClass MetaClass
vertx Vertx
LocationVerticle
start() Object
stop() Object
locationService cationService
eventBus Object
definedConfiguration Map
QueryVerticle
start() Object
stop() Object
routes RoutesDefinition
server Object
routeMatcher Object
eventBus Object
definedConfiguration Map
FastEagleVerticle
start() Object
stop() Object
definedConfiguration ap
Place
fullName String
lastUpdated Date
state String
extraInfo List
location Object
zipCode String
itrf_coordinates List
height Object
dateCreated Date
city String
nad27_coordinates List
town String
provider List
sexagesimal_coordinates List
RoutesDefinition
findPlacesByLatLon(Object) Object
findPlacesByName(Object) Object
findPlacesBy(Object) Object
container Object
eventBus Object
definedConfiguration Object
LocationService
solvePlaceByLatLon(Object, Object) Map
solvePlaceByName(Object) Map
place Place
http Object
currentKey Object


sd FastEagle
HardAnt
PlacesVerticle
LocationSolverService
Database
Google Places
1: getPlace()
2: getData()
2.1: returnPlace()
alt
3: solvePlace()
[Place not Found]
3.1: solvePlaceByReverseGeoCoding()
3.1.1: returnSolvedPlace()
3.1.1.1: savePlace()
3.1.1.1.1: returnPlace()

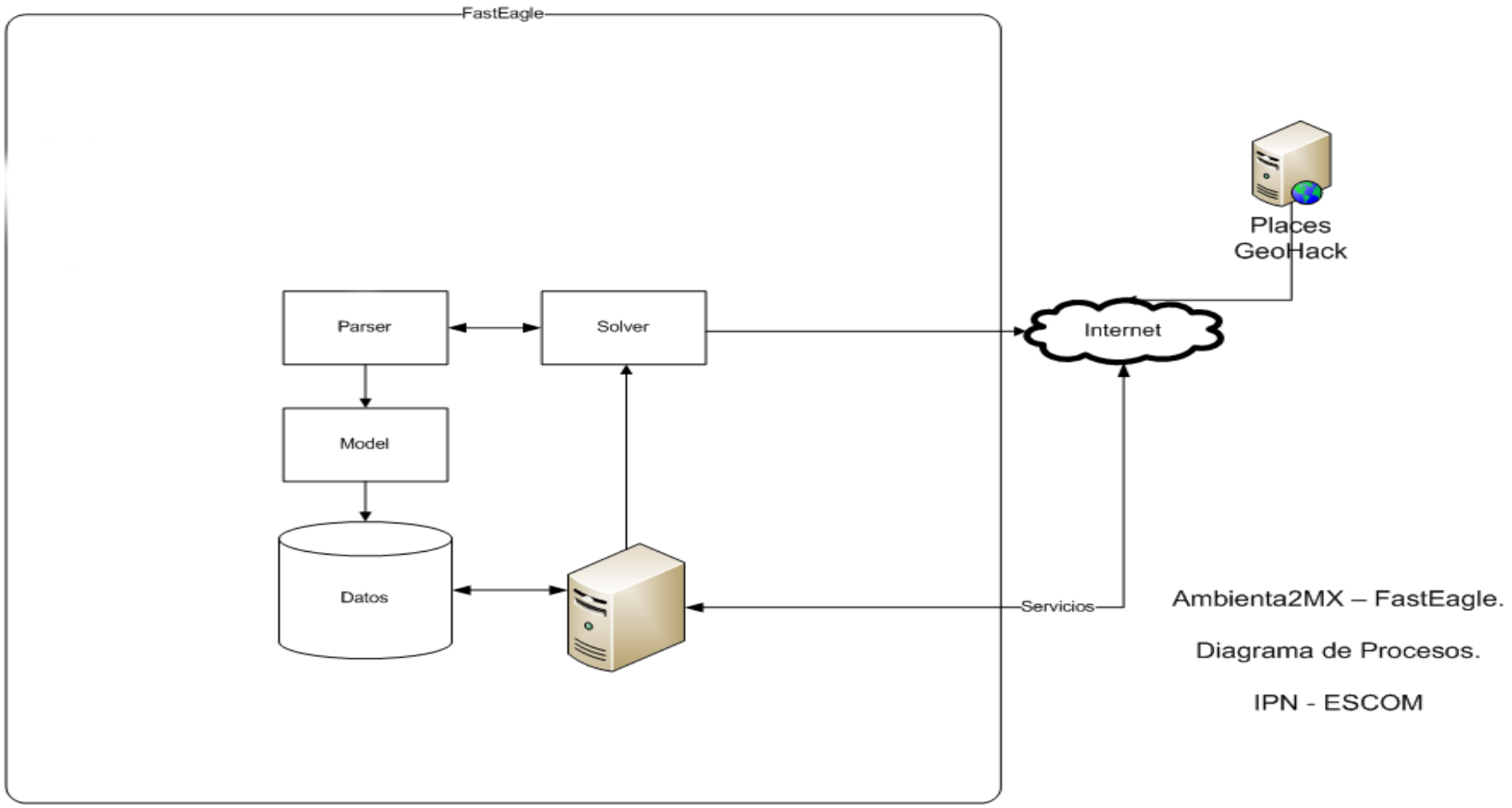


Figura 8.14: Diagrama por bloques de Fast Eagle

En el diagrama se muestran tres módulos básicos, estos forman parte del núcleo de Fast Eagle (véase Figura 8.12), también podemos observar que se cuenta con la interacción de servicios de terceros como Google Places, también se cuenta con la exposición de los servicios a través de un servidor web.

***Parser* tomará los datos que el proceso de validación le arroje para transformar al estado propuesto por el equipo de trabajo (Véase modelo de datos). Considerando un proceso de resolución en caso de que la información proporcionada por el INEGI se encuentre incompleta no sea válida.**

Para toda la información que carezca de datos correctos *Solver* buscará una resolución en servicios de terceros, después de la resolución, los datos serán guardados en el gestor de bases de datos bajo el formato propuesto por el equipo de trabajo.

***Model* es la capa de interacción con la base de datos, ésta se encarga de las operaciones mejor conocidas como CRUD (Create, Read, Update and Delete), persistiendo la información en MongoDB, utilizando los canales de Vert.x para su convivencia con la antes mencionada.**

Para poder exponer los datos, se hará uso de un servidor web minimalista orientado a micro servicios desarrollado en Vert.x, éste será un servicio público que formará parte de la infraestructura final de Ambienta2MX.

El servicio expuesto se encargará de las búsquedas a nivel base de datos y en caso de no encontrar la información buscará en terceros para poder agregarla a la base de datos y así ir mejorando el contenido de nuestro índice cartográfico.

Considerando las bondades que nos brinda el framework Vert.x, se generó un canal para la resolución de la información utilizando el servicio de Google Maps, éste canal interactúa de forma directa con el servicio de places expuesto a todos los usuarios o módulos del sistema.

La comunicació principal será de tipo REST, para el proceso de obtención de información relacionada al territorio nacional.

8.4.5. Factibilidad

El desarrollo del proyecto se fue orillando a la eliminiación de ciertos procesos ya adecuación de tecnologías. Por ejemplo, la lectura de los archivos del INEGI (Fuentes expuestas en formato CSV), fue reemplazada por el uso de un script de migración entre un gestor público de tipo MySQL hacia un servicio Mongo, realizando también la generación de los índices geográficos.

Un problema importante fue la determinación de la cantidad de Verticles (Compomente de Vert.x) por qué un exceso de elementos a desplegar se veía reflejado en la saturación del servicio de base de datos Mongo, esto fue determinado desplegando 20, y posteriormente disminuir el valor hasta llegar al valor de 4.

8.4.6. Implementación

El código y el proyecto como tal se encuentran en línea utilizando Git[38] en conjunto con Github[?] (véase Figura 8.13), para el proceso de control de versiones. Para la gestión de tareas y desarrollo del proyecto fue utilizado Gradle en conjunton con Vert.x.

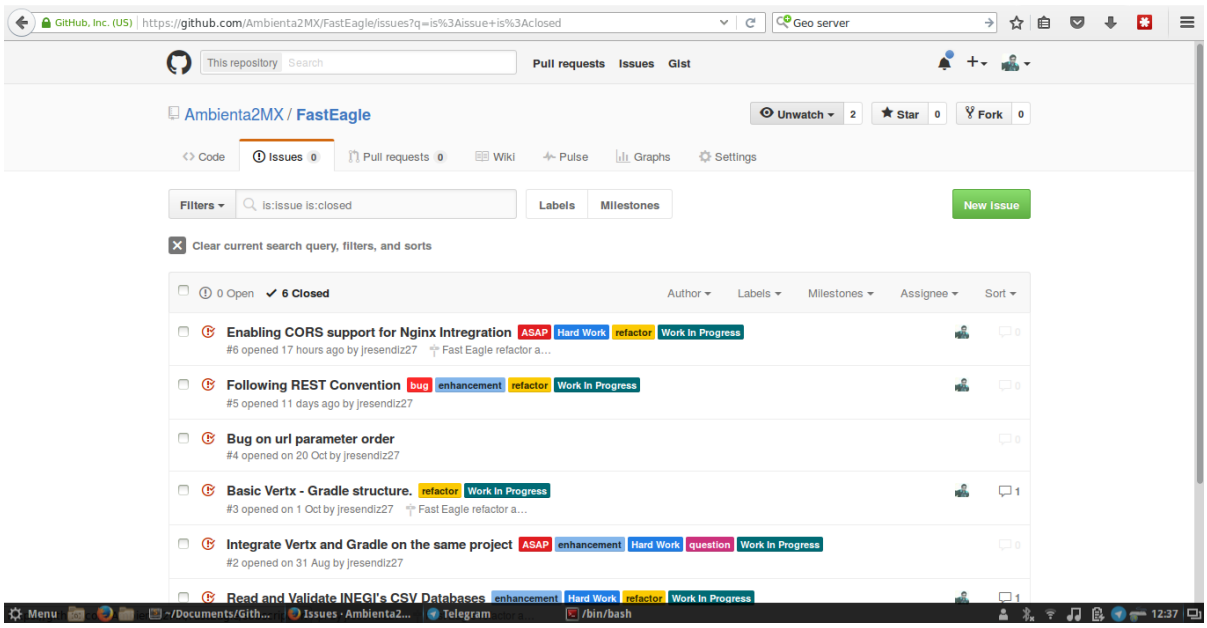
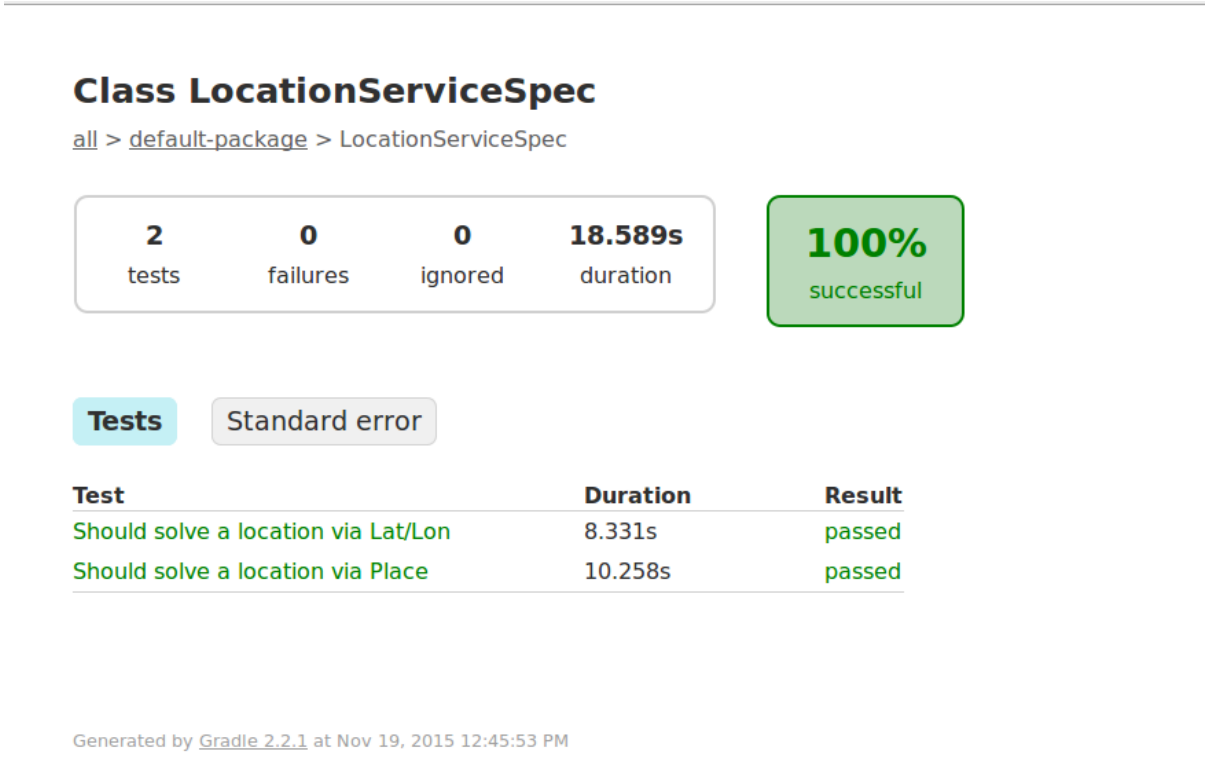


Figura 8.15: Fast Eagle, Integración con Github.

Las ventajas que nos ofrece Git/Github es la integración con procesos de desarrollo utilizando metodologías ágiles. La siguiente imagen muestra el control y manejo de issues (análogos a objetivos o features) que fueron desarrollados para cumplir con los objetivos y mejoras planteadas para cada módulo.

8.4.7. Pruebas y Capturas de pantalla

A continuación se muestran algunas pruebas realizadas y las capturas de pantalla del servicio REST y la respuesta qué este nos regresa.



Class LocationServiceSpec

all > default-package > LocationServiceSpec

2	0	0	18.589s
tests	failures	ignored	duration

100% successful

Tests | Standard error

Test	Duration	Result
Should solve a location via Lat/Lon	8.331s	passed
Should solve a location via Place	10.258s	passed

Generated by Gradle 2.2.1 at Nov 19, 2015 12:45:53 PM

Figura 8.16: Fast Eagle, Pruebas de Fast Eagle.

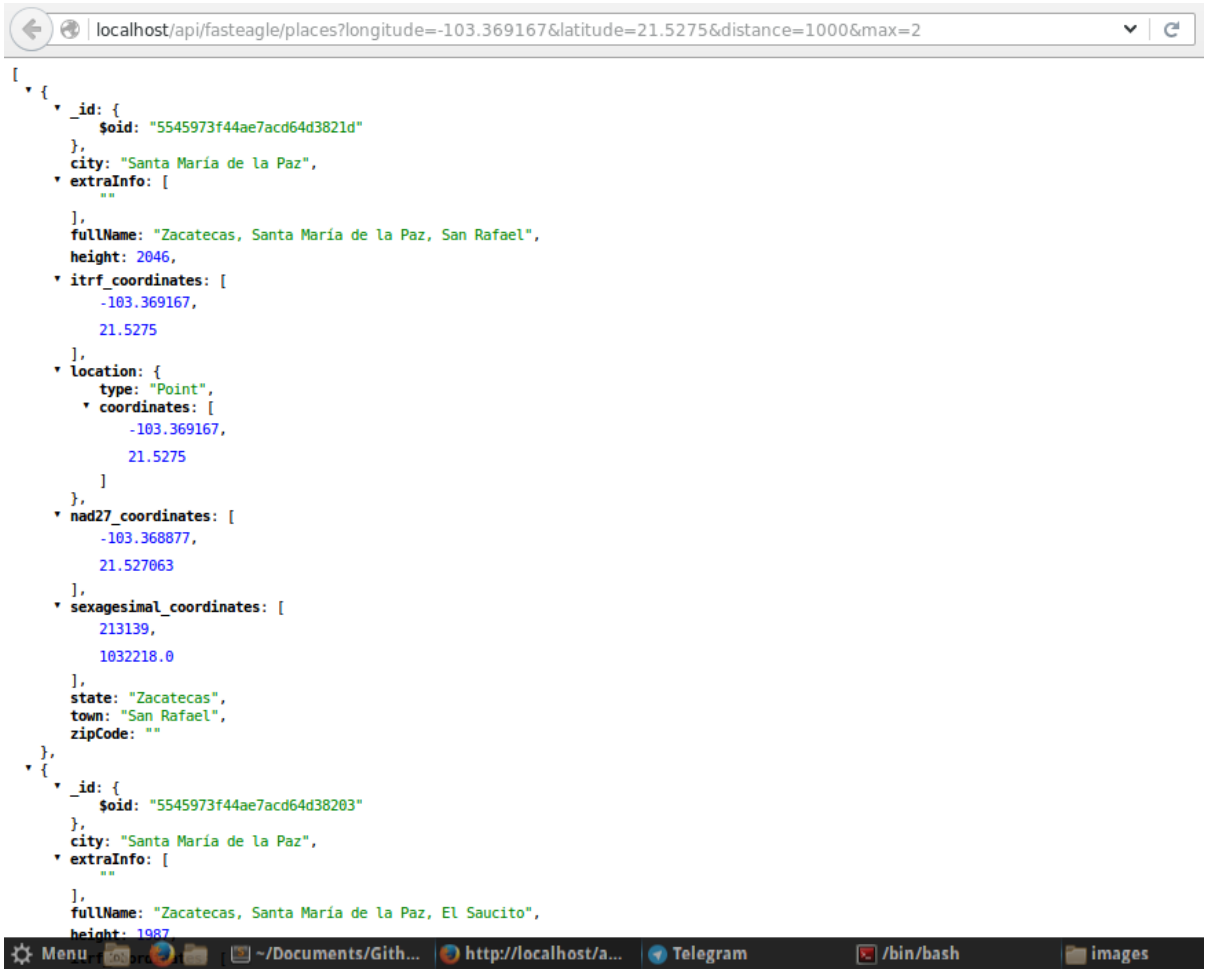


Figura 8.17: Fast Eagle, Respuesta del servicio rest.

8.5. Hard Ant

8.5.1. Definición y objetivos

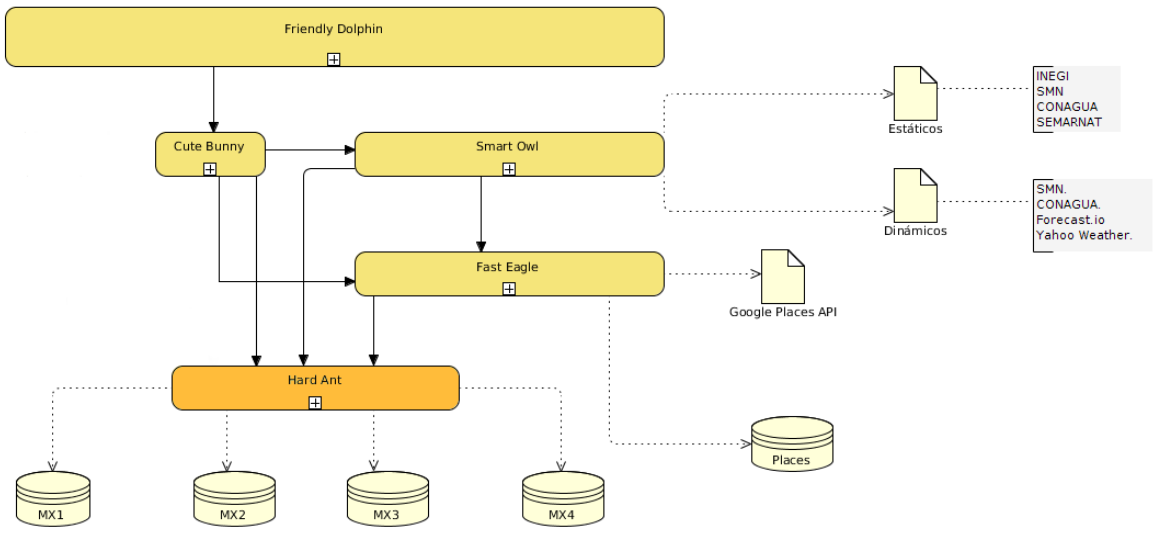


Figura 8.18: Hard Ant, Módulo de Ambienta2MX.

Hard Ant es uno de los bloques funcionales base de Ambienta2MX, su función principal es la de enrutar las peticiones a las bases de tipo MX además de brindar la solución cartográfica (a nivel ubicación) interactuando con Fast Eagle (véase Figura 8.16).

Como complemento a la arquitectura, también contará con el proceso del registro masivo de información, exponiendo servicios que el módulo Smart Owl usará de forma constante para persistir la información estandarizada de diversas fuentes de información.

Hard Ant tiene como objetivo brindar los canales de acceso a las bases MX definidas en el diagrama general mediante servicios HTTP, estos servicios cumplen con la tarea de inserción y extracción de la información.

Este módulo es lo que sería considerado la capa del modelo de datos en un patrón MVC, ya que es la que tiene contacto de forma directa con los datos almacenados en las bases de datos orientadas a documentos gestionadas por Mongo.

8.5.2. Alcances

Hard Ant fue diseñado para poder satisfacer la demanda en las cuatro bases de datos distrubidas, contando con un índice que se encarga de enviar la consulta a una base defindа con base en la ubicación que le proporciona Fast Eagle.

Éste módulo también cuenta con la tarea de la generación de archivos de tipo CSV y JSON, que pueden ser utilizados para los fines que el usuario requiera. Todos los accesos realizarán mediante servicios de tipo REST siguiendo una Convención sobre configuración.

Se utilizará un pool de conexiones a la base para garantizar el acceso o escritura a los datos además de brindar la posibilidad de respuestas asincronas y no bloqueantes entre las consultas realizadas.

8.5.3. Restricciones

Para nuestro caso de estudio sólo se toma la información de la base de MX4, debido a que es la que contiene la información del Distrito Federal. Una de las limitantes más grandes es el consumo de memoria para la generación de archivos ya qué en principio se guardan de forma temporal y luego son enviados al cliente. La infraestructura que estamos manejando es limitada en recursos debido a que es gratuita.

8.5.4. Arquitectura

A continuación se mostrará el de clases que define la estructura de Hard Ant.

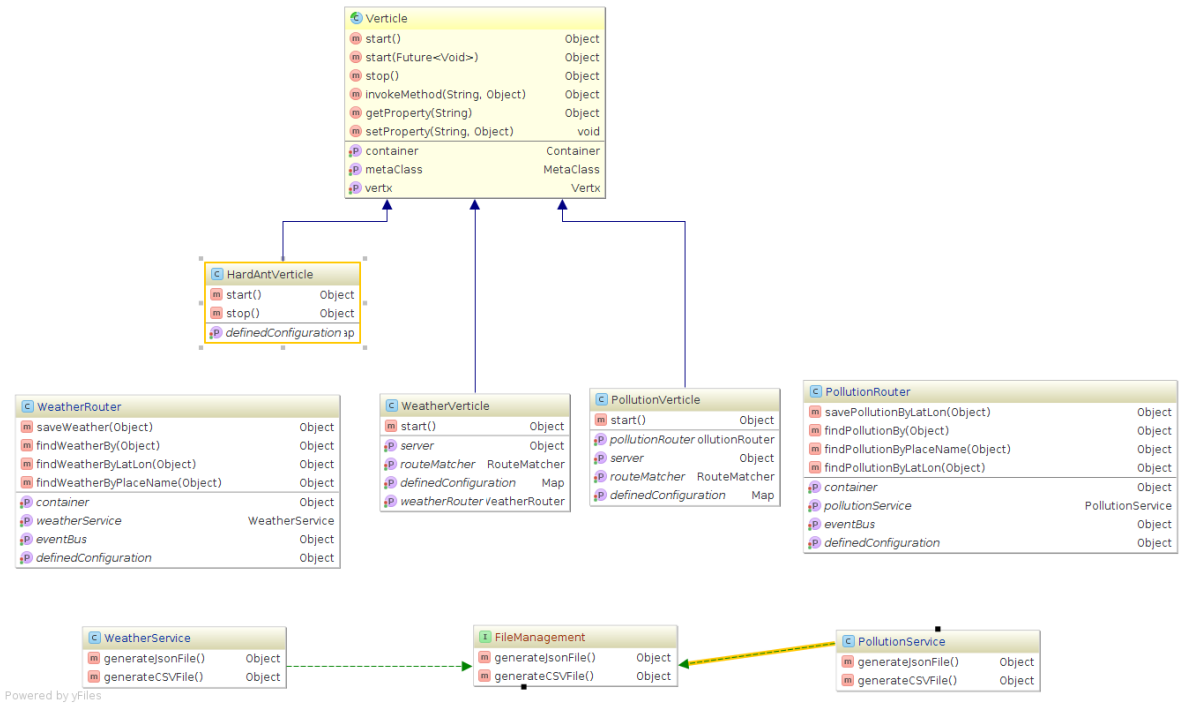


Figura 8.19: Diagrama de clases de Hard Ant

Las clases mostradas se ven reflejadas en el uso y adaptación de "Verticles", que son los componentes que se encargarán de atender peticiones, ya sea vía servicios REST o por comunicación mediante canales (Sockets), cómo se muestra en el siguiente diagrama a bloques.

En el diagrama se puede apreciar la replicación de "Verticles" que interactúan a través del mismo canal de información, esto brinda disponibilidad del servicio ya que las peticiones son atendidas y procesadas no sólo por un elemento existente si no varios (véase Figura 8.17). La interacción con este servicio se realizará utilizando servicios REST, utilizando el módulo de Cute Bunny como medio de comunicación con la vista que tendrá el usuario final.

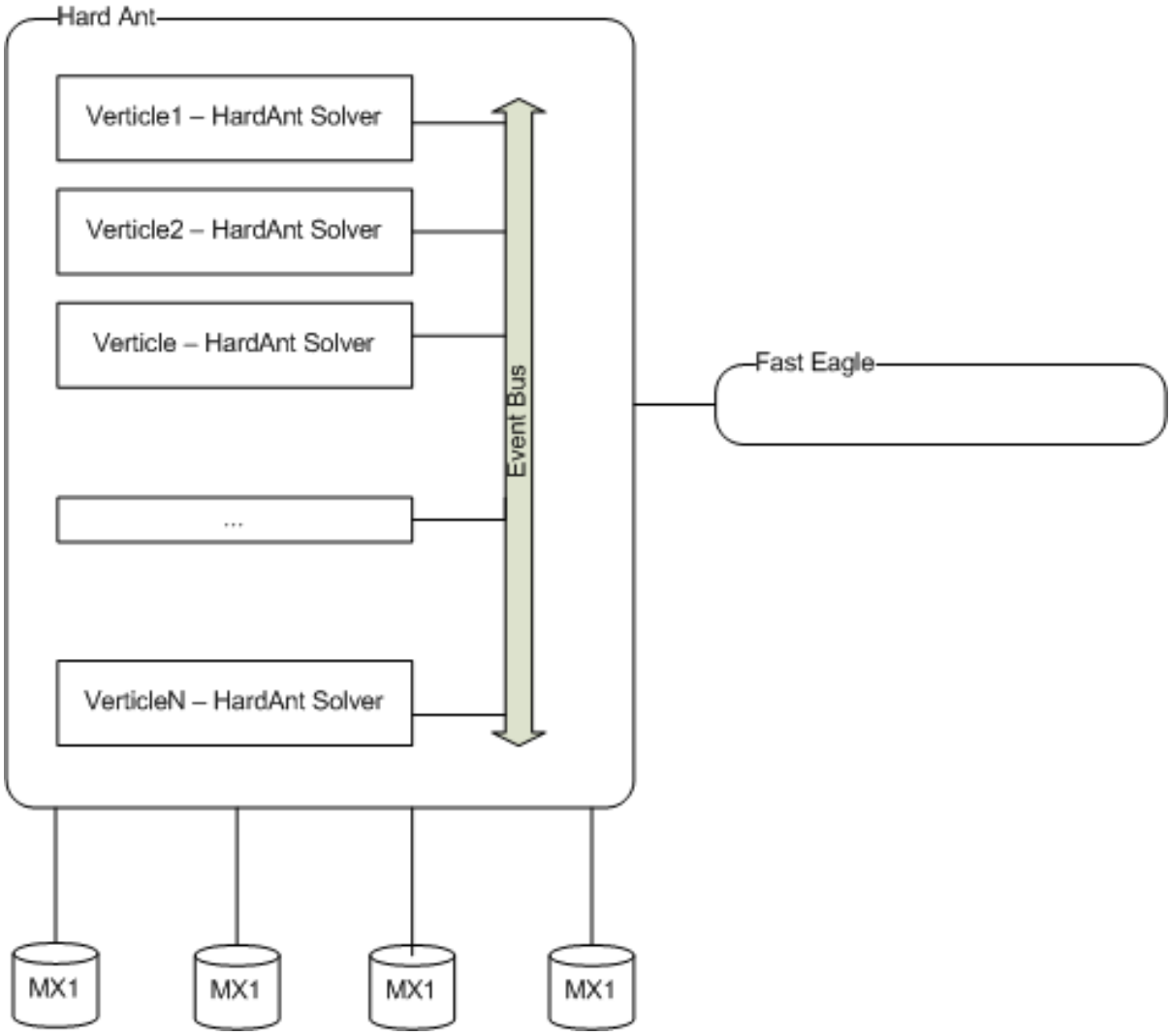


Figura 8.20: Diagrama a bloques de Hard Ant

Considerando trabajos más pesados (Obtención de datos climáticos considerando un radio, cadena de búsqueda o bien información de un punto definido) se utilizarán procesos en segundo plano, esto es posible gracias a la implementación de "Verticles" nativas de Vert.x, tecnología que será usada para el desarrollo y despliegue final de éste módulo.

Actualmente se propone el despliegue de varias "Verticles", además de las que el mismo framework despliega para el consumo de las bases de datos [33].Quedando en ocho, cuatro encargadas de la comunicación y distribución de la información y las cuatro restantes para atender peticiones (véase Figura 8.18).

De forma específica, también se hará uso de los canales implementados por Vert.x para generar una comunicación con las bases de datos con la finalidad de poder generar la distribución de estas. Los canales podrían ser accedidos siempre y cuando se pertenezca a la misma red o bien al cluster de Vert.x, esta modalidad no ha sido habilitada por cuestiones de seguridad.

8.5.5. Factibilidad

Considerando el mismo modo de trabajo que en

8.5.6. Implementación

Hablando más a detalle, Hard Ant, puede ser desplegado utilizando la línea de comandos gracias a las funciones que nos provee Gradle, framework utilizado para el maquetado, resolución de dependencias y gestión de tareas del proyecto. [34]

El código y el proyecto como tal se encuentran en línea utilizando Git[38] en conjunto con Github[?], para el proceso de control de versiones (véase Figura 8.19). Para la gestión de tareas y desarrollo del proyecto fue utilizado Gradle en conjunton con Vert.x.

Las ventajas que nos ofrece Git/Github es la integración con procesos de desarrollo utilizando metodologías ágiles. La siguiente imagen muestra el control y manejo de issues (análogos a objetivos o features) que fueron desarrollados para cumplir con los objetivos y mejoras planteadas para cada módulo.

Este módulo se puede encontrar en cualquier otra máquina, debido al método de comunicación utilizado entre los módulos, sólo basta con realizar la configuración de las direcciones o dominios donde se encuentra alojado para qué este puede pueda ser accedido.

8.5.7. Pruebas y Capturas de pantalla

A continuación se muestran algunas pruebas realizadas y las capturas de pantalla del servicio REST y la respuesta qué este nos regresa.

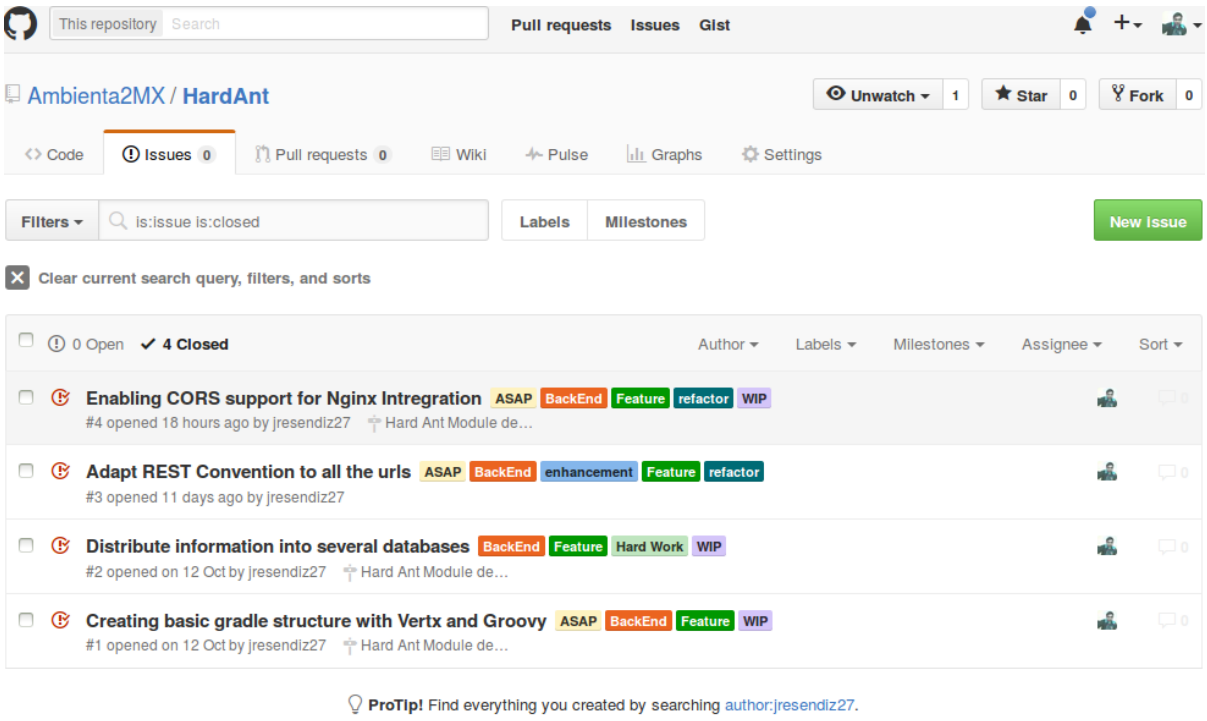


Figura 8.21: Hard Ant, Integración con Github.

Figura 8.22: Hard Ant, Pruebas unitarias.

Figura 8.23: Hard Ant, Respuesta del servicio rest.

Capítulo 9

Análisis y gestión de riesgos

9.1. Definición y clasificación

Los objetivos de la gestión de riesgos son identificar, controlar y eliminar las fuentes de riesgo antes de que empiecen a afectar el cumplimiento de los objetivos del proyecto.

El riesgo siempre implica una incertidumbre y una pérdida potencial, es necesario llevar una cuantificación de estos parametros, por lo que suelen ser clasificados en diferentes categorias por ejemplo:

- Riesgos del proyecto
- Riesgos técnicos
- Riesgos de negocio

Es necesario llevar un proceso de administración de riesgos y planes de contingencia para poder controlar esos inesperados eventos. Primero se tienen que identificar los riesgos, considerando principalmente los más potenciales, posteriormente analizarlos y dandoles una prioridad, generar planes de contingencia para finalmente supervizarlos y actuar conforme a lo acordado.

Los riesgos pueden ser clasificados respecto a estudios previos o con relación a la experiencia de trabajo que tengan los desarrolladores y analistas del proyecto, sin embargo, generalmente suelen asignarseles ciertas ponderaciones por convención, como se muestra a continuación:

- Muy bajo (<10 %)
- Bajo (10 - 25 %)
- Moderado (25 - 50 %)
- Alto (50 - 75 %)
- Muy Alto (>75 %)

Todos los riesgos deben encontrarse también clasificados entre cualquiera de las cuatro valoraciones: Insignificante, Tolerable, Serio, Catastrófico; los planes de contingencia suelen ser desarrollados para aquellos riegos con probabilidad de moderada a muy alta considerando, tomando en cuenta un impacto serio o castastrófico.

A lo largo del desarrollo de Ambienta2MX surgieron y surgirán eventos y sucesos inesperados que serán considerados como riesgos potenciales dependiendo el impacto que estos lleguen a tener.

Dentro del proceso de planeación del proyecto se decidió delegar o depender de los servicios de terceros en cuanto a infraestructura física, esto implica que se disminuyen riesgos de instalación o mantenimiento a equipos de cómputo o redes computacionales.

Sin embargo, esto nos deja a merced de nuestros proveedores de servicios. La siguiente tabla muestra los riesgos más significativos considerados a lo largo del desarrollo del proyecto y sus respectivos planes de acción.

Tabla de Riesgos				
Descripcion	**Tipo de Riesgo**	**Valoración**	**Porcentaje**	**Plan de acción**
No disponibilidad de los servidores de Amazon	Técnico	Catastrófico	1 %	Migración parcial a servicios de hosting gratuitos como Heroku u Openshift.
Falta de presupuesto	Proyecto	Serio	25 %	Buscar un proceso de incubación en empresas como Apache, Eclipse y migrar la plataforma a servicios de hosting gratuitos como Heroku u Openshift.
Falta por razones personales de miembros del equipo	Proyecto	Serio	25 %	Rediseñar y adaptar las tareas con nuevos integrantes y habilitar forma de trabajo remota considerando fines de semana.
Extinción de recursos de información	Proyecto, Técnico	Catastrófico	10 %	Cambiar las reglas de negocio y buscar nuevas fuentes de información como SaaS y desplegar los módulos de estandarización nuevamente.
Necesidad de escalabilidad de la plataforma	Técnico	Serio	10 %	Definir el tipo de escalabilidad a usar dependiendo los recursos monetarios existentes.
Ataques de Denegación de Servicios (DoS)	Técnico	Tolerable	10 %	Cambiar el tipo de dirección en los servidores de Amazon y agregar caché al sistema.
Adaptación a nuevos estandares	Técnico	Tolerabe	15 %	Rediseñar y extender la funcionalidad de la aplicación, dando soporte a ambos modelos de intercambio de datos.

Cuadro 9.1: Analisis de Riesgos

Capítulo 10

Conclusiones

Se alcanzaron los objetivos planteados en el protocolo, a continuación se listan:

- Estandarizar variables ambientales.
- Mostrar la información recaudada mediante servicios REST.
- Visualizar la información mediante mapas y tablas.
- Generar las bases de datos necesarias y direccionar las peticiones de los servicios.

Durante el desarrollo del proyecto, el equipo de trabajo se vió inmerso en ciertos conflictos, todos ellos pudieron verse resueltos después de generar un ambiente de dialogo apropiado.

Se cometieron ciertos errores durante la planeación lo cual trajo consigo qué muchos de los objetivos iniciales planteados en el protocolo (el que se iba a entregar justo antes del paro de labores del IPN en el año 2014) se vieran truncados o bien eliminados.

Uno de los conflictos más grandes fue la salida de un compañero cercano del equipo de trabajo por razones personales, esto aumentó la carga de trabajo para los actuales representantes de Ambienta2MX, tomando en cuenta que todos ya formabamos parte de un ambiente laboral, los tiempos y presiones aumentaron durante mediados de Trabajo Terminal 1 y todo el desarrollo de Trabajo Terminal 2.

La planeación es algo que como técnicos suele ser muy complicado, sin embargo, se realizó lo posible por cumplir con los objetivos propuestos en el protocolo. Cabe destacar que la planeación fue cambiada varias veces, debido a los conflictos antes mencionados.

El conocimiento técnico adquirido a lo largo de estos meses fue muy considerable, ya que sirvió como complemento para las labores externas a la escuela despempeñadas por los ponentes de Ambienta2MX, mejorando así su enfoque como futuros ingenieros.

Considerando avances en el ámbito personal, hablando como equipo, podemos decir que este tipo de proyectos dejan un buen sabor de boca debido a la cantidad de vivencias que éstos dejan, ayudan a mejorar la formación de los futuros ingerieros con base en trabajos que tienen una aplicación real (no siempre se tiene una respuesta a corto plazo), y soluciones que deben contener lo mínimo para que sean de calidad.

Cómo equipo de trabajo, esperamos que este proyecto sea de utilidad o sirva como marco de referncia para futuras generaciones que deseen aplicar un proceso de estandarización de datos más ambicioso y aprendan del conocimiento que se queda como legado en éste documento.

Capítulo 11

Trabajo a Futuro

Gracias a una arquitectura orientada a servicios se puede extender la funcionalidad del sistema sin tener que invertir demasiado tiempo en el desarrollo de nuevas tareas o funciones.

Un ejemplo visto durante el desarrollo fue la implementación de un proceso de predición y consultas de tipo histórico. Estas consultas cuentan con una gran utilidad para poder realizar el proceso de análisis de datos climatológicos o de índices de contaminación.

También se pueden adaptar o agregar más fuentes de datos o bien de información, considerando bien, qué estos datos deben contener información georefernciada (Latitud/Longitud), para que puedan ser consultados y almacenados en las diversas bases de datos propuestas por el equipo de trabajo; considerando también el modelo de datos que debe ser generado en estructura dentro de los módulos pertinentes.

Bibliografía

[1] Mapa Digital de México, INEGI 2014, Available at: http://gaia.inegi.org.mx/mdm6/

[2] Modelo de datos CIM, 2013, Available at: https://earthsystemcog.org/projects/es-doc-models/cim

[3] GIS México - Sistemas de Información Geográfica, Available at: http://www.sigmexico.com.mx/

[4] GEOJson Format Specification, 2008, Available at: http://geojson.org/geojson-spec.html

[5] GIS (Geographic information system), 2010, Available at: http://education.nationalgeographic.com/education/encyclopedia/geographic-information-system-gis

[6] INEGI, Acerca del INEGI, N/A, Available at: http://www.inegi.org.mx/inegi/acercade/default.aspx

[7] Web Scrapping Introduction, Geoff Boeing, Berkeley University, 2014, Available At: http://dlab.berkeley.edu/sites/default/files/training_materials/web-scraping-talk.pdf

[8] Convention over configuration, Microsoft Developers Network, 2012, Available At: https://msdn.microsoft.com/en-us/magazine/dd419655.aspx.

[9] INSPIRE Project, Aboute INSPIRE, Available at: http://inspire.ec.europa.eu/

[10] NoSQL Explained, MongoDB Courseware, Available at: https://www.mongodb.com/nosql-explained

[11] Seven databases in seven weeks: A guide to Modern Databases and NoSQL movement, Eric Redmond, Pragmatic Bookself, 2013.

[12] MongoDB Case Study: Foursquare, Mongodb Courseware, Available at: http://www.mongodb.com/post/15400944604/mongodb-case-study-foursquare

[13] Weather Underground, About the Data, Available at: http://www.wunderground.com/about/data.asp

[14] Forecast.io, Data Sources, Available at: http://forecast.io/raw/

[15] Servicio Meteorológico Nacional, Conceptos, Available at: http://smn.cna.gob.mx/index.php?option=com_content&view=article&id=23&Itemid=120

[16] The JSON Data Interchagnge Format, ECMA International, 2013, Available at: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[17] Base de datos Estadísticos BADESNIARN, Semarnat, Available at: http://www.semarnat.gob.mx/temas/estadisticas-ambientales/badesniar?De=SNIARN

[18] Geographic System Information, Geographic Information Centre (Canada), Available at: http://gic.geog.mcgill.ca/gis-resources/

[19] Geoplatform Overview, Geoplatform.gov, 2015, Available at: https://www.geoplatform.gov/overview-page

[20] Common Core Metadata Schema, OpenData.gov, 2013, Available at: https://project-open-data.cio.gov/schema/

[21] Open Government Inititative, The White House, 2009, Available at: https://www.whitehouse.gov/open

[22] Agile Methodology, 2015, Available at: http://agilemethodology.org/

[23] The Scrum Reference Card, Michael James, 2015, Available at: http://scrumreferencecard.com/reference-card-de-scrum/

[24] Agile Methodologies for Software Development, Mike McLaughlin, Available at http://www.versionone.com/agile-101/agile-development-methodologies-scrum-kanban-lean-xp/

[25] Guidelines for Test-Driven Development, Jeffrey Palermo, 2006, Available at https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx

[26] Refactoring, Martin Fowler, Available at http://refactoring.com/

[27] Continuous Integration, Martin Fowler, Available at http://www.thoughtworks.com/continuous-integration

[28] Refactoring Improving the design of existing code, Martin Fowler, Pragmatic Bookshelf, 1999

[29] Prototyping Model, Margaret Rouse, N/A, Available at http://searchcio.techtarget.com/definition/Prototyping-Model

[30] Algorithms Make Better Predictions — Except When They Don't, Harvard Bussiness Review, Thomas C.Redman, 2014, Available at https://hbr.org/2014/09/algorithms-make-better-predictions-except-when-they-dont

[31] REST vs SOAP The difference between Soap and REST, Hacking Management, 2010, Available at: http://spf13.com/post/soap-vs-rest

[32] CONAGUA. Misión y Visión, 2012, Available at: http://www.cna.gob.mx/Contenido.aspx?n1=1&n2=27

[33] Understading Vert.x Architecture Part II, Tim Fox, 2014, Available at: http://www.cubrid.org/blog/dev-platform/understanding-vertx-architecture-part-2/

[34] Gradle beyond the bascis, Tim Berglund, O'Reilly, 2013.

[35] 2dsphere index running on Mongodb, Mongodb Team, 2015, Available at: https://docs.mongodb.org/manual/tutorial/build-a-2dsphere-index/

[36] Single Page Web Application: Javascript End to End, Michael S. Mikowski, Manning, 2013.

[37] Crowdsourcing Why the Power of the Crowd Is Driving the Future of Business, Jeff Howe, 2009

[38] DevOps El siguiente nivel de agilidad, Patrick Debois, Revista Software Guru, 2015.

Glosario

- **JSON**: Javascript Object Notation, Es un formato ligero de intercambio de datos.
- **ITRF**: Acrónimo de International Terrestrial Reference System (Marcos de Referencia Terrestre Internacional).
- **NAD27**: Acrónimo de North American Datum of 1927. Marco de referencia terrestre usado por el INEGI hasta el año 1998.
- **GIS**: Sistema de Información Geográfica.
- **API**: Application Programming Interface, conjunto de rutinas, protocolos o herramientas para construcción de software.
- **REST**: Representational State Transfer, Arquitectura de sofware para sistemas web basada en un patrón "Convention over configuration" con operaciones que funciona usando el protoclo HTTP como base.
- **SMN**: Servicio Meteorológico Nacional.
- **CONAGUA**: Comisión Nacional del Agua.
- **INEGI**: Instituto Nacinal de Estadística y Geografía.
- **Políglota**: En área informática, que tiene soporte o comprende varios lenguajes de programación.
- **StakeHolder**: Personas u organizaciones que participan o tienen un interes en un proceso específico o definido dentro de un empresa. Se convierten en fuente clave de información para desarrollar un sistema o mejorar algún proceso.
- **SaaS**: Acrónimo de Software as a Service. Es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC).

Historias de Usuario

SmartOw1l

- US1 Estándarizar información de diferentes fuentes
  - **Como** usuario de SmartOwl
  - **Quiero** consultar la información climática actual
  - **De tal manera** que la información se obtenga con un estándar definido
- Criterios de Aceptación
  - Mostrar de qué región del país provienen los valores de las variables.
  - La información debe presentarse en formato JSON, XML y texto plano dependiendo del parámetro contentType que se envíe en la petición.

Friendly Dolphin

- US1 Reporte de información climática actual
  - **Como** usuario de Friendly Dolphin
  - **Quiero** consultar la información climática actual
  - **De tal manera** que pueda generar un reporte con la información estandarizada.
- Criterios de Aceptación
  - Seleccionar una región del pais.
  - Se deberá mostrar la información en texto plano.
  - Opción para exportar la información en formato JSON o XML.
- US2 Historial de información climática
  - **Como** usuario de Friendly Dolphin
  - **Quiero** consultar el historial de los datos climáticos.
  - **De tal manera** que pueda visualizar de manera gráfica el cambio de los valores de las variables climáticas a través de un periodo de tiempo
- Criterios de Aceptación
  - Consultar la información entre una fecha de Inicio y una fecha Final.
  - Se deberá mostrar la información de las variables climáticas en el periodo seleccionado.
  - Opción para exportar la información en formato JSON o XML.

Servicios REST

La ventaja al usar servicios de tipo REST, es la sencillez al cambiar el contenido que se expone sin tener que cambiar o generar un protocolo de comunicación ya que toma como base HTTP .

Simplemente es necesario con un cliente, desde un navegador web hasta clientes dedicados a servicios REST, para poder accedera a los recursos expuestos. Todas las API's rest siguen una convención de verbos tomados de la base de HTTP (GET, POST, PUT, DELETE).

En comparación con servicios de tipo SOAP, no se requiere una alta atomicidad y ni transacciones, una de las razones por la que los servicios WS suelen ser usados.

Finalmente, los servicios de tipo REST pueden tener respuestas en diversos formatos, principalmente JSON y XML, esto brinda al desarollador o a la persona que consula como tratar la respuesta por bibliotecas de terceros o nativas.

El soporte para formatos JSON es nativo en los navegadores web, otra de las razones por las cuales éstos servicios han ido ganando mercado ya que el desarrollo de aplicaciones Web ha aumentado de forma drástica en los últimos años. [31]

Fuentes externas de datos

Una de las fuentes usadas fue la de CONAGUA, institución de caracter público y mantenida por el Gobierno Federal, cuya misión es preservar las aguas nacionales y bienes públicos para su administración sustentable.

Una de las funciones de la CONAGUA es contar con los registros de clima e información ambiental de las fuentes que tiene distribuidas a lo largo del territorio nacional, éstas fuentes carecen de un formato definido lo cual complica en gran forma el análisis de datos.[32]

Para el desarrollo de la etapa final del proyecto se procedió a realizar un scrapping de la información expuesta por sus servicios y adaptarlos al modelo de datos propuesto por el equipo de trabajo.

```
Estacion:        NUEVA ROSITA,  COAH
Operada por:     SMN EMAS
Longitud:        101.19'48"  Latitud: 27.55'12"  Altitud: 366

DD/MM/AAAA HH:MM       DIRS       DIRR       VELS       VELR       TEMP
           HR          PB       PREC    RAD-SOL
17/11/2015 22:50         257        273      13.70      28.80       26.1
           10      963.6         0.0      136.0
17/11/2015 23:00         275        261      15.80      27.70       25.9
           10      963.6         0.0      103.0
17/11/2015 23:10         283        272      11.90      24.50       25.5
           11      963.7         0.0       72.0
17/11/2015 23:20         291        296      11.60      23.80       25.0
           11      963.9         0.0       45.0
17/11/2015 23:30         287        276      11.00      20.90       24.1
           12      964.0         0.0       24.0
17/11/2015 23:40         284        282       9.00      16.90       23.2
           13      964.1         0.0       12.0
17/11/2015 23:50         288        285      10.30      18.40       22.4
           14      964.2         0.0        3.0
18/11/2015 00:00         284        275       8.40      14.80       21.6
           15      964.3         0.0       -1.0
18/11/2015 00:10         292        293       9.00      15.10       21.2
           15      964.3         0.0       -1.0
18/11/2015 00:20         305        294       8.70      12.60       20.5
           16      964.4         0.0       -1.0
18/11/2015 00:30         322        322       9.30      15.10       19.8
           18      964.7         0.0       -1.0
18/11/2015 00:40         326        327       8.10      12.60       19.5
           18      964.9         0.0       -1.0
18/11/2015 00:50         236        316       4.60       9.40       17.9
           23      965.1         0.0       -1.0
18/11/2015 01:00          36         17       7.80      12.60       16.2
           27      965.3         0.0       -1.0
```

```
18/11/2015 01:10       130        62      4.50      7.90      15.4
           39     965.6       0.0      -1.0
18/11/2015 01:20       160       141      4.00      9.00      14.1
           38     965.8       0.0      -1.0
18/11/2015 01:30       163       176      4.70      6.80      13.1
           44     965.8       0.0      -1.0
18/11/2015 01:40       170       134      4.20      9.00      13.3
           52     965.9       0.0      -1.0
18/11/2015 01:50        34        30     12.70     19.80      13.0
           36     966.1       0.0      -1.0
18/11/2015 02:00        44        31      4.20     11.20      13.4
           40     966.1       0.0      -1.0
18/11/2015 02:10       105         5      4.30      5.80      11.6
           41     966.2       0.0      -1.0
18/11/2015 02:20       163        18      4.60      6.10      10.5
           42     966.3       0.0      -1.0
18/11/2015 02:30       111        16      4.50      6.50       9.9
           45     966.4       0.0      -1.0
18/11/2015 02:40       141       356      4.70      7.60       9.1
           51     966.6       0.0      -1.0
18/11/2015 02:50        23         7      5.80     10.40       9.0
           54     966.6       0.0      -1.0
18/11/2015 03:00       291       359     10.10     13.30       8.7
           60     966.9       0.0      -1.0
18/11/2015 03:10       333       352      3.90      9.00       9.0
           54     967.1       0.0      -1.0
18/11/2015 03:20        56        15      4.90      7.20       9.3
           49     967.1       0.0      -1.0
18/11/2015 03:30        27        39      5.90      6.80       8.7
           52     967.1       0.0      -1.0
18/11/2015 03:40        29        31      7.10      8.30       8.1
           54     967.1       0.0      -1.0
18/11/2015 03:50        24        12      7.00      9.70       7.8
           62     967.2       0.0      -1.0
18/11/2015 04:00        30        22      6.50      7.60       7.3
           60     967.2       0.0      -1.0
```

Instalación del gestor de documentos Mongo

Para sistemas basados en Ubuntu (12.04).

Es necesario agregar la llave de los servidores que tienen las fuentes de mongo a las llaves del sistema actual, eso se realiza mediante el siguiente comando:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

Después se procede a agregar los servidores de Mongo a la lista de distribución y actualizar los paquetes del sistema.

```
echo "deb http://repo.mongodb.org/apt/ubuntu precise/mongodb-org/3.0 multiverse"
| sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list

sudo apt-get update
```

Finalmente se procede a la instalación de mongo, eso se hace mediante la siguiente línea en la terminal.

```
sudo apt-get install -y mongodb-org
```

Al finalizar la instalación, se puede acceder al gestor mediante la siguiente línea

```
mongo
```

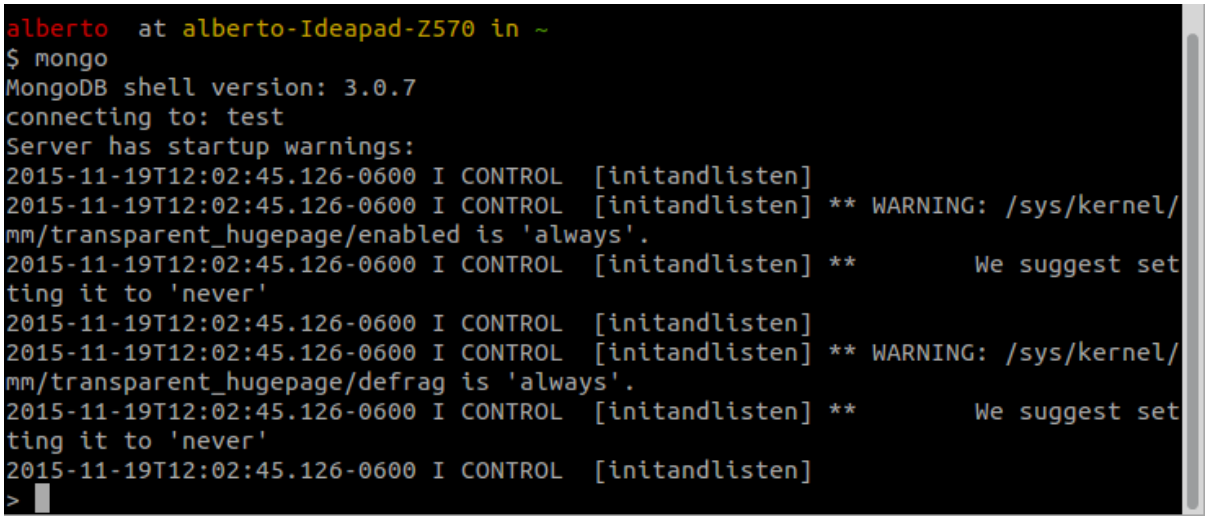


Figura 11.1: Captura del gestor de documentos Mongo.

Mostrando finalmente el acceso al gestor como lo muestra la siguiente imagen.

Ahora es necesario realizar la creación de la base de tipo Places (Fast Eagle), utilizando la siguiente línea en una terminal nueva.

```
mongoimport --db=Ambienta2MX-Places --jsonArray --collection=Places -v placesdb.j
```

El archivo placesdb.json se encuentra dentro del disco del proyecto.

Para poder utilizar los índices geográficos de mongo es necesario crearlos, para ello es necesario crear las bases de tipo MX en el servidor o servidores deseados y generar un índice de la siguiente forma.

```
db.<DesiredCollection>.createIndex(  <LocationField> : "2dsphere"  )
```

Cambiando los valores de "DesiredCollection" y "LocationField" por los valores en las colecciones de Places, Weather o Pollution en el primero y location en el último.

Instalación de Gradle

Para sistemas basados en Unix (Linux y Mac).

Para poder tener los módulos de Ambienta2MX corriendo es necesario contar con una versión de la máquina virtual de java en ejecución, o bien un JDK (Java Development Kit), curl, git o wget (véase la instalación de dichas bibliotecas en el sistema de su preferencia), a continuación se muestra :

```
curl -s http://get.sdkman.io | bash
```

Para poder instalar las versiones necesarias de Groovy, Vertx y Gradle se utilizarán los siguientes comandos.

```
sdk install gradle
```

:

Después de instalar gradle, se puede acceder a la carpeta de los fuentes, "HardAnt", "FastEagle" o "SmartOwl" y ejecutar el siguiente comando:

Mostrando finalmente el acceso al gestor como lo muestra la siguiente imagen.

```
gradle tasks
```

:

Esto bajará las bibliotecas a usar por el proyecto y mostrará todas las tareas que dependen de éste, como se muestra en la siguiente captura de pantalla.

Ahora es necesario realizar la creación de la base de tipo Places (Fast Eagle), utilizando la siguiente línea en una terminal nueva (véase Figura 10.2).

```
alberto  at alberto-Ideapad-Z570 in ~/Documents/Github/Ambienta2MX/HardAnt [Branch : master*, # of commits : 30
]
$ gradle tasks
:tasks

------------------------------------------------------------
All tasks runnable from root project
------------------------------------------------------------

Default tasks: assemble

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles classes 'main'.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles classes 'test'.
uploadArchives - Does a maven deploy of archives artifacts

Build Setup tasks
-----------------
wrapper - Generates Gradle wrapper files. [incubating]

Documentation tasks
-------------------
groovydoc - Generates Groovydoc API documentation for the main source code.
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
components - Displays the components produced by root project 'HardAnt'. [incubating]
dependencies - Displays all dependencies declared in root project 'HardAnt'.
dependencyInsight - Displays the insight into a specific dependency in root project 'HardAnt'.
help - Displays a help message.
model - Displays the configuration model of root project 'HardAnt'. [incubating]
projects - Displays the sub-projects of root project 'HardAnt'.
properties - Displays the properties of root project 'HardAnt'.
tasks - Displays the tasks runnable from root project 'HardAnt'.

IDE tasks
---------
cleanEclipse - Cleans all Eclipse files.
cleanIdea - Cleans IDEA project files (IML, IPR)
eclipse - Generates all Eclipse files.
idea - Generates IDEA project files (IML, IPR, IWS)

Verification tasks
------------------
check - Runs all checks.
Menu  ~/Documents/Gith...  Usage - sdkman - M...  /bin/bash  Ter
```

Figura 11.2: Tareas de Gradle en el proyecto Hard Ant.

Instalación de Nginx

Para sistemas basados en Unix (Linux y Mac).

La instalación se puede realizar desde el gestor de paquetes dependiendo la distribución a usar, el ejemplo actual se encuentra en un sistema de tipo Ubuntu en su versión 14.04.

Sólo basta con instalar Nginx desde la línea de comandos utilzando el siguiente comando

```
sudo apt-get install nginx -y
```

Esto bajará las bibliotecas, archivos fuente y dependencias necesarias para instalar Nginx, posteriormente se procede a la edición del archivo de configuración para generar la unión con los demás módulos de Ambienta2MX.

Se tendrá que modificar el archivo ubicado en "/etc/nginx/sites-enabled/default", agregando líneas o bien reemplazando el contenido del archivo con lo que se muestra a continuación:

```
upstream api_fasteagle {
 server    127.0.0.1:7777;
}
server {
 listen 80 default_server;
 listen [::]:80 default_server ipv6only=on;
 root /usr/share/nginx/html;
 index index.html index.htm;
 server_name localhost;

 location / {
  try_files $uri $uri/ =404;
 }

 location /api/fasteagle {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    rewrite ^/api/fasteagle/?(.*) /$1 break;
    proxy_pass http://api_fasteagle;
    proxy_redirect off;
  }
}
```

La configuración de Nginx puede ser replicada para cualquier servicio o servidor externo, utilizando la directiva upstream, que funciona como un proxy entre el servidor web y los servidores de aplicación.