Universidad de las Ciencias Informáticas

Facultad 1

Herramienta de mapas sin conexión para Nova

Autor:

Adackny Castillo Fernández

Tutores:

Msc. Yoandy Pérez Villazón

Ing. Yanet Cabeza Chávez

La Habana, junio 2017

¨Año 59 de la Revolución¨

Herramienta de mapas sin conexión para Nova

Declaración de autoría

Yo Adackny Castillo Fernández, con carnet de identidad 93052408180, declaro ser el único autor del presente Trabajo de Diploma y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo con carácter exclusivo. Para que así conste firmo la presente declaración jurada de autoría en La Habana a los _____ días del mes de _____ del año _____.

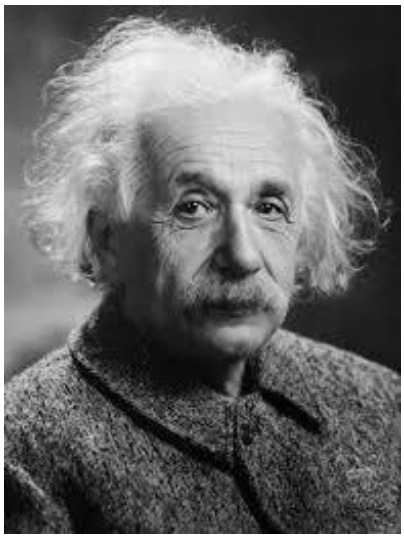
_____

Firma del autor

_____ _____

Firma del tutor Firma del tutor

Herramienta de mapas sin conexión para Nova



¨ *Somos arquitectos de nuestro propio destino.* ¨

Albert Einstein

Herramienta de mapas sin conexión para Nova

Dedicatoria

A mi madre y a mis dos hermanas Mery y Mariela. Espero que estén más que orgullosas por todo lo que he logrado y lo que me queda por lograr.

Herramienta de mapas sin conexión para Nova

Agradecimientos

Quiero agradecerle a dios, mi mayor guía en mi camino al éxito. A mi familia entera, especialmente a mis hermanas Mery y Mariela, mi sobrino, mi padre, mis tíos, a Shirley y Gena por estar siempre ahí para preocuparse por mí y darme el ánimo y las fuerzas para llegar a cumplir una de mis metas. A mis tutores (Yoandy y Yanet) y a Juan Manuel Fuentes, que me dieron su apoyo y depositaron su confianza en mí. También quiero agradecerles a todos los profesores que me han formado durante toda la carrera, ya que me brindaron sin lucro alguno un valioso tesoro, el cual es el amplio y alto conocimiento que ahora poseo. Por último y no menos importante, quiero agradecerles también a mis amistades y compañeros de aula. Han sido muy especiales estos 5 años compartidos.

Herramienta de mapas sin conexión para Nova

Resumen

Los servicios de georreferenciación brindan a los usuarios una manera interactiva de manipular los mapas geográficos, permitiéndoles agregarles información como lo son marcadores de sus lugares favoritos, rutas, conocer distancias entre lugares e incluso proveerles información topológica de áreas geográficas. Hoy en día casi todos los sistemas operativos cuentan con herramientas que brindan estos servicios y la mayoría de estos requieren usar *internet*. Actualmente, las herramientas que posee Nova de este tipo no satisfacen las necesidades de los usuarios, por lo cual se pone en una posición desventajosa con respecto al resto de los sistemas para los usuarios que frecuentan el uso de estos. Por tal motivo, en la presente investigación se propuso el desarrollo de una herramienta de mapas sin conexión para Nova. Para dar cumplimiento a la solución se hizo un estudio de los conceptos referentes a la georreferenciación y las tecnologías usadas para lograrlo, como lo son la librería *Leaflet* y el motor de visualización *web WebKit*. Además, se presentaron las características de dicha solución, así como su análisis y diseño, dando como resultado una aplicación híbrida definida por capas, que le brinda a los usuarios la facilidad de no usar conexión a *internet*. También se respetaron los estándares de codificación definidos para el desarrollo de la solución, implementándola usando las tecnologías web como *JavaScript* junto con las funcionalidades nativas del sistema mediante el lenguaje de programación *Python*. Luego se hicieron las pruebas funcionales y de aceptación, para finalmente, quedar realizado el despliegue de esta en los ordenadores que cuentan con Nova.

Palabras clave:

Aplicación híbrida, georreferenciación, mapa, sistema de información geográfica

Índice general

Índice de ilustraciones

Índice de tablas

Herramienta de mapas sin conexión para Nova

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) han permitido acercar a personas distantes a miles de kilómetros y han abarcado todas las áreas de la vida cotidiana del ser humano. Llegando a ser una herramienta fundamental en áreas como la medicina, la educación, la ingeniería, la geolocalización, e incluso el entretenimiento del ser humano, ampliando sus capacidades tanto físicas como intelectuales, y contribuyendo de esta manera al propio desarrollo de la sociedad.

Cuba ha optado por llevar las TIC a la sociedad con el objetivo de aumentar la calidad de desarrollo del país, así como lograr un mejor nivel de vida de sus ciudadanos (Elizalde, 2015). Con este objetivo y dada la incapacidad de seguir el ritmo de avance de las nuevas tecnologías y por los altos precios de las licencias en el mercado y de los contratos con empresas extranjeras, además del riesgo que conlleva el uso de tecnologías informáticas privativas para la seguridad nacional, se decidió fomentar el uso de *software* libre y de código abierto como una alternativa para informatizar la sociedad cubana.

Como parte de este proceso de migración a plataformas de código abierto, con el fin de lograr la soberanía tecnológica, se crea en la Universidad de las Ciencias Informáticas (UCI) la Distribución Cubana GNU/Linux Nova. Actualmente el continuo desarrollo y perfeccionamiento de esta distribución se lleva a cabo en el Centro de Soluciones Libres (CESOL).

Nova, en la actualidad es la distribución cubana recomendada por la comisión de informatización y ciberseguridad, según acuerdo de abril de 2015. Esta hace uso del núcleo Linux e incluye determinados paquetes de aplicaciones informáticas que permiten satisfacer las necesidades de la migración que se está llevando a cabo en Cuba como parte del proceso de informatización de la sociedad. Desde el 2010 existe una alianza de trabajo entre la empresa GEDEME del Grupo de la Electrónica y la UCI, el objetivo de la misma es la instalación masiva de Nova en todas las computadoras que se comercializan en el país por la industria nacional.

Herramienta de mapas sin conexión para Nova

Cuba, tanto en el sector estatal como en el particular presenta dificultades con el acceso a *internet*[1] para disfrutar de servicios como la compra y venta en línea, el uso de redes sociales y los servicios de georreferenciación como los que brindan *Google Maps*, *OpenStreetMaps*, *Mapbox*, *Waze* y *Apple Maps*. Los servicios de georreferenciación ayudan a los ciudadanos e instituciones a conocer distancias entre lugares, localizar y definir geográficamente puntos de interés, así como también la posibilidad de informarse acerca de servicios básicos que se le brindan a la población como rutas de ómnibus y conocer estadísticas o datos demográficos de una determinada región. Diversas aplicaciones proveen estos servicios y están disponibles para plataformas como *Android*, *Windows*, *iOS* y sistemas basados en GNU/Linux. El uso de estas aplicaciones se ha difundido en los últimos años debido a la creciente innovación y producción de las tecnologías móviles y dispositivos inteligentes, lo cual ha hecho posible explotar su utilidad, ganando mayor aceptación por una amplia variedad de usuarios de estas plataformas.

Con el avance de las telecomunicaciones, el acceso a *internet* por parte de la población mundial se ha incrementado, trayendo como resultado que muchos servicios pasen a estar de manera remota, lo cual es la causa de que la gran mayoría de los servivios de georreferenciación trabajen mediante la conexión a un servidor en línea. Nova, a pesar de traer por defecto una herramienta de georreferenciación, la cual es *Gnome Maps*, no puede satisfacer las necesidades de la mayoría de los clientes y usuarios cubanos por el hecho de que esta requiere también del uso de *internet* para poder trabajar. Esto puede causar que los usurios valoren a Nova en una posición desventajosa con respecto al resto de las plataformas en cuanto a este tema, y terminen recurriendo a otras para acceder a estos servicios, las cuales los tienen disponibles y trabajan de manera local; dígase por ejemplo, *OsmAnd*, disponible para *Android* e *iOS*. Por lo tanto se hace necesario brindar una alternativa más para los usuarios de Nova para acceder a servicios de georreferenciación. Dado que se espera que este sea la distrubución que cubra todas las necesidades de la población cubana en el proceso de migración a *software* libre y debido a que la mayoría de estas personas no cuentan con acceso a *internet*, se plantea el siguiente **problema de investigación:** ¿Cómo lograr que los usuarios de Nova puedan disponer de servicios de georreferenciación sin necesidad de conexión a *internet*?

[1] Internet: conjunto descentralizado de redes de comunicación interconectadas entre ellas.

Herramienta de mapas sin conexión para Nova

Se plantea como **objeto de estudio** las aplicaciones de georreferenciación en mapas sin conexión a *internet*, enmarcado en el **campo de acción**, las aplicaciones de GNU/Linux para georreferenciación en mapas sin conexión a *internet*.

Se define como **objetivo general:** Desarrollar una aplicación informática que permita la georreferenciación de elementos sin conexión a *internet* para Nova.

Para darle cumplimiento al objetivo planteado, se han proyectados los siguientes **objetivos específicos**:

1. Sistematizar sobre las herramientas y técnicas empleadas en los Sistemas de Información Geográfica (SIG) para la georreferenciación de datos sin necesidad de conexión a *internet*.
2. Analizar y diseñar una solución informática para Nova que permita la identificación, visualización y definición de elementos y trayectorias sobre un mapa cuyo funcionamiento se base en datos locales.
3. Implementar un sistema que permita la visualización, representación y consulta de datos georreferenciados en un mapa sin necesidad de conexión a *internet*.
4. Probar la solución para validar su correcto funcionamiento.

Se tiene como **idea a defender**, que la realización de una aplicación de georreferenciación en mapas sin conexión a *internet* disponible para Nova incrementará la accesibilidad y el uso de los servicios de georreferenciación en Cuba.

Los métodos de investigación utilizados fueron:

- El **análisis histórico-lógico:** Se emplea con el objetivo de verificar como evolucionaron teóricamente las aplicaciones de mapas desconectados, para de este modo poder seleccionar las técnicas, herramientas y algoritmos que se van a utilizar en el desarrollo de la herramienta de mapas sin conexión para Nova.
- El **inductivo-deductivo:** Permite llegar a conclusiones generales a partir de hechos que conforman la teoría, o a partir de dichas teorías arribar a conclusiones sobre casos particulares que se llevaron a cabo en la práctica, además de que permite plantear los objetivos principales para la realización de la herramienta de mapas sin conexión.

- La **modelación:** Es utilizado para la representación de los diagramas del diseño de la solución propuesta.
- La **observación:** es utilizada para obtener de forma directa la información de la realidad objetiva de la georreferenciación de forma local sin necesidad de la conexión a *internet* en el contexto de los sistemas operativos GNU/Linux.
- El **análisis documental:** es utilizado para la revisión de la bibliografía necesaria en el proceso de investigación de la implementación de aplicaciones de mapas que trabajan sin conección a *internet*.

El presente documento se encuentra estructurado de la forma siguiente:

En el **Capítulo 1** se abordan los principales aspectos teóricos que sustentan la investigación, se realiza un estudio de las técnicas y herramientas utilizadas en el proceso de creación de mapas informáticos que trabajan sin conexión. Se seleccionan las principales herramientas y tecnologías a utilizar.

En el **Capítulo 2** se realiza un análisis en correspondencia con las conclusiones del capítulo 1 que fundamenta la solución que se propone. Se presentan los diagramas, arquitectura de la herramienta, funcionalidades, diseño de clases y otros elementos del diseño de software que permiten una mejor comprensión de la solución.

En el **Capítulo 3** se muestran los estándares de codificación, los resultados de la valoración de la propuesta a partir de la simulación de varios entornos de prueba y mediante la aplicación de un método de consulta a expertos.

Herramienta de mapas sin conexión para Nova

Capítulo 1. Fundamentación Teórica

Introducción

A continuación, se hará un estudio de los conceptos asociados al tema de esta investigación y se analizarán algunas herramientas de mapas que permiten la georreferenciación sin conexión a *internet*. A partir de ahí se podrán elegir cuales son las características que la solución debe tener y para lograrlo se hará la elección de las herramientas, las tecnologías y la metodología de desarrollo que se empleará para su realización.

1.1 Principales conceptos y definiciones asociados al tema

1.1.1 Mapa

Un mapa es cualquier tipo de representación geográfica de algún territorio, en una superficie plana, una superficie bidimensional, tridimensional o esférica. Un mapa ayuda a las personas a ubicarse, como por ejemplo, cuando quieren saber cómo viajar de un lado a otro (trayectoria) o hacia dónde están viajando o donde están (Flor de loto, 2013).

Si bien es cierto, que los mapas han simplificado en gran escala diversas actividades humanas, dada al resumen de la búsqueda de un destino, también se han originado distintos tipos de Mapa, en los que se muestra la distribución de un sistema cableado o red de elementos que conforman una ruta, pasaje o trayectoria. Como por ejemplo, en una compañía hidrológica, aparte de un mapa topográfico que sirve para determinar la estabilidad y el tipo de suelos también usan mapas de red de tuberías y afluentes de ríos quebradas y yacimientos acuáticos, a fin de concretar una ruta para crear redes para el consumo humano (Ciencia M, 2011).

1.1.2 Tipos de mapas

De acuerdo a lo que se quiere representar existen diversos tipos de mapas, por ejemplo, hay algunos que representan el relieve de un lugar mientras otros representan divisiones territoriales, trayectoria de viajes etc. En todo caso, cualquiera sea el mapa siempre debe tener algunos elementos básicos (simbología, escala, título) para poder comprenderlo (Prado, 2012). Algunos de estos ejemplos son:

- **Mapas Físicos**

Estos mapas representan las características físicas del territorio, como por ejemplo el límite de costa, la hidrografía, los lagos, las curvas de nivel y el relieve. Se caracteriza por tener una simbología en colores donde va de la gama total de verdes hasta llegar a la gama total de colores café (Ciencia M, 2011).

- **Mapas Políticos**

  Estos mapas representan divisiones territoriales con fines administrativos o legales en donde los límites son de gran importancia y en cambio otras características, como relieve y la hidrografía son de importancia secundaria (Ciencia M, 2011).

- **Mapas Topográfico**

  Estos mapas representan de forma precisa los aspectos visibles del terreno, como alturas, depresiones, presencia de vegetación, de caminos o edificaciones. Además, ubican a los fenómenos en su real localización. Por ello, se dice que su función es fundamentalmente de posición (Ciencia M, 2011).

- **Mapas Histórico**

  Estos mapas representan hechos históricos determinados, los que pueden ayudar a comprender de mejor forma la historia. Pueden abarcar desde viajes de navegantes hasta la ubicación de pueblos antiguos (ejemplo: Mesopotamia, Egipto, Imperio Romano, la ciudad de Atenas) (Ciencia M, 2011).

Aunque el resultado de la presente investigación puede hacer uso de cualquier tipo de mapa, solo se utilizarán los mapas topográficos. Esto permitirá al usuario tener una mejor representación de las ciudades y poblados, pudiendo aprovechar mejor en este tipo de mapas los servicios de adicionarle rutas o marcadores de sitios de interés.

1.1.3 Geolocalización y georreferenciación

Debido a la semejanza que existe entre estos dos conceptos, se hace necesario aclarar brevemente que son y cuáles de ellos se usarán en la presente investigación.

Herramienta de mapas sin conexión para Nova

Geolocalización

La geolocalización es aquel proceso que se encarga de determinar la posición de algo en particular en la tierra; en otras palabras, la geolocalización alude al posicionamiento referente a la localización de un objeto ya sea animado o inanimado, que se presenta por medio de un vector o punto, en un sistema de coordenadas y *datum*[2] determinado. Este proceso se realiza generalmente en los sistemas de información geográfica. Entonces podemos decir que la geolocalización se encarga específicamente en obtener la localización de una persona, empresa, evento, ciudad o pueblo, en un punto geográfico exacto que es determinado por medio de ciertas coordenadas, usualmente provenientes de satélites, pero que cabe destacar que también pueden provenir de otros dispositivos como los móviles (Ciencia M, 2011). *Facebook Places, Twitter Places, Google Latitude y Waze* son ejemplos de herramientas que se utilizan para la geolocalización.

Georreferenciación

La georreferenciación es el uso de coordenadas del mapa para asignar una ubicación espacial a entidades cartográficas. Todos los elementos de una capa del mapa tienen una ubicación geográfica y una extensión específicas que permiten situarlos en la superficie de la Tierra o cerca de ella. La capacidad de localizar de manera precisa las entidades geográficas es fundamental tanto en la representación cartográfica como en los Sistemas de Información Geográfica **(Environmental Systems Research Institute Inc, 2016)**.

Debido a que la solución al problema planteado en esta investigación se caracteriza por ser una solución de georreferenciación, se muestran este concepto y el anterior, con el objetivo de evitar confusión a la hora

[2] *Datum*: en geodesia es un conjunto de puntos de referencia en la superficie terrestre con los cuales las medidas de la posición son tomadas y un modelo asociado de la forma de la tierra (elipsoide de referencia) para definir el sistema de coordenadas geográfico.

de referirse a ellos. Por eso, en esta investigación se hará uso solo del concepto de georreferenciación y se investigarán las aplicaciones de georreferenciación local.

1.1.4 Mapas digitales y servicios basados en la geolocalización

Se entiende por mapas digitales al conjunto de datos que representan información geoespacial y sus atributos. En los últimos años se ha visto cómo todo tipo de mapas digitales iban adquiriendo cada vez mayor protagonismo en la red. Un gran número de ayuntamientos han creado sus propios callejeros con información de sus municipios, equipamientos y servicios. Muchas *startups*[3] tienen en los mapas digitales y en los servicios basados en la localización su principal estrategia. Este es el caso de empresas de búsqueda de alojamientos turísticos, de transporte público, negocios locales, rutas museísticas y un largo etcétera.

Los servicios basados en la localización (*Location Based Services* o LBS) son generalmente aplicaciones que ofrecen servicios personalizados en tiempo real al usuario basándose en la localización geográfica de su dispositivo. El éxito y la aparición de una gran cantidad de LBS no es casual, sino que, como manifiestan diversos autores, las búsquedas basadas en la localización son una de las funcionalidades que más expectativas despiertan entre los usuarios de *internet*, especialmente entre los usuarios de dispositivos móviles.

La preponderancia de este tipo de aplicaciones y su uso en servicios de interés general y públicos implican la necesidad de diseñarlas o adaptarlas de manera que resulten accesibles para todo el mundo (Rubén Alcaraz Martínez, 2015).

1.1.5 Sistemas de Información Geográfica

Un Sistema de Información Geográfica (SIG) es una integración de información con herramientas informáticas cuyo principal objetivo es la obtención de datos relacionados con el espacio físico (el mundo real). Así pues, un SIG es un *software* específico que permite a los usuarios crear consultas interactivas, integrar, analizar y representar de una forma eficiente cualquier tipo de información geográfica referenciada asociada a un territorio, conectando mapas con bases de datos. El uso de este tipo de sistemas facilita la

[3] Se les llama así a las empresas emergentes apoyadas en la tecnología.

visualización de los datos obtenidos en un mapa con el fin de reflejar y relacionar fenómenos geográficos de cualquier tipo, desde mapas de carreteras hasta sistemas de identificación de parcelas agrícolas o de densidad de población. Además, permiten realizar las consultas y representar los resultados en entornos web y dispositivos móviles de un modo ágil e intuitivo, con el fin de resolver problemas complejos de planificación y gestión, conformándose como un valioso apoyo en la toma de decisiones (Confederación de Empresarios de Andalucía, 2010). La solución que se requiere para cumplir con el objetivo de esta investigación entra en la clasificación de SIG, por lo cual esta investigación se enfocará en las características que estos poseen. Ahora se verá una de las características fundamentales que distinguen a todo SIG de hoy.

1.1.6 Capas

Los SIG muestran los mapas y la información contenida en este a través de capas. Estas son básicamente una forma de mostrar múltiples niveles de información independientes cada uno. Es buena idea imaginarse a estas capas como hojas de papel transparente con ciertas representaciones cartográficas, las cuales se van apilando una arriba de la otra, hasta que la información que provee cada una de estas hojas queda superpuesta. Es necesario destacar que el orden en que se apilan estas capas es muy importante, ya que de seleccionarse mal podría ocurrir que la información que muestra una capa pueda ocultar la información que muestra la capa inferior a esta. Las capas no son un concepto de Cartografía en sí, pero son una técnica muy usada en los diseños de aplicaciones SIG. Estas aplicaciones deben tener al menos una capa, conocida como Capa Base (Hazzard, 2011). Esta capa se sitúa en la parte más baja de la pila de capas y esa posición es invariante para todas las aplicaciones SIG. El resto de las capas son las que poseen información como los marcadores, rutas y puntos turísticos y normalmente pueden dibujarse en cualquier orden entre ellas.

1.2 Estudio de soluciones existentes que trabajan sin conexión a *internet*

Para lograr un mejor entendimiento de cómo debería ser la solución que se le va a dar al problema de estudio se hace un estudio de las herramientas más notables en el contexto actual, así como también sus ventajas y desventajas. Esto proporcionará una idea bien clara de lo que realmente es útil y cuáles serían las características que convendría que la solución tuviera.

KDE Marble

Herramienta de mapas sin conexión para Nova

Es una aplicación de geodesia que permite mostrar mapas en 3 dimensiones (*3D*), desarrollada por KDE para el uso en las computadoras personales (PC) y teléfonos inteligentes. Soporta integración con mapas en línea como los de *OpenStreetMap* y tiene funcionalidades como buscar direcciones, crear mapas, medir distancias y recuperar información detallada de lugares de los cuales incluso sean poco conocidos. Soporta tanto la navegación en línea como fuera de línea. Está escrita en C++ y usa *Qt4*, por lo que, para ejecutarse, el sistema operativo debería poseer dicha librería (Rahn, et al., 2016). Esto es un inconveniente a la hora de distribuir Nova, dado que este deberá portar también dicha librería y por consecuencia requeriría más espacio en disco.

GOSM

Esta es un visor de mapas de *OpenStreetMap* escrito en el lenguaje de programación C usando *Gtk+* para sistemas basados en Linux. Ofrece características como visualizar mapas con *Mapnik/Osmarenderer*, accede a capas almacenadas de forma local. También es capaz de medir distancias entre puntos y longitudes de caminos, exportar los mapas en forma de grandes imágenes que los contienen y exporta además, documentos en formato *PDF*[4] de áreas seleccionadas. Este *software* a pesar de tener las características de trabajar con datos locales no es factible adoptarlo como solución dado que, aunque es de código abierto, su mantenimiento resultaría extremadamente costoso, sin contar también que sería prácticamente imposible adaptarlo a que brinde servicios que un determinado cliente necesite.

GpsDrive

GpsDrive es un sistema de navegación de autos (incluyendo bicicletas, barcos y aviones) el cual muestra la posición proveída por un receptor de *GPS* y la muestra en un mapa. Dichos mapas son autoseleccionados para una mejor resolución dependiendo de la posición en la que uno se encuentre y pueden ser descargados de *internet*. Esta aplicación está escrita en C y usa *Gtk+* bajo licencia GPL. Se encuentra disponible para Linux y Mac OSX. Como podemos observar, el uso de receptores GPS es algo que no viene por defecto con las computadoras de escritorio y la mayoría de las *laptops*, por lo que se requeriría agregarles dispositivos que realizaran dicha función.

[4] *Portable Document File.*

OsmAnd

Esta es una aplicación de mapas y de navegación por *GPS* que se ejecuta sobre los sistemas operativos Android e iOS para tabletas y teléfonos inteligentes. Es un proyecto de código abierto colaborativo bajo licencia GPL e incorpora en su totalidad datos libres de *OpenStreetMap*. Entre sus características están la navegación que trabaja lo mismo en línea que de manera local, enrutamiento y búsqueda de lugares por direcciones, por coordenadas geográficas y alrededor de una ruta definida por el usuario. También muestra la posición y dirección en la que se está en el mapa, puede mostrar mapas disponibles en servidores en línea e incluso mostrar las vistas de los satélites. Esta última aplicación no es compatible con Nova, por lo cual se llega a la conclusión de que independientemente de que se posea su código fuente, no sería viable utilizarla como propuesta de solución.

QGIS

Esta aplicación provee un rango amplio de funcionalidades tales como mostrar datos vectoriales y *raster* en varios formatos como JPEG, *GeoTIFF*, PNG, ERDAS IMG y *ArcInfo ASCII GRID*, crear, editar, gestionar y exportar datos y el análisis de datos. Está disponible para los sistemas operativos *Windows* y sistemas basados en GNU/Linux. Este es un proyecto libre de código abierto y aunque mantenerlo es posible, adaptarlo a las necesidades de los clientes cubanos sería un proceso costoso y tedioso debido a la excesiva cantidad de código fuente que este posee, ya que se está hablando de un proyecto muy maduro que empezó desde el año 2002. Otro aspecto en contra de una posible selección de este proyecto como una solución al problema planteado es que su complejidad de uso es alta y por lo tanto no es apta para todo tipo de usuario. También es importante mencionar que está desarrollada usando *Qt*, lo cual trae los inconvenientes ya mencionados con *KDE Marble*.

1.2.1 Valoraciones sobre estudio de las soluciones existentes

De las soluciones antes analizadas se puede concluir que no se usará ninguna de las aplicaciones anteriores como solución por los siguientes motivos:

- *KDE Marble* no posee integración con *Gtk*, lo cual traería una carga de más para Nova por tener que incluir además las librerías y dependencias de *Qt*, por lo cual no será usada como solución.

- *GOSM* es riesgoso adoptarlo como solución porque su mantenimiento y adaptación a las necesidades de los usuarios cubano resultaría más costoso que el desarrollo de un nuevo proyecto de este tipo.
- *GpsDrive* es un sistema que se usa principalmente para navegación y su uso en las computadoras personales y laptops hace que este pierda su verdadera utilidad. También hay que tener en cuenta que, de hacerlo, se estaría forzando a los usuarios que usan computadoras de escritorio a agregarle un dispositivo receptor de *GPS* a su ordenador.
- *OsmAnd* está disponible actualmente para sistemas *Android* e *iOS*, por lo que portar esta aplicación hacia Nova sería casi equivalente a desarrollar un proyecto desde cero.
- *QGIS* provee una amplia variedad de funcionalidades, pero su compleja interfaz de usuario hace que no sea apta para todos los usuarios. También trae el inconveniente de que está desarrollada con *Qt*, trayendo consigo las consecuencias mencionadas con *KDE Marble*.

Con respecto a las funcionalidades que exponen cada una de las aplicaciones anteriores se concluye que:

- Sería útil darle la posibilidad al usuario de trabajar de manera local, o sea, sin conexión a *internet*, ni a ningún tipo de servidor[5].
- La solución debería exponer servicios para que el usuario agregue marcadores en el mapa, así como rutas y la posibilidad de eliminar estos en caso deseado.
- Debería guardar los datos que el usuario le agregue al mapa, como lo son las rutas y marcadores que este adicione de manera persistente en el disco duro; además de actualizar los cambios que este les haga a dichos datos.

[5] Los servidores son equipos ubicados en algún lugar de *internet* u otro tipo de red que proveen servicios a aplicaciones de los usuarios que se conectan a estos. Generalmente son implementados vía *software* y están ubicados en una sola computadora remota, aunque en la actualidad se estila más la distribución de estos en varias computadoras mediante la arquitectura de *software* Microservicios.

- Es importante que posea una interfaz gráfica sencilla, ya que esta herramienta va dirigida a la población cubana en su totalidad, por lo que es conveniente que hasta un usuario no experimentado en el uso de los SIG pueda trabajar con ella.
- La portabilidad, un tema no menos importante que ningún otro dentro del desarrollo de *software*, y por el cual se están haciendo muchos esfuerzos por parte de todos los desarrolladores de *software* a nivel mundial en estos días, de plena Era de la Información, se debe tener en cuenta en esta solución, por el hecho de que en estos momentos se planea su uso para Nova, pero en los días venideros, es muy probable que se quiera portar hacia otras plataformas.

1.3 Tipos de aplicaciones

En esta sección se trata el último punto que se mencionó en las características que se decidió que debería tener la solución al problema de investigación. Por lo que se habla de los tipos de aplicaciones que se desarrollan hoy en día en el panorama global, ya sean las aplicaciones nativas, las cuales usan directamente el sistema operativo[6], las aplicaciones *web*, o las aplicaciones híbridas que se están haciendo populares en estos días donde la portabilidad es un tema crítico. Finalmente se decidirá por cual enfoque de los tres antes mencionados se optará.

Aplicaciones nativas

Las aplicaciones nativas, en principio, se desarrollan utilizando el lenguaje específico para cada plataforma, por ejemplo, *Java* en *Android*, *Objective-C* y *Swift* en *iOS* o *C#* y *Visual Basic* en *Windows*. Esto nos permite acceder a las *APIs* (Interfaz de Programación de Aplicaciones, por sus siglas en inglés) disponibles. Dado lo anterior mencionado, las aplicaciones nativas son desarrolladas única y exclusivamente para la plataforma que provee la *API.* Estas ofrecen el mejor rendimiento de todos los tipos de aplicaciones existentes, pero con el coste de perder todas las capacidades de portabilidad hacia otras nuevas plataformas (Collado, 2016).

Estas aplicaciones no necesariamente se compilan al código binario de la máquina física donde se van a ejecutar, díganse, por ejemplo, las aplicaciones de *Java* o las desarrolladas en *.Net* o incluso las de *Python*.

[6] Estas también incluyen a las aplicaciones que se ejecutan sobre intérpretes, como las desarrolladas en *Java* o *Python*.

Su clasificación se basa en que tienen soporte para utilizar las características particulares del sistema operativo donde se ejecutan. En realidad, este tipo de aplicaciones son nativas de la plataforma para las que fueron desarrolladas, no del sistema operativo donde se ejecutan. Así las aplicaciones de *Java* son aplicaciones nativas de la JVM[7], las que se crean para *Windows* usando C#, por ejemplo, son aplicaciones nativas de la CLR[8] de *.Net*, no de *Windows* en sí y las que se desarrollan para Android no son en realidad nativas de dicho sistema operativo, sino de la ART[9]. Esto es debido a que estas plataformas tienen un conjunto de instrucciones propio y administran tanto las aplicaciones que se ejecutan sobre ella como su memoria y operaciones de entrada y salida, abstrayendo al programador de estas funciones del sistema operativo real, quedando para este como un sistema operativo de alto nivel, del cual sus llamadas al sistema serían todas las funciones que dicha plataforma provee.

Aplicaciones *web*

Son aplicaciones desarrolladas con tecnologías *web* con el objetivo de ser visualizadas y utilizadas en cualquier equipo de cómputo que contenga un navegador. Es una aplicación de *software* que se codifica en un lenguaje soportado por los navegadores *web* en la que se confía la ejecución a dicho navegador. Toda la información se guarda de forma permanente en grandes servidores de *internet* y estos envían a los dispositivos o equipos conectados a ellos los datos que requieren en ese momento. Este tipo de aplicación es muy popular en el contexto actual debido a la independencia del Sistema Operativo que estas proveen, lo cual hace que esta se pueda ejecutar en cualquier plataforma que esté conectada a *internet* (GCF Community Foundation International , 2016).

Aplicaciones híbridas

Una aplicación híbrida se desarrolla utilizando las tecnologías y lenguajes de la *web* como por ejemplo *HTML, JavaScript* y *CSS*, para luego ser empaquetada y encapsulada dentro de una aplicación nativa que se encarga de utilizar el motor de interpretación *web* para que su contenido sea mostrado y utilizado. Se puede entender a este tipo de aplicaciones básicamente como un pequeño sitio *web* ejecutándose

[7] Siglas de *Java Virtual Machine*.

[8] Siglas de *Common Language Runtime*.

[9] Siglas de *Android Runtime*.

localmente en un visor *web* dentro de una aplicación. Estas ofrecen un rendimiento ligeramente inferior al de una aplicación nativa pero muy superior al de una aplicación *web*. Además, dado que son desarrolladas usando tecnologías *web*, tienen un pobre soporte de funciones nativas del sistema, por lo cual hay que proveérselo a estas a través del código nativo que se encarga de encapsularla y mostrarla mediante el visor *web*, usando mecanismos de comunicación entre dicho código nativo y el código *JavaScript* que está en la componente *web* (Malkani, 2016).

Este tipo de aplicaciones ha impactado en los últimos años, especialmente en el desarrollo de aplicaciones móviles, debido a la creciente necesidad de lograr una mayor portabilidad hacia todos los dispositivos que existen y los nuevos que se incorporan al mercado, todo ello con el mínimo coste de desarrollo posible. A continuación, se muestran algunas de las características de los enfoques antes mencionados (enfoques *web*, nativo e híbrido) (Departamento de Hacienda y Finanzas, 2012).

Ilustración 1 - Características de las aplicaciones web, híbridas y nativas.

Como se puede observar, las aplicaciones híbridas, a pesar de su peor desempeño que las aplicaciones nativas, nos ofrecen todas las ventajas de las aplicaciones *web*. Para mitigar la carencia de soporte nativo,

estas se comunican con la parte nativa de la aplicación, como se mencionó anteriormente, y esta es la encargada de ejecutar las tareas de bajo nivel que la parte *web* requiera. Esto permite delegarle tareas a la componente nativa que computacionalmente serían muy costosas de realizar utilizando el código interpretado *JavaScript*. Finalmente, basándose en los argumentos anteriores, se decide optar por el enfoque híbrido, lo cual permitirá desarrollar una aplicación con un costo bajo en poco tiempo, logrando, además, la facilidad de portar esta hacia otras plataformas si se requiere en el futuro, por lo que se satisface el último punto de las valoraciones sobre el estudio de las soluciones existentes, el cual es el de la portabilidad.

1.4 Herramientas, metodología y tecnologías que se van a utilizar en este trabajo

Seguidamente en este apartado vamos a ver las herramientas necesarias para lograr la realización de la solución propuesta.

1.4.1 Metodología de desarrollo a utilizar

El proceso de desarrollo de la solución hace uso de la metodología AUP-UCI. Dado que esta es la metodología definida por la Universidad de las Ciencias Informáticas para el desarrollo de sus proyectos, es innecesario hacer un estudio de otras metodologías, por lo que esta investigación se centra directamente en las características de ella solamente.

AUP-UCI es una variación (o adaptación, como se prefiera llamarle) de la metodología de desarrollo AUP[10]. Esta está compuesta por las siguientes fases (Rodríguez Sánchez, 2015):

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante

[10] *Agile Unified Process*.

el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.4.2 Lenguajes de programación

Es un hecho que el lenguaje de programación es la herramienta básica de cualquier desarrollador de *software*. Estos son los medios que permiten crear *software* de cualquier tipo, incluso crear otros lenguajes de programación en sí; también son el idioma con el que se le indica a una computadora[11] lo que ella debe hacer. Los lenguajes a usar en esta investigación son *JavaScript*, *HTML*, *CSS* y *Python*, por lo que a seguidamente se habla de ellos y del uso que tienen en este trabajo.

JavaScript v5

Este es un lenguaje interpretado, publicado por *ECMA International*[12]. Este se define como orientado a objetos, basado en prototipos[13], imperativo, de tipificado débil y dinámico. Su uso es principalmente en las aplicaciones web del lado del cliente, por lo cual los navegadores vienen con motores que interpretan este lenguaje. JavaScript es un lenguaje interpretado, por lo que no hace falta compilar su código para poder usarlo, es decir se puede ejecutar directamente en cualquier navegador web sin necesidad de procesos intermedios. Se escoge la versión 5 de este lenguaje (*ECMAScript* 5) para el desarrollo de la solución de la presente investigación, porque es la que todos los navegadores soportan completamente en el momento. Su uso en la solución del problema de esta investigación es muy crucial debido a que es el lenguaje con

[11] Debe entenderse por computadora a los ordenadores personales en sí, teléfonos celulares, relojes inteligentes, en fin, cualquier cosa que su funcionamiento recaiga en el uso de una memoria principal, una unidad de procesos central y puertos de entrada y salida para comunicarse con el exterior.

[12] Organización internacional basada en membresías de estándares para la comunicación y la información.

[13] Es un estilo de programación en el cual los objetos no son creados mediante instanciación de clases, sino, mediante la clonación de otros objetos o mediante la escritura de código por parte del programador.

que se implementa la lógica de las aplicaciones *web* del lado del cliente, va a servir en este trabajo para implementar la lógica de los componentes desarrollados con tecnologías *web* en la solución.

Python 2.7

Python es un lenguaje de programación desarrollado como proyecto de código abierto y es administrado por la empresa *Python Software Foundation*. Se trata de un lenguaje de programación de scripts, que permite dividir el programa en módulos reutilizables desde otros programas *Python*. También viene con una gran colección de módulos estándar que proporcionan entrada y salida de ficheros, llamadas al sistema, sockets e interfaces gráficas de usuario. Se trata de un lenguaje interpretado, lo que permite ahorrar el proceso de compilado (Alegsa, 2016). Su uso se hace necesario para acceder a las funcionalidades que nos provee Gtk+. Dado que el tipo de aplicación a desarrollar, como ya se había mencionado, es una aplicación híbrida, todos los componentes nativos de esta serán desarrollados usando este lenguaje. Es seleccionado este y no otro lenguaje como C, por ejemplo, por su sencillez, posee un código fuente legible y claro y además facilita el fácil mantenimiento de la aplicación en el futuro.

HTML5

HTML es un lenguaje de marcado de hipertexto (como sus siglas en inglés indican: *HiperText Markup Language*) que se usa para la elaboración de páginas web. Es un estándar definido por el *W3C*[14] y se considera el lenguaje web más importante siendo su invención muy crucial en la aparición, desarrollo y expansión de la *World Wide Web*[15] (*WWW*). Este lenguaje dispone de ciertas ventajas como por ejemplo que es interpretado, por lo que no es necesario compilarlo y todos los navegadores web lo soportan por defecto, además de que se usa para representar los elementos visuales de una página web, este posee un formato que es entendible por el ser humano, lo cual facilita su edición y aumenta la productividad de los desarrolladores. Posiblemente sea el lenguaje más común debido a la gran cantidad de páginas web que existen hoy en día en la *internet.* En este trabajo se usará la última versión llamada *HTML5*, la cual posee nuevas características como la definición de nuevas etiquetas para mostrar los contenidos multimedia, la

[14] *World Wide Web Consortium.*

[15] También conocida como *internet* o la red de redes.

modificación de la etiqueta <input>, etiquetas semánticas para remplazar el uso extensivo de etiquetas <*div*>, el objeto *Canvas* y muchas otras características más. Es totalmente necesario este lenguaje dado que es el que se usa para representar lo que se quiere mostrar en una aplicación web.

CSS3

Esta es la última versión del lenguaje de hojas de estilos en cascadas *CSS* que amplía su anterior versión *CSS2.1*. Trae nuevas características como las esquinas redondeadas del modelo de cajas del diseño web, sombras, gradientes, y una de las más interesantes, las transiciones o animaciones. Su uso en la implementación de la solución propuesta es mínimo[16], por lo que se mencionará muy poco en este trabajo.

Conociendo los lenguajes de programación necesarios para implementar una herramienta de mapas híbrida[17] que trabaje sin conexión, es necesario también alguna librería que nos ayude a gestionar y visualizar las capas de los mapas. En la siguiente sección se mencionan las dos librerías más populares para la creación de este tipo de aplicaciones en el ámbito de la *web*.

1.4.3 Librería a utilizar

OpenLayers 3

OpenLayers es una librería escrita en JavaScript para visualizar mapas basados en la web, ofreciendo todos los componentes necesarios para trabajar con información geográfica en el lado del navegador. En esta versión el renderizador[18] fue actualizado para soportar *Canvas*, *WebGL* o elementos DOM, dependiendo de las capacidades del navegador. Además, el tamaño en disco de la librería ha sido reducido drásticamente con respecto a sus versiones anteriores (Santiago, 2015).

Leaflet v1.0.3

[16] Incluso se podría prescindir de él para la implementación sin que el sistema deje de funcionar o pierda apariencia visual.

[17] Véanse los tipos de aplicaciones.

[18] Componente de software, dado el contexto, que se encarga de dibujar en la pantalla.

Herramienta de mapas sin conexión para Nova

Leaflet es también una librería escrita en *JavaScript* y de código abierto que se centra en un conjunto básico de características, de este modo pretende mantenerse lo más ligera posible (33KB de tamaño). Se basa en la simplicidad, rendimiento y usabilidad (Morales, 2016). Aunque es pequeña, posee casi todas las características y funcionalidades necesarias para trabajar con mapas basados en la web, dado que esta fue hecha para crear mapas interactivos basados en la web. Tiene soporte para complementos, lo que permite adicionarle funcionalidades nuevas que no estén en el núcleo de esta. A continuación, se muestra una tabla comparativa entre *Leaflet* y *OpenLayers*:

Tabla 1 - Comparación entre Leaflet y OpenLayers (Morales, 2016).

	Leaflet	*OpenLayers*
Documentación disponible	Poca ,aunque la documentación oficial (*www.leafletjs.com/reference-1.0.3.html*) es suficiente para conocer todo lo que provee esta librería.	Mucha
Herramientas de edición	No	Si
Funcionalidades disponibles	Solo las básicas. Evolutiva a través de complementos.	Es un proyecto más maduro, por lo cual es abundante en funcionalidades.
Complejidad de uso.	Poca	Mucha
Curva de aprendizaje	Baja	Alta
Tamaño en disco	33 KB	3.08 MB

Dado los datos anteriores, para la solución de esta investigación se valora elegir *Leaflet* como librería para implementar los servicios de georreferenciación y mostrado de mapas, basándose en que esta es más sencilla de utilizar y su curva de aprendizaje es más baja que la de *OpenLayers*. Considerando también el hecho de que su núcleo solo posee las funcionalidades necesarias para mostrar mapas e interactuar con el usuario, a diferencia de *OpenLayers*, esta nos evitara tener código en memoria de más, pudiéndole agregar algún complemento en caso de necesitar alguna funcionalidad más avanzada.

1.4.4 Herramienta de modelado

Se hará uso de la herramienta CASE[19] *Visual Paradigm Enterprise Edition* v8.0 para el proceso de modelado de los diagramas del diseño. Esto es debido a que es un *software* libre, es de alta eficiencia y permite realizar ingeniería tanto directa como inversa, muy utilizada en los proyectos de la UCI y el autor de la presente investigación está muy familiarizado con ella. Como consecuencia se usará también el lenguaje de modelado UML[20], un lenguaje visual para ilustrar las ideas del diseño. Se usará para modelar los diagramas de paquetes, los diagramas de clases del diseño y diagramas de componentes según se haga necesario.

1.4.5 Editor de texto

Para el desarrollo de la solución del problema de investigación se hará uso del editor de texto *Geany*, de uso libre y disponible para sistemas GNU/Linux. Esta herramienta posee características como lo son el resaltado de sintaxis, el autocompletado de código, y una consola integrada para *Python*. Soporta una gran variedad de lenguajes de programación, entre los cuales se encuentran *HTML*, *JavaScript*, *CSS* y *Python*, lo cual será de mucha para este trabajo. Esto permitirá el desarrollo en una sola herramienta, sin importar que se estén desarrollando dos componentes independientes cada uno, que luego se integrarán formando una sola aplicación[21]: el contenedor nativo que se desarrollará en *Python* y el contenedor *web* que se desarrollará usando *HTML*, *CSS* y *JavaScript*.

Valoraciones parciales del capítulo 1

A partir del análisis realizado en el presente capítulo valoramos que el estudio de todos los conceptos asociados al objeto de estudio ha permitido llegar a un mejor nivel de comprensión del problema a resolver. Se pudo analizar las herramientas y tecnologías existentes para así seleccionar las que mejor se acomodaron a la posible solución del problema. No es factible seleccionar ninguna de las herramientas

[19] Ingeniería de *Software* Asistida por Computadoras (por sus siglas en inglés).

[20] Lenguaje Unificado de Modelado, por sus siglas en inglés.

[21] Es importante recordar que se decidió desarrollar una aplicación híbrida, la cual usa en conjunción tecnologías nativas del sistema operativo con tecnologías *web*. Véase la sección de los tipos de aplicaciones.

estudiadas dado las características que ellas presentan, las cuales algunas no resuelven el problema y otras dejan nuevos problemas abiertos, como lo son problemas de mantenimiento.

Capítulo 2. Análisis y Diseño

Introducción

En este capítulo se presentan las soluciones correspondientes al diseño de la aplicación a desarrollar y se selecciona la arquitectura a partir de la cual se va a desarrollar el *software*. Estas abarcan desde los requisitos funcionales y los no funcionales hasta los diagramas de clases correspondientes, incluyendo además los diagramas de colaboración, el diagrama de paquetes y los de casos de usos.

2.1 Propuesta de Solución

En este trabajo se propone como solución una herramienta de mapas que funcione de manera local (sin necesidad de ningún dato proveniente de *internet*) que brinde al usuario interactividad, tales como las opciones de agregar marcadores de sus lugares favoritos y rutas específicas, así como brindarle la posibilidad de eliminaros o modificarlos. También debe incluir la característica de guardar los datos que el usuario haya agregado o cualquier cambio que haya hecho en estos; como también debe tener la característica de cargar los datos guardados cada vez que esta inicie. Dicha herramienta deberá poseer los datos necesarios y mapas de manera local. Será desarrollada mediante un enfoque híbrido usando tanto tecnologías *web* como nativas del sistema operativo (en este caso, la distribución de Nova). Esto nos dará mayor portabilidad entre plataformas, pudiéndose llevar la solución hacia otros sistemas operativos en el futuro. Su código fuente tendrá dos separaciones lógicas, las cuales son el contenedor nativo que provee soporte para el acceso a funcionalidades nativas del sistema operativo y la otra es el contenedor *web*, que contiene la parte de la aplicación que será desarrollada con tecnologías *web* y la cual será la que se pueda portar hacia otras plataformas si se necesitara en un futuro. Se hará uso del motor de visualización *web Webkit* para interpretar y mostrar el contenido de la aplicación, por lo cual requerirá del acceso a las capacidades de la librería *Gtk* para el uso de la ventana donde se mostrará la instancia de la herramienta en ejecución. Todo esto evitándole al usuario de utilizar algún navegador web[22], trabajando como una

[22] Aplicación que se usa para interpretar el contenido de las páginas web, renderizarlas y comunicarse con servidores remotos para enviar y recibir datos asociados a estas páginas.

aplicación de escritorio. Además de utilizar la librería *Leaflet*, la cual le proveerá las funcionalidades a la aplicación para mostrar dicho mapa, crear capas nuevas, agregarles marcadores y rutas.
Conociendo lo que se va a desarrollar, es buen momento para pasar a extraer los requisitos funcionales del sistema.

2.2 Requisitos funcionales

Los requisitos funcionales son enunciados acerca de las funcionalidades que el sistema debe proveer y de cómo debería reaccionar ante ciertas entradas o eventos particulares (Sommerville, 2011). Estos se pueden clasificar como requisitos de usuario o requisitos del sistema. Este apartado se centra solamente en los requisitos del sistema dado que son los que brindan más detalles acerca de la funcionalidad que va a brindar la aplicación.
Para que la solución final cumpla con los objetivos se definieron los siguientes requisitos funcionales del sistema:

RF1 Mostrar el mapa.
RF2 Marcar puntos en el mapa.
RF3 Editar puntos del mapa.
RF4 Eliminar puntos del mapa.
RF5 Trazar rutas en el mapa.
RF6 Editar rutas en el mapa.
RF7 Eliminar rutas del mapa.
RF8 Insertar un nuevo nombre a algún punto.
RF9 Insertar un nuevo nombre a alguna ruta.
RF10 Centrar la vista del mapa en un punto del mapa visible.
RF11 Guardar los datos de los marcadores y rutas en un fichero cuando la aplicación se vaya a cerrar.
RF12 Cargar los datos de los marcadores y rutas almacenados previamente en un fichero cuando la aplicación se inicia.

2.3 Requisitos no funcionales

Los requisitos no funcionales expresan restricciones sobre la implementación del sistema. Pueden relacionarse con propiedades emergentes de dicho sistema tales como la fiabilidad, su tiempo de respuesta o el uso de algún tipo de tecnología en particular. Los requisitos funcionales se suelen aplicar al sistema como un todo, en lugar de a características individuales del sistema (Sommerville, 2011). Estos se suelen dividir en diferentes tipos de acuerdo al tipo de restricción que plantean (solo se mencionan los del tipo que se usan en esta propuesta de solución):

- Requisitos de usabilidad.
- Requisitos de desarrollo.

A continuación, se describen los requisitos no funcionales utilizados en la solución propuesta:

Requisitos de usabilidad

RNF1 La aplicación va dirigida principalmente a usuarios inexpertos, por lo que debe contar con pocas funcionalidades, evitando así frustrar el uso de esta aplicación a este tipo de usuarios.

RNF2 La aplicación debe de ser minimalista en cuanto a apariencia.

Requisitos de desarrollo

RNF1 Se necesita el uso de la biblioteca de JavaScript *Leaflet*.

RNF2 Se deben usar los lenguajes de programación *JavaScript*, *HTML5* y *CSS3* para la interfaz de usuario encargada de mostrar el mapa.

RNF2 Debe usarse el lenguaje de programación *Python* para acceder a las funcionalidades que brindan *Webkit* y *Gtk*.

RNF4 Se necesita el uso de *Webkit* como motor de renderizado de páginas *web.*

RNF5 La aplicación debe de ser capaz de ejecutarse en la distribución cubana de GNU/Linux Nova usando el entorno de escritorio *Gnome*.

2.4 Historias de usuario

Tabla 2 - Historia de usuario 1.

Historia de Usuario	
Número: HU_1	**Nombre del requisito:** Mostrar el mapa de Cuba.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 1

Herramienta de mapas sin conexión para Nova

Prioridad: Alta	**Tiempo estimado:** 8 horas
Riesgo en desarrollo: Alto	**Tiempo real:** 5 horas

Descripción:

Cuando la aplicación se ejecuta se muestra en la ventana principal el mapa a partir de ficheros locales con las imágenes que conforman la capa base. Para ello debe mostrarse un panel, el cual será el área de dibujo de dicho mapa. Dicho panel debe ocupar un área amplia de la interfaz del usuario para que esta le proporcione una mejor visibilidad al usuario. En el área de dibujo del mapa se mostrarán los controles para adicionar y editar los marcadores y rutas en la parte superior derecha del mapa. También se mostraran los controles de alejar y acercar la vista del mapa (*zoom*) en la parte superior izquierda.

Observaciones:

Prototipo de Interfaz:

Herramienta de mapas sin conexión para Nova

Tabla 3 - Historia de usuario 2.

Historia de Usuario	
Número: HU_2	**Nombre del requisito:** Marcar puntos en el mapa.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 5 horas
Riesgo en desarrollo: Alto	**Tiempo real:** 4 horas
Descripción: Cuando el usuario presiona sobre el botón con la imagen de un marcador en la parte superior derecha del panel del mapa, se le muestra el puntero del ratón con un marcador, el cual puede mover y colocar en donde considere oportuno.	
Observaciones: Puede cancelar la acción del botón de colocar marcadores pulsando sobre el botón que dice “cancelar” que aparece al lado de este. Si el usuario presiona sobre otro botón antes de colocar el marcador, se le deshabilitará colocar este, y se le llevará directo a la acción que realiza ese otro botón.	
Prototipo de Interfaz:	

Tabla 4 - Historia de usuario 5.

Historia de Usuario	
Número: HU_5	**Nombre del requisito:** Trazar rutas en el mapa.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 5 horas
Riesgo en desarrollo: Alto	**Tiempo real:** 5 horas
Descripción: Cuando el usuario presiona sobre el botón de la esquina superior derecha con la imagen de un segmento de recta, se le habilitara a colocar una ruta. El usuario deberá marcar los puntos de uno en uno, uniéndose los últimos dos	

puntos marcados mediante una línea recta. Para terminar de marcar la ruta, este deberá presionar de los tres botones que aparecen al lado del botón de marcar rutas, el que dice “terminar”.

Observaciones:

Si el usuario decide deshacer todo lo que ha hecho a partir de que haya presionado el botón de marcar rutas, este deberá presionar el botón que dice “cancelar” de los tres que aparecen al lado del botón de rutas. Otra forma de cancelar cuando este marcando alguna ruta, es presionando sobre el botón de marcador, el de edición o el de borrar, lo cual lo llevará a la acción que realizan esos botones.

El usuario puede eliminar el último punto que marcó de la ruta presionando sobre el botón que dice “borrar último punto” de los tres botones que aparecen al lado del de ruta.

Prototipo de Interfaz:

El resto de las historias de usuario se describen en el Anexo 1 por motivos de limpieza y estética relacionada con este capítulo.

2.5 Diseño

El diseño de *software* brinda la comprensión de este antes de su implementación, lo cual permite que se haga una representación formal del proyecto final, con la cantidad de detalles necesarios para lograr su realización y evitar confusiones entre los desarrolladores. Para ello se parte del diseño de la arquitectura que satisface los requisitos a implementar, para luego ir refinándola hasta los diagramas de paquetes, de clases e interacción.

2.5.1 Arquitectura seleccionada

La arquitectura de *software* se puede ver como una estructura de alto nivel que agrupa el sistema completo. Esta especifica cómo deben encajar juntas las piezas para construir la solución. La decisión tomada durante el proceso de creación de la arquitectura es muy fundamental para el sistema porque ellas son las bases sobre las que se van a tomar las decisiones futuras (Hanmer, 2013). También, según (Pressman, 2010 pág. 207), la arquitectura del *software* de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del *software*, sus propiedades externas visibles y las relaciones entre ellos.

Dado que se utiliza un enfoque híbrido para desarrollar la aplicación[23], la arquitectura del sistema se diseña usando el patrón arquitectónico N-capas. Este patrón arquitectónico divide el sistema en capas, interactuando en él, cada capa con la capa inferior a ella solamente (Hanmer, 2013). Por lo que queda la solución propuesta dividiéndose el sistema en dos componentes fundamentales, los cuales contienen la parte nativa (nombrado *Native_Container*) y la parte desarrollada con tecnologías web (nombrado *Web_Container*) respectivamente. El componente contenedor de la parte nativa está dividido a su vez en capas también. En este caso se cuentan con tres capas, las cuales forman el núcleo de las funcionalidades de la herramienta de mapas. Dichas capas son: la capa de presentación (*Presentation*), la de la lógica del negocio (*Logic*) y la de acceso a datos (*Data*). En la solución de esta investigación no se hizo necesario acceder a servicios nativos del sistema operativo desde el código escrito en *JavaScript*. En caso que sea necesario hacer estas llamadas hacia abajo en un futuro se podría hacer fácilmente. Solo basta con implementar un componente que brinde dichos servicios desde la capa externa *Native_Container* hacia las

[23] Véase el capítulo 1 para introducirse en el tema de las aplicaciones híbridas.

capas internas que están dentro de la capa externa *Web_Container*. Dicho componente será visto como una capa interna adicional que tendría el modelo formado por las capas *Presentation*, *Logic* y *Data* respectivamente. La capa *Presentation*, a su vez, incluye el paquete que contiene la librería *Leaflet* en su interior.

Para ver mejor como podrían quedar estructurados estos componentes se muestra a continuación una imagen con el diagrama de paquetes del diseño de la solución:

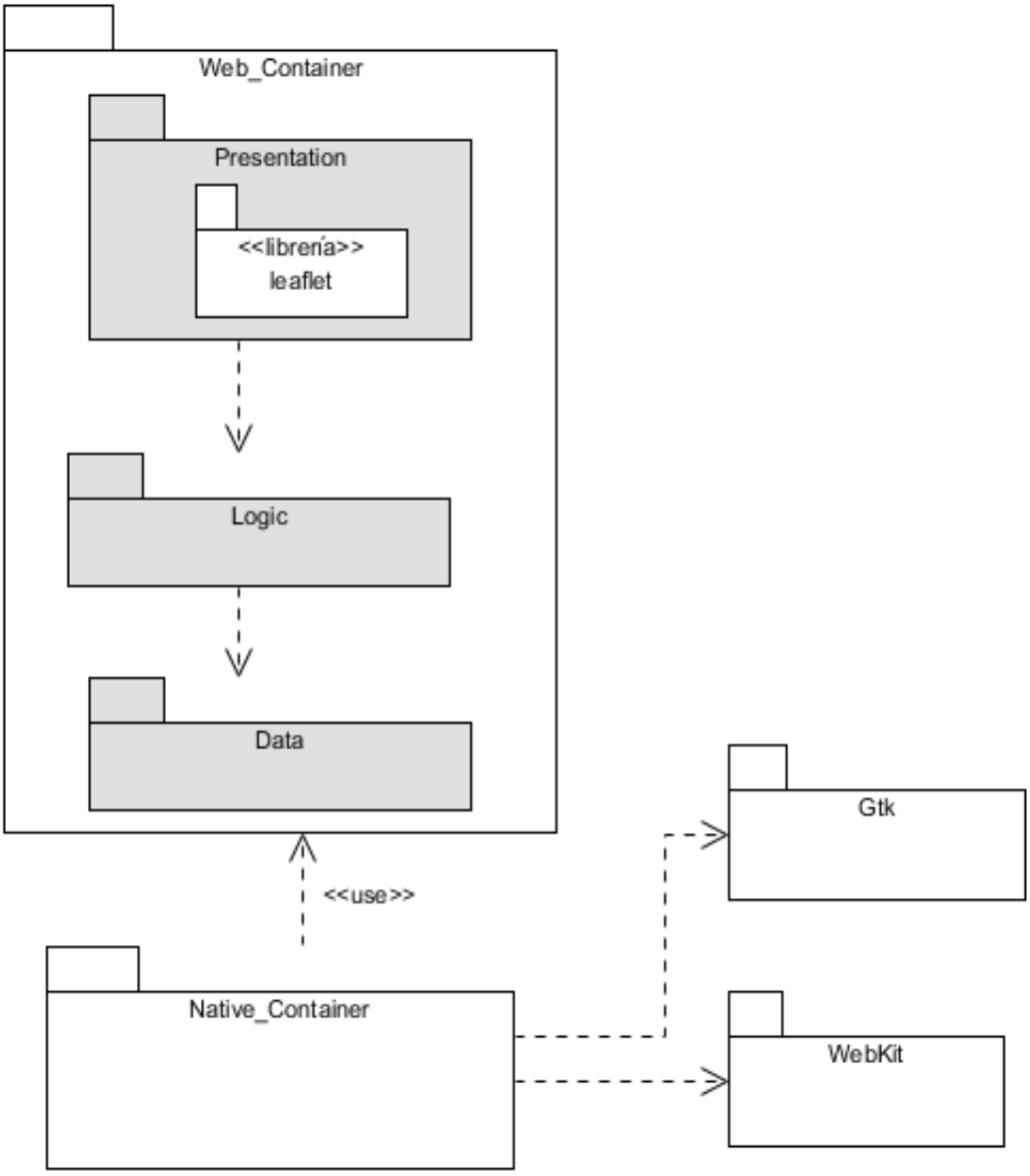


Ilustración 2- Diagrama de paquetes de la aplicación.

La aplicación se compone de los dos módulos principales llamados *Native_Container* y *Web_Container*. Además, el módulo *Web_Container* está compuesto de tres paquetes o subsistemas: el paquete de *Presentation* que contiene todo el código fuente relacionado con la construcción de la vista y sus interacciones con el usuario, el paquete *Logic*, que encapsula la lógica de la aplicación, como lo es la

interacción entre los elementos del mapa y la información que posee cada uno; por último, está el paquete *Data* que tiene el propósito de proveer las funcionalidades relacionadas con el acceso y almacenamiento de la información que contiene cada elemento del mapa representado en el paquete *Logic*. Cabe destacar del diagrama anterior, que el paquete que contiene la librería *Gtk* y el que contiene a *WebKit* no son partes desplegables de la aplicación de la aplicación en sí, sino que esta depende de ellos, y el estereotipo nombrado "*use"* tiene la finalidad de indicar una referencia débil, que se diferencia de la dependencia común en que esta referencia no necesita estar resuelta.

2.5.2 Diagrama de clases del diseño

A continuación, se muestra como queda el diseño basado en clases de la propuesta de solución.

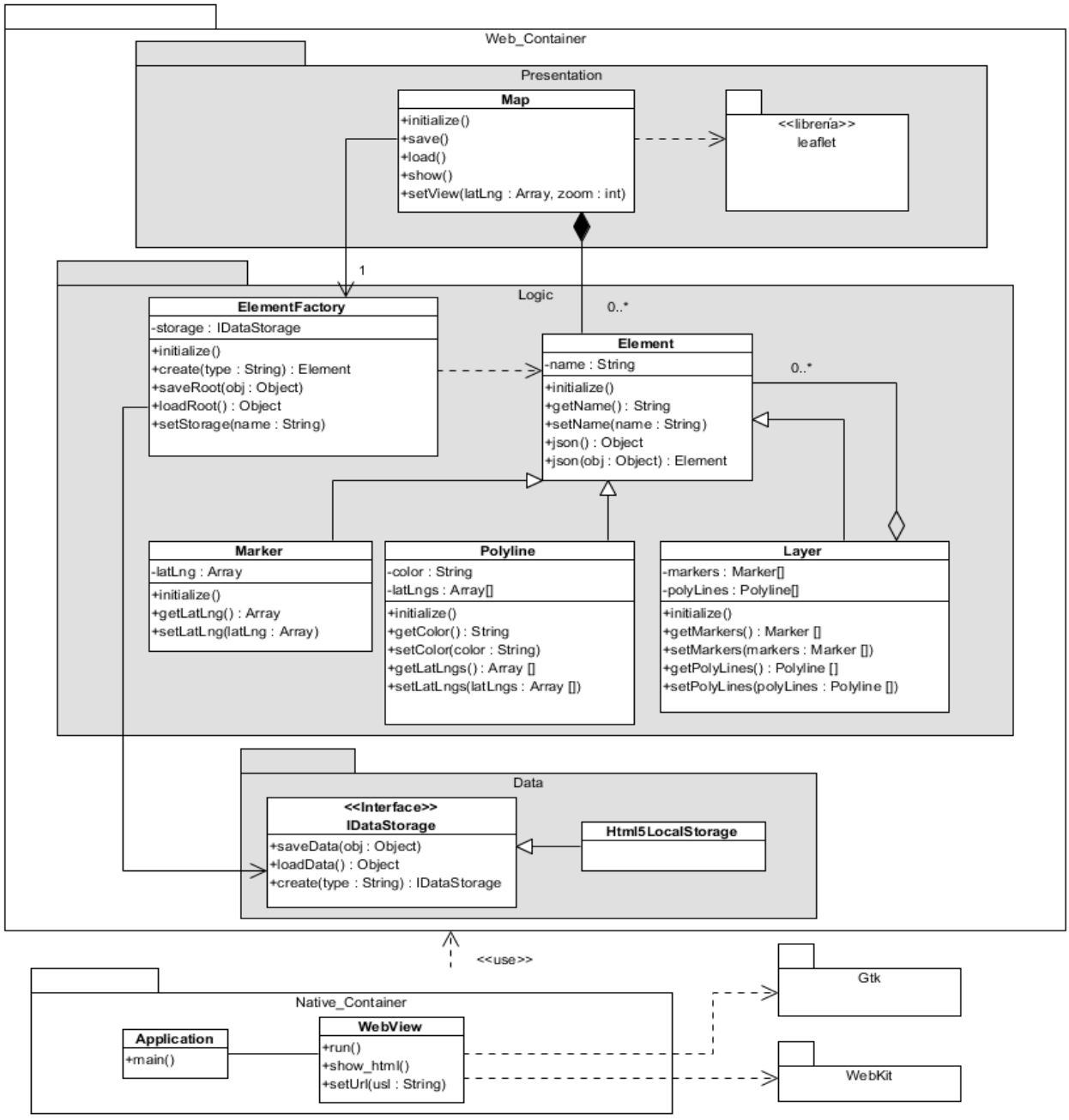


Ilustración 3 - Diagrama de clases del diseño.

De la figura anterior, las clases contenidas dentro del paquete *Web_Container* son desarrolladas usando los lenguajes *web* y las del paquete *Native_Container* usando el lenguaje de programación *Python*. La clase *Map* es la encargada de mostrar el mapa, así como sus rutas y marcadores de puntos. La clase *WebView* tiene como objetivo mostrar todo el contenido de la página web donde se dibujan el mapa, sus elementos y sus controles. Para el resto de las clases es fácil indicar cuál es su propósito, por lo que se verán a continuación los patrones de diseños empleados.

2.5.3 Patrones de diseño

Según plantea (Sommerville, 2011 pág. 189), el patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede reutilizarse en diferentes configuraciones. El patrón no es una especificación detallada, más bien, puede considerarla como una descripción de sabiduría y experiencia acumuladas, una solución bien probada a un problema común. En el contexto del *software* también existen patrones, encontrándose estos en la arquitectura, en el diseño de los componentes y clases de la arquitectura, y patrones específicos de cada lenguaje de programación, los cuales son también llamados idiomas. Este apartado se enfoca en los del diseño orientado a objetos, los cuales son conocimiento que se puede reutilizar y son de mucha utilidad a la hora de definir las clases que conforman un componente determinado del *software*.

2.5.3.1 Patrones *GRASP*

Los patrones *GRASP* (*General Responsibility Assignment Software Patterns*), como lo indica su nombre, son utilizados para la asignación de responsabilidades; en este caso, a las clases. Estos ayudan comúnmente a responderse la pregunta sobre qué clase debe ser la responsable de qué tareas generales, además de que son un conjunto de buenas prácticas que se deben tener en cuenta en todo momento cuando se está trabajando con el diseño orientado a objetos (Larman, 2003).

Experto

Consiste en asignarle la responsabilidad a la clase que cuenta con la información necesaria para cumplir dicha responsabilidad (Larman, 2003). Se hace uso de este patrón en la clase *Polyline* del paquete *Logic*, la cual posee los métodos necesarios para hacer visibles los datos referentes a su posición, nombre y color, que solo ella conoce.

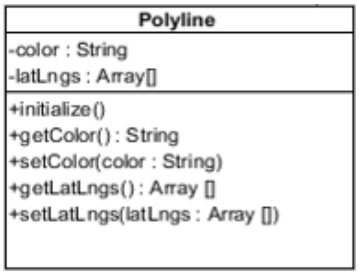


Ilustración 4 - Patrón Experto en la clase Polyline.

Creador

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Este principio consiste en asignarle la responsabilidad a la clase que cuenta con la información necesaria para cumplir con la responsabilidad. Así, una clase B tiene la responsabilidad de crear una instancia de la clase A si se cumplen uno o más de los casos siguientes (Larman, 2003):

- B agrega objetos de A.
- B contiene objetos de A.
- B registra instancias de objetos de A.
- B utiliza más estrechamente objetos de A.
- B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un experto con respecto a la creación de A).

Para la clase *Map* con respecto a la clase *Element* se cumplen la tercera y la quinta condición. Por lo que se define que la clase *Map* crea las instancias de la clase *Element*. Para ello utiliza la clase *ElementFactory* con el objetivo de desacoplar directamente el código referente a la creación de estos objetos. Esto trae como ventaja que la clase *Map* no necesite conocer el proceso de creación de estos objetos, sin embargo, si necesita conocer los datos necesarios para crearlos.

2.5.3.2 Patrones *GoF*[24]

Fabricación

Este patrón permite la creación de objetos sin necesidad de especificar su clase. Esto significa que la clase subyacente y los métodos para crear sus objetos pueden ser reemplazados en cualquier momento sin necesidad de cambiar la interfaz de creación de sus objetos que se utilizan en otros lugares del sistema (Odell, 2014). En la solución propuesta se utiliza este patrón en dos lugares distintos: en la clase *ElementFactory* para crear las instancias de los descendientes de la clase abstracta *Element*, y en la interfaz IDataStorage, para crear las instancias de los descendientes de la clase *IDataStorage*, que en este caso es una sola, *HTMLLocalStorage*.

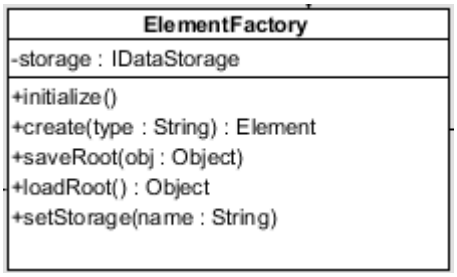


Ilustración 5 - Patrón Fabricación en la clase ElementFactory.

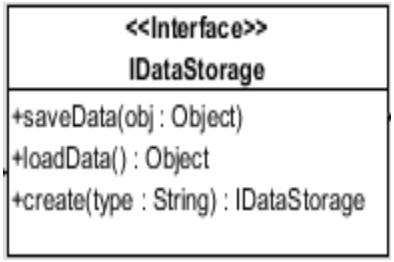


Ilustración 6 - Patrón Fabricación en la clase IDataStorage.

[24] *GoF*: Banda de los cuatro, o en inglés, *Gang of Four*, publicados por Gamma, Helm, Johnson y Vlossodes en 1995. Estos patrones son una serie de plantillas que dan soluciones concretas (dado que esas soluciones son dadas en un contexto determinado) a problemas específicos que se presentan comúnmente en el diseño orientado a objetos.

Valoraciones parciales del capítulo 2

El proceso seguido durante este capítulo para el análisis y el diseño de la aplicación de mapas para Nova ha permitido una mayor comprensión de la estructura del *software*, además de que posibilitó definir el dominio del problema y el espacio de soluciones para este en el contexto de esta investigación. Se utilizó el diseño de la arquitectura para obtener una visión de alto nivel de cómo estará construido el *software*. También se modelaron los componentes que componen la arquitectura mediante el diseño orientado a objetos. Para esto se utilizó el diagrama de clases del diseño, el cual brindó una perspectiva más cercana a la implementación del producto, además que junto con esto se definieron también los patrones de diseño que se evidenciaron en el *software*.

Capítulo 3. Implementación y Prueba

Introducción

En este capítulo se procede a implementar la solución propuesta utilizando los lenguajes, herramientas, metodologías y tecnologías antes mencionadas. Además de que se procede con la definición de un estándar de codificación, el cual le dará uniformidad al código fuente. Finalmente se procederá con la ejecución de las pruebas. Para ello se han de seleccionar los niveles y métodos de prueba que resulten pertinentes para lograr la confiabilidad e integridad de la herramienta que se necesita construir.

3.1 Implementación

Durante la implementación se procede a codificar la solución propuesta cumpliendo con el diseño que se estableció durante el análisis y diseño de la misma. Según lo define (Rodríguez Sánchez, 2015), esta se realiza de manera iterativa e incremental, repitiendo en cada iteración el flujo de trabajo que pasa por las disciplinas Requisitos, Análisis y Diseño, Implementación y Pruebas Internas.

3.1.1 Diagrama de componentes

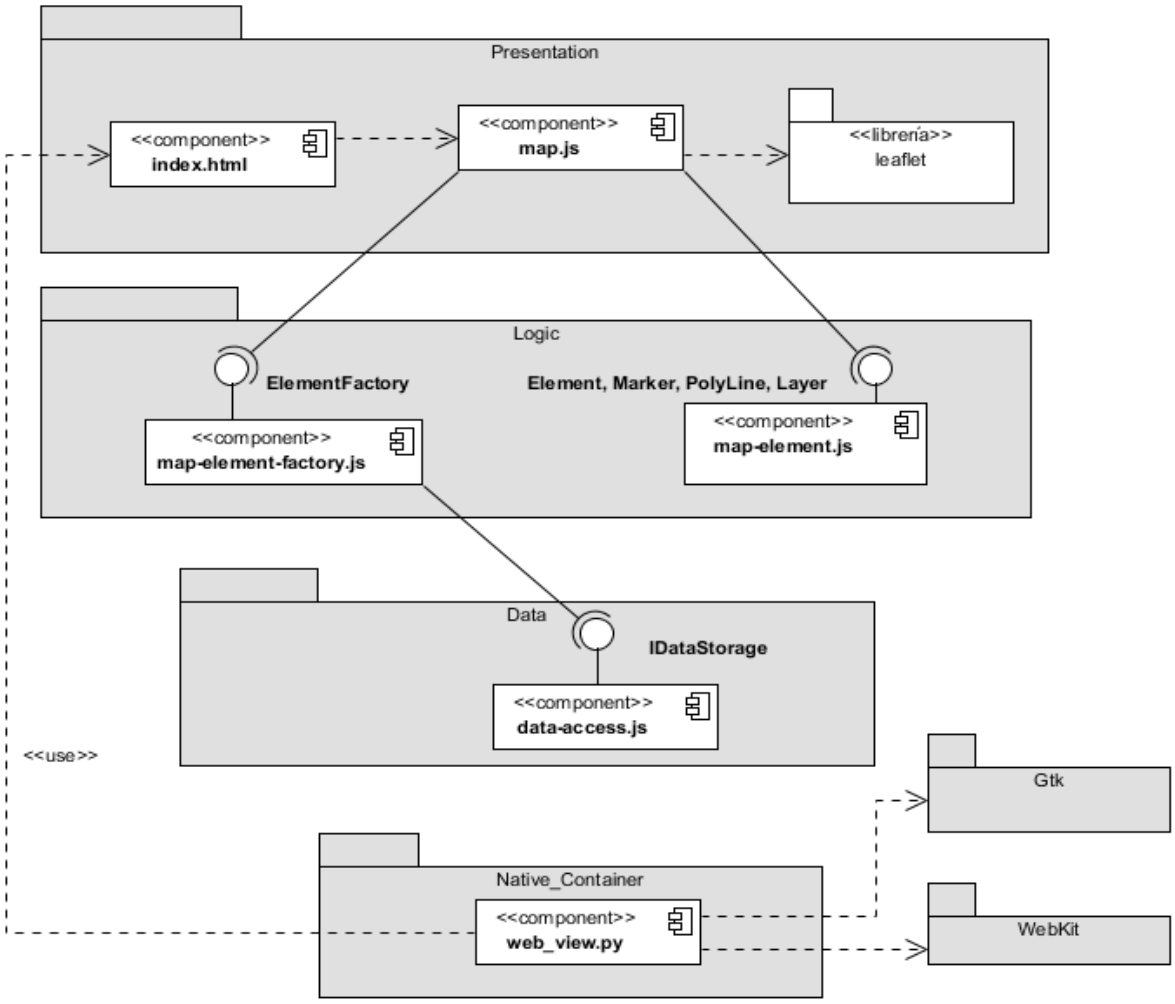


Ilustración 7 - Diagrama de componentes de la herramienta.

La figura anterior muestra los componentes desplegables de la herramienta de mapas. De ellos, solo el componente "*map-element.js*" y "*data-access.js*" se pueden reutilizar de manera independiente cada uno, dado que ninguno de ellos tiene dependencias con otros componentes del sistema. El resto de los componentes se puede reutilizar también, pero con la limitante de que hay que proveer las dependencias que ellos poseen, como se muestra en la figura anterior.

Los componentes del diagrama anterior tienen las siguientes responsabilidades:

- ***index.html***: Este componente es el encargado de ofrecer el contenedor sobre el cual se dibujará el mapa. En el estarán los elementos *HTML* que componen el mapa y sus elementos tales como marcadores y rutas.
- ***map.js***: En este componente se encapsulan las funcionalidades y algoritmos tanto para mostrar el mapa como sus elementos y controles de interfaz de usuario. Aquí es donde se manejan los eventos del usuario y se instancian los componentes que forman la vista. Requiere la librería *Leaflet*, la cual es la que realmente se encarga de procesar los datos del mapa como las capas y elementos, para luego mostrarlos.
- ***map-element.js***: Este componente contiene todas las clases de la lógica del negocio. Tales clases son: *Element*, *Marker*, *Polyline* y *Layer*, cada una de las cuales posee la información necesaria para representar las entidades que componen al mapa en la solución propuesta.
- ***map-element-factory.js***: Este componente se encarga de brindar un objeto para crear las instancias de las clases que componen al mapa de manera que no se requiera conocer específicamente cuáles son esas clases desde el componente *map.js*.
- ***data-access.js***: Este componente abstrae el acceso a los datos persistentes de la aplicación, los cuales son la información que se guarda en los objetos de los elementos del componente *map-element.js*. Su utilidad está en que este componente encapsula cualquier funcionalidad y mecanismo real para almacenar los datos de manera persistente, de manera que desde el exterior de este no se necesita conocer cómo escribir ficheros en disco o leer de bases de datos. Solo es necesario conocer las interfaces que este expone e invocarlas con los parámetros correctos.
- ***web_view.py***: Este es uno de los componentes más importantes del sistema. Sin la existencia de este, la herramienta sería una aplicación *web*, por lo que es por este componente que dicha herramienta puede verse y ejecutarse como una aplicación de escritorio. Funciona como un pequeño navegador *web* (aunque está muy lejos de ser como *Mozilla Firefox*) que muestra el contenido que contiene toda la lógica relacionada con la creación e interacción del mapa en una ventana nativa del sistema operativo. Se auxilia de *Gtk* y *WebKit* para lograrlo, por lo que para desplegarlo para su uso en otros sistemas se debe tener en cuenta esto.

3.1.2 Estándares de codificación de la herramienta

Con el objetivo de garantizar que el código fuente fuera uniforme, fácil de leer y entendible, se definió un estándar de codificación que reúne una serie de pautas a cumplir. A continuación, se describen ejemplos de ello:

Longitud de línea

Las líneas no deben tener una longitud superior a los 80 caracteres.

Saltos de línea

Cuando una línea sea más larga que los 80 caracteres establecidos se le agrega un salto de línea. Teniendo en cuenta que se pueden agregar saltos de línea en las siguientes situaciones:

- Después de una coma.
- Antes de un operador.

Declaraciones de variables

Las variables y funciones se declaran usando la notación *camelCase*, que consiste en escribir los identificadores con letras minúsculas y en el caso de que este compuesto de más de una palabra, se unen todas y se escribe la primera letra de todas ellas (exceptuando a la primera) con letra mayúscula.

Los nombres de las funciones constructoras[25] y variables que tienen referencia a algún objeto de una clase se escriben con la notación *PascalCase*, que es igual que la notación *camelCase*, pero con la particularidad de que la primera letra del identificador siempre se escribe con mayúscula.

Apertura y cierre de llaves

Las llaves que delimitan a bloques de código, cuerpo de funciones, de expresiones condicionales, ciclos y clases se abren en la misma línea que comienza la instrucción que declara de los constructores anteriores y se cierran en una línea aparte, requiriendo que en esa línea no se escriba ninguna otra instrucción.

[25] Una función constructora es una función común y corriente en JavaScript que tiene la finalidad de crear instancias de objetos cuando se usa el operador *new*. Para más información, consulte el libro *JavaScript: the Definitive Guide* (sexta edición) de David Flanagan.

3.2 Pruebas de *software*

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente es necesario probar el *software* para descubrir y corregir la mayor cantidad de errores posibles antes de ser entregado. Su objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores (Pressman, 2010).

3.2.1 Niveles de prueba

Al considerar el proceso de prueba desde un punto de vista procedural, las pruebas dentro del contexto de la ingeniería de *software* en realidad son una serie de cuatro pasos que se implementan de manera secuencial **(Pressman, 2010)**.

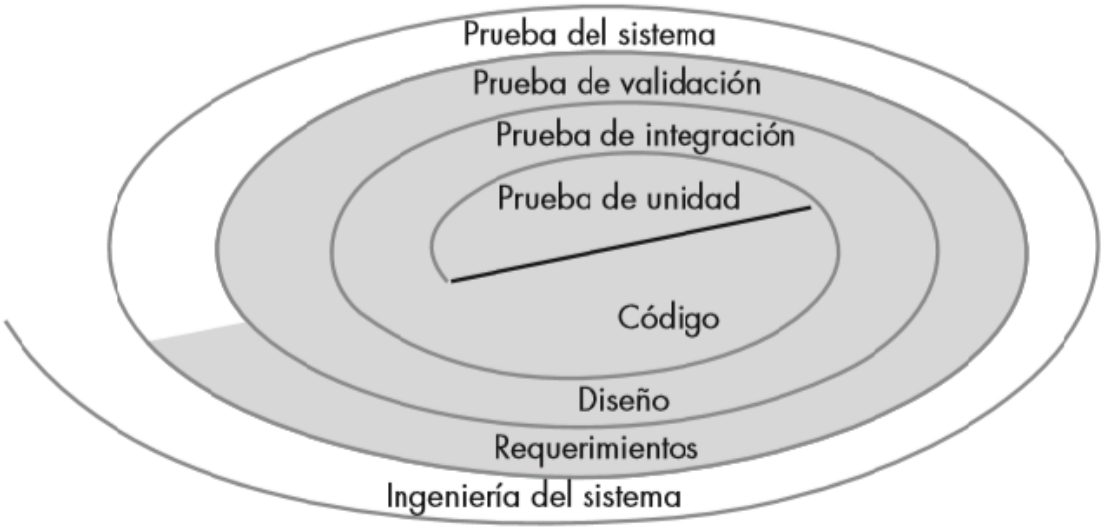


Ilustración 8 - Niveles de prueba.

Las pruebas en el nivel de unidad se concentran en las unidades más pequeñas del *software* (un componente, una clase o un procedimiento) tal como se implementaron. Por otra parte, las pruebas en el nivel de integración se concentran en el diseño, dígase la arquitectura del *software*, por ejemplo, mientras que las pruebas de validación se centran en los requisitos funcionales. Por último, las pruebas del sistema

ponen a prueba el *software* y otros elementos que componen el sistema cuando están funcionando como un todo.

En este trabajo solo basta con concentrarse en las pruebas de los niveles de validación y del sistema. Las pruebas unitarias no se documentarán aquí, por el simple motivo de que estas pruebas son completamente innatas al proceso de codificación del *software*. Además, las clases de la solución, que serían las unidades a ser probadas, tienen cierto grado de alta cohesión, por lo que depurarlas es un proceso sencillo relativo a los desarrolladores de esta.

Pruebas del sistema

Las pruebas de sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica (Pressman, 2010). En la solución propuesta se aplican casos de pruebas basados en historias de usuario con el objetivo de validar el funcionamiento esperado de la interfaz de usuario. Estas se realizan mediante el uso del método de caja negra, el cual nos permite probar el *software* a través de sus interfaces y los requisitos funcionales de este, sin tener que conocer el más mínimo detalle de su implementación interna. Entre los tipos de errores que se pretenden encontrar con las pruebas que usan el método de caja negra están los siguientes:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores de comportamiento o desempeño.
- Errores de inicialización y de terminación.

Pruebas de validación

Estas pruebas se realizan en este trabajo con el objetivo de que el cliente evalúe y emita su criterio de aceptación con respecto a la solución. Como técnica se escoge la de tipo alfa. Esta se lleva a cabo, por un cliente, en el lugar de desarrollo, donde se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas que se realizan con esta técnica se llevan a cabo en un entorno

controlado y para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente (Cabeza Chávez, 2015).

A continuación, se realizan pruebas de validación a los requisitos funcionales a partir de las descripciones de las historias de usuario, utilizando el método de prueba de caja negra, y de este, el método de partición de equivalencia. Véase (Pressman, 2010).

3.2.2 Diseño de casos de prueba basados en historias de usuario

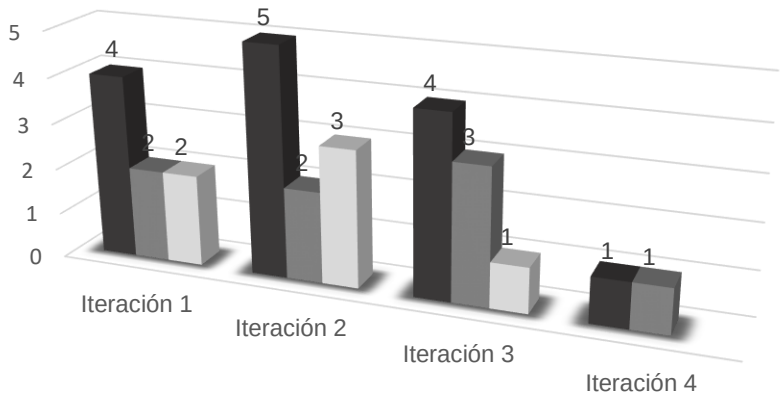
Tabla 5 - Caso de prueba 6: Insertar un nuevo nombre a algún marcador.

Escenario	Descripción	V1 campo de texto con el título del marcador	Respuesta del sistema	Flujo central
EC 6.1 Insertar el nuevo nombre.	La herramienta captura lo que el usuario escribió en el campo de texto del nombre y cambia el nombre del marcador a ese valor.	Mi Casa	Cuando el usuario presiona con el botón izquierdo del ratón sobre el marcador en un momento posterior, se le muestra un pequeño diálogo que sale del marcador (*popup*) con el nombre nuevo.	1. El usuario selecciona la opción de insertar nuevo nombre sobre algún marcador. 2. El usuario introduce el nombre. 3. El usuario presiona el botón “OK”, indicando que ha introducido el nuevo nombre.
EC 6.2 No insertar el nuevo nombre.	El usuario no escribe nada en el campo de texto del nombre.		Cuando el usuario presiona con el botón izquierdo del ratón sobre el marcador en un momento posterior, no se muestra nada.	

3.3 Resultados de los casos de prueba

De las pruebas de caja negra realizadas a la herramienta se detectaron solo 8 no conformidades en 4 iteraciones, de ellas solo 2 eran de errores ortográficos y el resto de comportamientos incorrectos. A continuación, se ilustra lo anterior en el siguiente gráfico[26].

Tabla 6 - Estadísticas de pruebas.



	Iteración 1	Iteración 2	Iteración 3	Iteración 4
No conformidades	4	5	4	1
Resueltas	2	2	3	1
Pendientes	2	3	1	

No conformidades Resueltas Pendientes

Valoraciones parciales del capítulo 3

En el transcurso de este capítulo se presentaron los estándares de codificación por los cuales se iban a regir los desarrolladores de la solución propuesta, resultando en una mayor claridad y limpieza del código fuente. Pruebas como las pruebas del sistema realizadas permitieron revelar y resolver 8 no conformidades.

[26] Cabe destacar que las no conformidades representadas en cada iteración incluyen también la no cnformidades de la iteración anterior.

Conclusiones

Al culminar el desarrollo de la presente investigación se arriba a las siguientes conclusiones:

- La sistematización de las herramientas de mapas que trabajan sin conexión a internet permitió determinar sus funcionalidades principales y luego implementarlas en la solución propuesta por esta investigación.
- El uso de la arquitectura N-capas y de las tecnologías *Leaflet*, *Python*, *WebKit*, *Gtk* y los lenguajes de la web; guiados por la metodología AUP-UCI, permitió implementar un software de acuerdo a los requisitos definidos.
- La aplicación de las pruebas, métodos y técnicas demostraron el correcto funcionamiento de la herramienta de mapas sin conexión para Nova, logrando la satisfacción del cliente y la calidad esperada por este.

Recomendaciones

Una vez concluida la investigación y el desarrollo de la propuesta de solución se recomienda:

- Agregar capas que sean estáticas y que posean la información sobre puntos de valor histórico, económico, político y de otros intereses, con la información de cada uno de ellos, para que los usuarios puedan aprovecharlas.
- Investigar cómo sería posible lograr que las rutas trazadas en el mapa se ajusten a las carreteras y caminos que se muestran en él.
- Incorporar en la herramienta desarrollada la capa base usando *Leaflet* a partir del formato *SVG*[27] u otro que use la representación de los datos de forma vectorial.

[27] *Scalable Vector Graphics*.

Referencias

Alegsa, Leandro. 2016. Información sobre Python: Alegsacom.ar. *Alegsa.com.ar.* [En línea] 13 de 6 de 2016. [Citado el: 7 de 3 de 2017.] http://www.alegsa.com.ar/Dic/python.php.

Añgoritmos de agrupamiento conceptuales: un estado del arte. **Alejandro, Guerra Gardón, Sandro, Vega Pons y Joser, Ruiz Shulcloper. 2012.** 2142, La Habana : s.n., 2012. ISSN 2072-6287.

2015. Business Insider. [En línea] 7 de 12 de 2015. [Citado el: 2016 de 11 de 11.] http://www.businessinsider.com/idc-smartphone-os-market-share-2015-12.

Cabeza Chávez, Yanet. 2015. *Módulo de JAVA para la herramienta Auditoría de Código Fuente.* La Habana : s.n., 2015.

Ciencia M. 2011. conceptodefinicion.de. [En línea] Ciencia M, 20 de 9 de 2011. [Citado el: 6 de 12 de 2016.] http://conceptodefinicion.de/mapa/.

Collado, Christian. 2016. Aplicaciones nativas vs híbridas, ¿qué son y cuáles son mejores? *Android4all.* [En línea] 21 de 1 de 2016. [Citado el: 10 de 5 de 2017.] https://andro4all.com/2016/01/apps-nativas-vs-apps-hibridas-ventajas-desventajas.

Confederación de Empresarios de Andalucía. 2010. Definición de SIG | Sistemas de Información Geográfica. *Sistemas de Información Geográfica.* [En línea] 2010. [Citado el: 4 de 1 de 2017.] http://sig.cea.es/SIG.

Dariem. 2014. Nova: ¿personalización o distribución?: HumanOS. *HumanOS.* [En línea] 10 de Febrero de 2014. [Citado el: 8 de Diciembre de 2016.] https://humanos.uci.cu/2014/02/nova-personalizacion-o-distribucion/.

Departamento de Hacienda y Finanzas. 2012. *Guía técnica para el desarrollo de soluciones móviles.* San Sevastian : Donostia, 2012.

Elizalde, Rosa Miriam. 2015. Cubadebate. *Díaz-Canel: Existe la voluntad de poner la Informatización y la Internet al servicio de todos (+ Video).* [En línea] Universidad de las Ciencias Informáticas, 20 de 2 de 2015. [Citado el: 22 de 6 de 2017.] http://www.cubadebate.cu/noticias/2015/02/20/diaz-canel-existe-la-voluntad-del-partido-y-el-gobierno-de-poner-la-internet-al-servicio-de-todos/.

Environmental Systems Research Institute Inc. 2016. Georreferenciación y sistemas de coordenadas: ArcGIS Resource Center. *ArcGIS Resources Center.* [Online] 2016. [Cited: 2 5, 2017.] http://resources.arcgis.com/es/help/getting-started/articles/026n0000000s000000.htm.

Flor de loto. 2013. Significados. [En línea] Flor de loto, 2013. [Citado el: 5 de 12 de 2016.] https://www.significados.com/mapa/.

Garcia, Carolina. 2016. Educar. [En línea] 22 de 2 de 2016. [Citado el: 5 de 12 de 2016.] http://www.educ.ar/sitios/educar/recursos/ver?id=14655.

GCF Community Foundation International . 2016. Informática Básica - ¿Qué es una aplicación web? *GCFAprendeLibre.* [En línea] 2016. [Citado el: 10 de 5 de 2017.] https://www.gcfaprendelibre.org/tecnologia/curso/informatica_basica/aplicaciones_web_y_todo_acerca_de_la_nube/1.do.

Hanmer, Robert. 2013. *Patern-Oriented Software Architecture for Dummies.* s.l. : John Wiley & Sons, Ltd., 2013. 978-1-119-96399-8.

Hazzard, Erik. 2011. *OpenLayers 2.10: Beginners's guide.* Birmingham : Packt Publishing Ltd., 2011. ISBN 978-1-849514-12-5.

Jesus Tramulla, Piedad Garrido. 2006. *Software libre para servicios de informacion digital.* Mexico : Pearson Prentice Hall, 2006.

Jesús Tramullas, Piedad Garrido. 2006. Fundamentos. *Software libre para servicios de información digital.* Madrid : Pearson Prentice Hall, 2006.

Larman, Craig. 2003. *UML y Patrones.* s.l. : Prentice Hall, 2003.

Malkani, Faiz. 2016. Native vs. Hybrid Apps, Strengths and Weaknesses. *xda-developers.* [En línea] 20 de 1 de 2016. [Citado el: 10 de 5 de 2017.] https://www.xda-developers.com/native-apps-hybrid-apps-the-strengths-and-weaknesses-of-todays-paradigms/.

Ministerio de Comunicaciones. 2015. Ministerio de Comunicaciones. *Ministerio de Comunicaciones: I Taller Nacional de Informatización y Ciberseguridad.* [En línea] DESOFT, 20 de 2 de 2015. [Citado el: 22 de 6 de 2017.] http://www.mincom.gob.cu/?q=node/767.

Molina, Mario Roberto. 2007. HDM. [En línea] 8 de 7 de 2007. [Citado el: 5 de 12 de 2016.] http://hablemosdehistoria.com/etiqueta/mapas-historicos/.

Morales, Aurelio. 2016. Las 10 mejores librerías JavaScript para web mapping en 2017. *MappingGIS.* [En línea] 31 de 3 de 2016. [Citado el: 2 de 5 de 2017.] https://mappinggis.com/2015/03/las-mejores-apis-javascript-para-webmapping/.

Moreno, Bulmaro Pacheco, Adrian Esquer. 2007. *Informática 1.* Mexico : 3era edicion, 2007.

Odell, Den. 2014. *Pro JavaScript Development: Coding, Capabilities, and Tooling.* Nueva York : Apress, 2014. 978-1-4302-6269-5.

Open Source Geospatial Foundation. 2016. MapServer. [En línea] 2016. [Citado el: 7 de 12 de 2016.] http://mapserver.org/es/about.html#about.

MapServer. 2016. MapServer. [En línea] 2016. [Citado el: 7 de 12 de 2016.]

Prado, Gabriel Vera. 2012 . Portal Educativo. [En línea] 2012 . [Citado el: 5 de 11 de 2016.] http://www.portaleducativo.net/tercero-basico/776/Tipos-de-mapas.

Pressman, Roger S. 2010. *Ingeniería del Software. Un enfoque Práctico.* México DF. : McGraw-Hill, 2010. 978-607-15-0314-5.

Rahn, Torsten and Niehüser, Dennis. 2016. The Marble Handbook. *The Marble Handbook.* [Online] 2016. [Cited: 1 19, 2017.] https://docs.kde.org/trunk5/en/kdeedu/marble/index.html.

Rodríguez Sánchez, Tamara. 2015. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : s.n., 2015.

Rubén Alcaraz Martínez, Mireia Ribera Turró. 2015. No Solo Usabilidad. [En línea] 9 de 1 de 2015. [Citado el: 7 de 12 de 2016.] http://www.nosolousabilidad.com/articulos/mapas_digitales.htm.

Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : s.n.

Santiago, Antonio. 2015. *The book of OpenLayers 3. Theory & practice.* s.l. : Leanpub, 2015.

Sommerville, Ian. 2011. *Ingeniería de software.* Naucalpan de Juárez : Addison-Wesley, 2011. 978-607-32-0603-7.

Stallman, Richard M. 2004. *Software libre para una sociedad libre.* Madrid : Traficantes de Sueños, 2004.

Anexo 1

Tabla 7 - Historia de usuario 3.

Historia de Usuario	
Número: HU_3	**Nombre del requisito:** Editar puntos en el mapa.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 2 horas
Riesgo en desarrollo: Medio	**Tiempo real:** 2 horas
Descripción: Cuando el usuario presiona el botón de editar, ubicado en el grupo de controles de la parte superior derecha del panel del mapa, se le muestran a este los elementos del mapa en modo de edición. El usuario entonces puede proseguir a editar los componentes arrastrándolos con el ratón.	
Observaciones: ❖ El botón “cancelar” deshace todos los cambios hechos a todos los elementos que se editaron. ❖ Si se presiona otro botón, también se deshacen los cambios hechos a todos los elementos. ❖ Solamente se guardan todos los cambios pulsando el botón “guardar”.	
Prototipo de Interfaz:	

Herramienta de mapas sin conexión para Nova

Tabla 8 - Historia de usuario 4.

Historia de Usuario	
Número: HU_4	**Nombre del requisito:** Eliminar puntos del mapa.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 3 horas
Riesgo en desarrollo: Medio	**Tiempo real:** 2 horas
Descripción: Cuando el usuario presiona el botón de eliminar, ubicado en el grupo de controles de la parte superior derecha del panel del mapa, este podrá eliminar los marcadores del mapa que el seleccione solo pulsado con el botón izquierdo del ratón sobre ellos.	
Observaciones: ❖ El botón “cancelar” deshace todos los cambios hechos a todos los elementos que se editaron. ❖ Si se presiona otro botón, también se deshacen los cambios hechos a todos los elementos.	

- ❖ Solamente se guardan todos los cambios pulsando el botón “guardar”.
- ❖ El botón “borrar todo” elimina todos los marcadores y rutas del mapa.

Prototipo de Interfaz:

Tabla 9 - Historia de usuario 6.

Historia de Usuario	
Número: HU_6	**Nombre del requisito:** Editar rutas en el mapa.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 2 horas
Riesgo en desarrollo: Medio	**Tiempo real:** 2 horas
Descripción: Cuando el usuario presiona el botón de editar, ubicado en el grupo de controles de la parte superior derecha del panel del mapa, se le muestran a este los elementos del mapa en modo de edición. El usuario entonces puede	

proseguir a editar las rutas del mapa arrastrando con el ratón los puntos que conforman a esta. Incluso puede eliminar puntos de la ruta presionando dos veces seguidas con el botón izquierdo del ratón (doble *click*) o adicionar nuevos puntos seleccionando y arrastrando los puntos intermedios a los puntos que definen la ruta.

Observaciones:

- El botón “cancelar” deshace todos los cambios hechos a todos los elementos que se editaron.
- Si se presiona otro botón, también se deshacen los cambios hechos a todos los elementos.
- Solamente se guardan todos los cambios pulsando el botón “guardar”.

Prototipo de Interfaz:

Tabla 10 - Historia de usuario 7.

Historia de Usuario	
Número: HU_7	**Nombre del requisito:** Eliminar rutas del mapa.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 3 horas

Herramienta de mapas sin conexión para Nova

Riesgo en desarrollo: Medio	**Tiempo real:** 2 horas

Descripción:

Cuando el usuario presiona el botón de eliminar, ubicado en el grupo de controles de la parte superior derecha del panel del mapa, este podrá eliminar las rutas del mapa que el seleccione solo pulsado con el botón izquierdo del ratón sobre ellos.

Observaciones:

- El botón “cancelar” deshace todos los cambios hechos a todos los elementos que se editaron.
- Si se presiona otro botón, también se deshacen los cambios hechos a todos los elementos.
- Solamente se guardan todos los cambios pulsando el botón “guardar”.
- El botón “borrar todo” elimina todos los marcadores y rutas del mapa.

Prototipo de Interfaz:

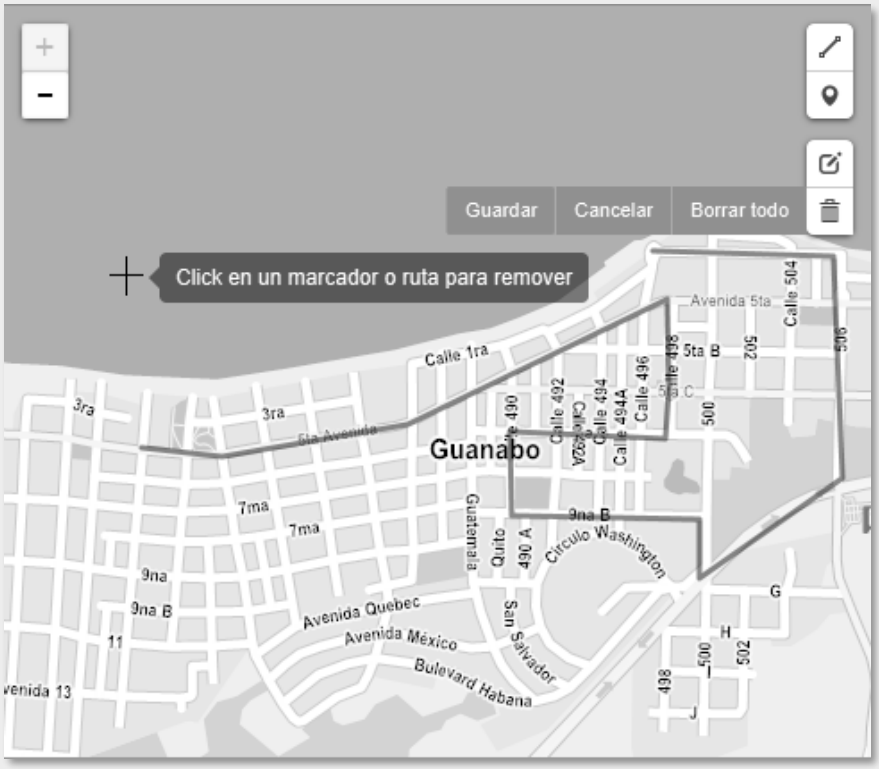


Tabla 11 - Historia de usuario 8.

Historia de Usuario	
Número: HU_8	**Nombre del requisito:** Insertar un nuevo nombre a algún punto.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 3
Prioridad: Baja	**Tiempo estimado:** 4 horas
Riesgo en desarrollo: Bajo	**Tiempo real:** 4 horas
Descripción: Cuando el usuario presiona el botón derecho del ratón sobre un marcador, se le muestra un menú que contiene la opción “insertar nombre”. Cuando se selecciona esta se le muestra al usuario un pequeño diálogo con un campo de texto y dos botones (de aceptar y cancelar) para que el usuario introduzca el nombre del punto.	
Observaciones:	
Prototipo de Interfaz:	

Herramienta de mapas sin conexión para Nova

Herramienta de mapas sin conexión para Nova

Tabla 12 - Historia de usuario 9.

Historia de Usuario	
Número: HU_9	**Nombre del requisito:** Insertar un nuevo nombre a alguna ruta.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 3
Prioridad: Baja	**Tiempo estimado:** 4 horas
Riesgo en desarrollo: Bajo	**Tiempo real:** 4 horas
Descripción: Cuando el usuario presiona el botón derecho del ratón sobre una ruta, se le muestra un menú que contiene la opción “insertar nombre”. Cuando se selecciona esta se le muestra al usuario un pequeño diálogo con un campo de texto y dos botones (de aceptar y cancelar) para que el usuario introduzca el nombre de la ruta.	

Herramienta de mapas sin conexión para Nova

Observaciones:
Prototipo de Interfaz:

Herramienta de mapas sin conexión para Nova

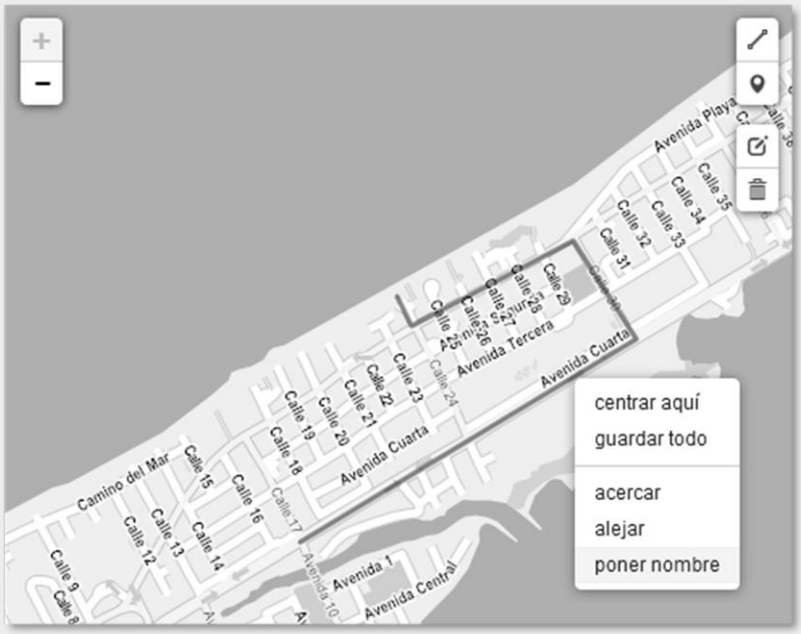


Tabla 13 - Historia de usuario 10.

Historia de Usuario	
Número: HU_10	**Nombre del requisito:** Centrar la vista del mapa en un punto del mapa visible.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 3
Prioridad: Baja	**Tiempo estimado:** 1 horas
Riesgo en desarrollo: Bajo	**Tiempo real:** 1 horas
Descripción: Cuando el usuario presiona el botón derecho del ratón sobre alguna parte aleatoria del mapa, se le muestra un menú que contiene la opción “centrar aquí”. Cuando se selecciona esta la vista del mapa se centra en la posición sobre la que el usuario presionó el botón derecho del ratón.	
Observaciones:	
Prototipo de Interfaz: 	

Tabla 14 - Historia de usuario 11.

Historia de Usuario	
Número: HU_11	**Nombre del requisito:** Guardar los datos de los marcadores y rutas en un fichero cuando la aplicación se vaya a cerrar.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 3
Prioridad: Media	**Tiempo estimado:** 3 horas
Riesgo en desarrollo: Medio	**Tiempo real:** 2 horas
Descripción: Cuando el usuario cierra la aplicación, automáticamente se guardan todos los cambios que este haya hecho en los datos del mapa (marcadores y rutas) en un fichero de forma local.	
Observaciones: El usuario puede guardar todos los cambios hechos a los datos del mapa antes de cerrar la aplicación presionando con el botón derecho del ratón y seleccionando la opción “guardar todo”.	
Prototipo de Interfaz:	

Tabla 15 - Historia de usuario 12.

Historia de Usuario	
Número: HU_12	**Nombre del requisito:** Cargar los datos de los marcadores y rutas almacenados previamente en un fichero cuando la aplicación se inicia.
Programador: Adackny Castillo Fernández	**Iteración asignada:** 3
Prioridad: Media	**Tiempo estimado:** 3 horas
Riesgo en desarrollo: Medio	**Tiempo real:** 3 horas
Descripción: Cuando el usuario ejecuta la aplicación, esta automáticamente carga todos los datos previamente guardados en un fichero local, que contiene la información sobre los marcadores y rutas que el usuario haya definido en otro momento.	

Herramienta de mapas sin conexión para Nova

Observaciones:
Prototipo de Interfaz: No contiene.