|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
| **Trabajo práctico integrador** |
| **Cátedra de Diseño de Sistemas**  **Año 2016**  **Fecha de actualización: 06/08/2016** |

# Tabla de contenido

[Tabla de contenido 2](#_Toc458210103)

[Objetivos Generales 3](#_Toc458210104)

[Evaluación y Condiciones de Aprobación 4](#_Toc458210105)

[Contexto 5](#_Toc458210106)

[Dominio de alto nivel 5](#_Toc458210107)

[Requerimientos de alto nivel 5](#_Toc458210108)

[Entrega 0: Análisis del dominio 7](#_Toc458210109)

[Entrega 1 7](#_Toc458210110)

[Cálculo de cercanía 7](#_Toc458210111)

[Cálculo de disponibilidad 7](#_Toc458210112)

[Búsqueda de puntos 8](#_Toc458210113)

[Requerimientos detallados 8](#_Toc458210114)

[Entrega 2 8](#_Toc458210115)

[Caso 1: Búsqueda de puntos de interés de nuestro sistema 8](#_Toc458210116)

[Caso 2: CGPs 9](#_Toc458210117)

[Caso 3: Bancos 9](#_Toc458210118)

[Entrega 3 11](#_Toc458210119)

[Entrega 4 13](#_Toc458210120)

[Proceso 1: Actualización de Locales Comerciales 13](#_Toc458210121)

[Proceso 2: Baja de POIs 14](#_Toc458210122)

[Proceso 3: Agregar acciones para todos los usuarios 14](#_Toc458210123)

[Proceso 4: Definición de un proceso múltiple 14](#_Toc458210124)

[Manejo de resultados / errores en la Ejecución del Proceso 14](#_Toc458210125)

[Entrega 5 15](#_Toc458210126)

[5.1. Modelado de datos 15](#_Toc458210127)

[5.2. Servicio REST 15](#_Toc458210128)

[5.3. Diseño e implementación de Interfaz de Usuario 15](#_Toc458210129)

[Entrega 6: Persistencia a un medio relacional 18](#_Toc458210130)

# Objetivos Generales

* Tomar decisiones de Diseño de una Solución Compleja identificando Ventajas y Desventajas.
* Comunicar el Diseño utilizando distintas herramientas (UML - código/pseudocódigo - prosa - DER).
* Identificar Requerimientos No Funcionales (Atributos de Calidad) y su impacto en la Arquitectura.
* Resolver Situaciones Problemáticas del Diseño utilizando Patrones de Diseño.
* Diseñar Artefactos de Prueba que permitan Validar los Requerimientos definidos.
* Construir un Modelo de Datos persistente acorde a la Implementación.
* Diseñar y Construir Interfaces de Usuario en Distintas Tecnologías.
* Diseñar la Comunicación con una Interfaz Externa al Sistema.
* Validar el Diseño propuesto utilizando herramientas de Testing.
* Comprender y Especificar la Arquitectura del Producto.
* Implementar la Solución en una Tecnología concreta para Validar el Diseño Propuesto.
* Comprender que la tecnología condiciona al diseño.
* Utilizar Herramientas de Soporte para el Diseño y Construcción de Software:
  + control de versiones
  + entorno de desarrollo integrado
  + herramientas de manejo de dependencias
  + herramientas de construcción
  + herramientas de concepción, y comunicación de diseño
* La Comunicación y el Trabajo en Equipo como elementos primordiales en el Proceso.

# Evaluación y Condiciones de Aprobación

* El Equipo: 4 a 6 Integrantes.
* Se evaluará en cada Entrega al Equipo y a cada Integrante.
* La aprobación será por Entrega Grupal y por Integrante.
* Se le informará al Equipo y a cada Integrante la Calificación Obtenida.
* La Calificación puede ser:
  + Insuficiente
  + Suficiente
  + Muy Bueno
  + Excelente
* El Alumno/Equipo podrá tener solamente 2(dos) Entregas con calificación Insuficiente que podrá ser recuperada en 2(dos) Instancias. En caso de obtener una Tercer Entrega con calificación Insuficiente el Alumno/Equipo no aprobará el Trabajo Práctico condición necesaria para la aprobación de la Asignatura.

# Contexto

El gobierno de la ciudad, con el objetivo descentralizar el acceso a la información y disminuir la congestión en centros de Gestión y Participación (CGPs), quiere llevar a cabo un proyecto ambicioso: contar con dispositivo táctiles en múltiples esquinas de la ciudad, conectados a internet, que permitan realizar trámites y obtener información.

Tras contactarse con la Regional Buenos Aires, de la Universidad Tecnológica Nacional., y recibir la propuesta en la Cátedra de Diseño de Sistemas, luego de varias reuniones y discusiones sobre cómo abordar el proyecto, todas las partes están de acuerdo en encarar una primera etapa en la que estos dispositivos sólo visualizarán puntos de interés[[1]](#footnote-1), en algunas locaciones particulares. Si el proyecto tiene éxito[[2]](#footnote-2), en etapas posteriores se incluirán más dispositivos y más funcionalidades.

# Dominio de alto nivelHow-to-add-your-place-of-business-or-points-of-interest-to-Google-Maps.jpg

Un punto de interés (*POI*, por sus siglas en inglés) es básicamente una posición en un mapa, que suele representarse mediante un ícono.

De este conocemos su dirección, calle principal, entre qué calles se encuentra, número, y datos más específicos como el piso, departamento, unidad, el código postal, localidad, barrio, provincia, país, además de las coordenadas geográficas resultado del proceso de geocodificación: latitud y longitud.

Algunos puntos de interés posibles son:

* Paradas de colectivos
* Centros de Gestión y Participación (CGP)
* Sucursales de Bancos
* Locales comerciales

# Requerimientos de alto nivel

Cada uno de estos dispositivos, más allá de sus particularidades físicas, es una computadora conectada a Internet. Los mismos serán provistos por otra empresa, “Touch.me”.

Nuestra responsabilidad será, entonces, diseñar, construir, probar y desplegar el sistema software que se ejecutará en estas máquinas.

El mismo cubrirá, al menos, las siguientes funcionalidades:

* Buscar puntos de interés, según criterios de búsqueda establecidos por el usuario
* Obtener información de un cierto punto
* Visualizar el resultado de las consultas
* Almacenar información de la consulta, con fines estadísticos.

Este sistema no sólo será utilizado por transeúntes en la vía pública, sino también por administradores del sistema, del Gobierno de la ciudad, que cargarán puntos desde distintos orígenes, y obtendrán estadísticas sobre las consultas.

# Entrega 0: Análisis del dominio

**Fecha de entrega: 22 de abril**

Para empezar, vamos a hacer un análisis preliminar del sistema: queremos identificar casos de uso y usuarios involucrados. Para ello, se solicita:

1. Modelar los Casos de Uso del dominio.
2. Modelar Diagramas de Clases en UML para comunicar el Diseño de los CU.
3. Definir Atributos de calidad y Requerimientos No Funcionales en base al Contexto del Sistema.

# Entrega 1

**Fecha de entrega: 22 de abril**

Queremos que las personas sean capaces de saber si pueden ir desde nos encontramos hacia otro lugar. Para ello vamos a ofrecerles tres servicios: consulta de cercanía, consulta de disponibilidad (horaria) y búsqueda de puntos.

### Cálculo de cercanía

En general, decimos que estamos cerca de un punto de interés si se encuentra a una distancia menor a 5 cuadras.

Sin embargo, hay excepciones:

1. Un parada de colectivo se considera cercana si estamos a menos de una cuadra.
2. Los CGP cumplen la condición de cercanía, si su coordenada está dentro de la zona delimitada por la comuna.
3. Para los locales comerciales, cada rubro define el radio de cercanía. Ejemplo: para las librerías escolares se considera cerca si está en un radio de 5 cuadras, para un kiosco de diarios en cambio son 2 cuadras, y así sucesivamente.

### Cálculo de disponibilidad

El cálculo de disponibilidad trata de saber si un POI está disponible en un cierto momento. Esto depende de cada tipo de POI:

1. El servicio de transporte de Colectivos está disponible a toda hora
2. Para los CGP:
   1. si se ingresa un valor x (el nombre de un servicio), la fecha debe estar en el rango de atención. Ej: si es sábado no se atiende Rentas, no está disponible. Si es lunes a las 20hs tampoco se encuentra disponible este servicio. En cambio si es lunes a las 10:30hs sí se encuentra disponible.
   2. si no se ingresa un valor x, considerar que exista al menos un servicio en el CGP que esté atendiendo en ese momento.
3. Para los Bancos, el mismo comportamiento que para los CGP pero considerando el horario de atención bancario (Lunes a Viernes de 10hs a 15hs)
4. Para los locales comerciales, considerar el horario de atención del local. Ejemplo: para Carrousel el horario de atención es de Lunes a Sábado de 10hs a 13hs y de 17hs a 20:30hs. Si son las 15 horas de un martes no debe estar disponible.

### Búsqueda de puntos

Finalmente, necesitamos buscar puntos en el mapa a partir de un texto libre, por ejemplo:

* Si se ingresa el texto "114", se debería encontrar las paradas del colectivo 114.
* Si se ingresa una palabra que corresponde a un rubro (por ejemplo "mueblería"), se deberían encontrar todos los cales de ese rubro.
* Si se ingresa una palabra clave, se debería encontrar todos los puntos que estén etiquetados mediante esa palabra clave.
* Si se ingresa parte de un servicio (por ejemplo "asesoramiento"), se deben encontrar todos CGPs que provean algún servicio parecido[[3]](#footnote-3) (por ejemplo "asesoramiento legal")

### Requerimientos detallados

1. Determinar cuándo un punto de interés está cerca de una coordenada geográfica.
2. Dado un momento (fecha:hora:minutos:segundos) y un valor x, determinar si un negocio está disponible.
3. Buscar un punto de interés en base a un texto libre[[4]](#footnote-4) [[5]](#footnote-5)

*Para el contexto y requerimientos definidos se solicita realizar el modelado de clases, la implementación en la tecnología seleccionada (considerando la aplicación de un patrón arquitectónico MVC) y los casos de prueba correspondientes.*

# Entrega 2

**Fecha de entrega: 3 de junio**

En esta iteración, la búsqueda por puntos de interés se vuelve más compleja:

* Por un lado, necesitamos agregar lo necesario al modelo de dominio para que un administrador pueda agregar, modificar y quitar puntos del mapa.
* Por otro lado, se incorporan nuevos orígenes de datos de puntos de interés, que se detallan a continuación.

### Caso 1: Búsqueda de puntos de interés de nuestro sistema

Muchos de los puntos de interés están almacenados en nuestro sistema. Esto no debe cambiar y seguir funcionando tal como hasta ahora.

### Caso 2: CGPs

Para consultar los CGP se dispone de un componente desarrollado en forma externa. El servicio acepta como parámetro un *string* que puede corresponder a la calle o zona, y devuelve la lista de CGPs disponibles según el siguiente formato:

Lista de CGPs, un objeto de la clase CentroDTO

* int: número de la comuna (ej: 3)
* string: zonas que incluye (ej. “Balvanera, San Cristóbal” para la comuna 3)
* string: nombre del director
* string: domicilio completo del CGP (ej: Junín 521)
* string: teléfono del CGP (4375-0644/45)
* lista de “serviciosDTO”: array de servicios que contiene
  + string nombre del servicio (ej: Atención ciudadana)
  + lista de “rangos servicio DTO”: Array con días de servicio que contiene
* int: número de día de la semana (ej: 1 = Lunes, 2 = Martes, etc.)
* int: horario desde (9)
* int: minutos desde (0)
* int: horario hasta (18)
* int: minutos hasta (0)

### Caso 3: Bancos

Para consultar las entidades bancarias se cuenta con un servicio desarrollado en forma externa. El servicio acepta como parámetro un string que representa el nombre del banco y otro con un nombre de servicio, y devuelve los bancos que cumplen ambos criterios (AND). La respuesta es en el siguiente formato JSON:

[

{ "banco": "Banco de la Plaza",

"x": -35.9338322,

"y": 72.348353,

"sucursal": "Avellaneda",

"gerente": "Javier Loeschbor",

"servicios": [ "cobro cheques", "depósitos", "extracciones", "transferencias", "créditos", "", "", "" ]

},

{ "banco": "Banco de la Plaza",

"x": -35.9345681,

"y": 72.344546,

"sucursal": "Caballito",

"gerente": "Fabián Fantaguzzi",

"servicios": [ "depósitos", "extracciones", "transferencias", "seguros", "", "", "", "" ]

}, ...

]

Se debe adaptar el resultado de la búsqueda creando o actualizando la lista de puntos de interés en el repositorio de la aplicación a diseñar (el caso 1).

**Requerimientos detallados**

* ABMC (Alta, Baja, Modificación, Consulta) de puntos del sistema
* Consulta (Búsqueda) de puntos extendida: se debe lograr que cuando se realice una consulta, se busquen puntos en todos los orígenes de datos. Se debe diseñar teniendo en cuenta que nuevos orígenes de datos pueden incorporarse en el futuro.
* Desarrollar casos de prueba para los puntos anteriores. Considerar:
  + El componente externo del caso 2 es lento (ejecutarlo es al menos 10 veces más lento que realizar la consulta contra nuestros datos)
  + El servicio externo del caso 3 es rápido, pero tiene un costo de servicio, que se imputa por cada pedido realizado al mismo.

# Entrega 3

**Fecha de entrega: 1 de julio**

El Gobierno está conforme con el avance del proyecto por lo que generó una reunión en la que participó parte de nuestro Equipo y relevó las necesidades, generando la siguiente especificación de Requerimientos que fue validada por el Cliente.

El Sistema deberá permitir:

* + Gestionar diferentes Perfiles de Usuario. Actualmente solo existen 2 Tipos, Administrador y Terminal.
  + Notificar al Administrador por mail en caso que una búsqueda demore más de X segundos. Este valor debe ser parametrizable.
  + Almacenar los resultados de las búsquedas. Se quiere conocer qué es lo que se buscó (frase buscada), la cantidad de resultados devueltos por el Sistema y el tiempo que demoró dicha consulta.
  + Obtener un reporte que permita totalizar la cantidad de búsquedas por Fecha. Ejemplo:

|  |  |
| --- | --- |
| Fecha | Cantidad Búsquedas |
| 01/04/2016 | 990 |
| 02/04/2016 | 1500 |
| 03/04/2016 | 1250 |

* + Generar un reporte que muestre la cantidad de resultados por búsqueda y por Terminal.

Ejemplo:

Parciales por Terminal

Usuario: terminalAbasto

|  |  |
| --- | --- |
| Fecha | Cantidad Resultados Parciales |
| 01/04/2016 | 30 |
| 02/04/2016 | 40 |
| 03/04/2016 | 50 |

Totales por Usuarios

|  |  |
| --- | --- |
| Usuario | Cantidad Resultados Totales |
| terminalAbasto | 120 |
| terminalFlorida | 50 |
| terminalTeatroColon | 70 |

* Activar/desactivar las acciones por Terminal en forma dinámica.

*Se deberán presentar los siguientes documentos:*

* *Modelado de clases*
* *Justificación de decisiones de diseño en el modelado de clases*
* *Casos de prueba previstos en Entrega 1 y Entrega 2*
* *Casos de prueba completos para requerimientos de Entrega 3*

*Se deberá contar con las funcionalidades correspondientes a las entregas 1, 2 y 3 totalmente implementadas y funcionando sin defectos (sólo se aceptarán defectos de criticidad baja).*

*Se deberá contar la implementación bajo un patrón arquitectónico MVC.*

*Se deberá contar con una interfaz de usuario que permita realizar la ejecución de las diferentes funcionalidades (no se requiere un diseño de interfaz de usuario, sólo la posibilidad de poder acceder a la funcionalidad en dicha interfaz).*

# Entrega 4

**Fecha de entrega: 26 de agosto**

En esta etapa se requiere actualizar los POIs a partir de diferentes orígenes y además asignar acciones a los Usuarios del Sistema. Para este punto un colega con experiencia nos aconseja generar Procesos de actualización asincrónicos.

El Administrador podrá disparar estos procesos y se deberán guardar los resultados de la ejecución.

1. Un usuario Administrador podrá seleccionar un proceso de una lista de procesos disponibles para ejecutarlo.
2. El resultado del proceso se debe almacenar en una lista donde se asocia la fecha y hora de inicio de ejecución, fecha y hora de fin de ejecución, el proceso ejecutado, el usuario, el resultado (ok, error) y opcionalmente un mensaje de error

### Proceso 1: Actualización de Locales Comerciales

Este proceso debe actualizar la lista de locales comerciales y la lista de palabras clave de asociadas a cada uno. Para este proceso se cuenta un archivo de texto plano que se nos enviará semanalmente con el siguiente formato:

nombre de fantasía;palabras clave (separadas por espacios)

Ejemplo:

Carrousel;colegio escolar uniformes modas

Se debe considerar que si el local no existe se debe crear. En cambio si existe se debe actualizar el listado de palabras claves.

Ejemplo: si Carrousel tuviera como palabras clave: 1) colegio, 2) uniformes, 3) blazers y viniera la línea con 4 palabras clave (colegio, escolar, uniformes y modas), la colección de palabras clave debería quedar de la siguiente manera: colegio, escolar, uniformes, modas (no importa el orden).

### Proceso 2: Baja de POIs

Este proceso tiene la responsabilidad de dar de baja un POIs que ya no esté activo. Para ello se cuenta con un Servicio REST[[6]](#footnote-6) que nos brinda el Gobierno de la Ciudad que devuelve:

* Valor de búsqueda del POI
* Fecha de baja

El proceso debe buscar el POI utilizando lo desarrollado en la Entrega I.

### Proceso 3: Agregar acciones para todos los usuarios

Este proceso tiene como objetivo agregar una lista acciones que puede realizar cada Usuario en el Sistema. Este proceso está vinculado a la Entrega 3. Dada una lista de Acciones por Usuario el proceso debe asignar/actualizar las acciones que puede realizar cada Usuario. Se debe considerar la posibilidad de deshacer esta acción (undo) y que vuelva todo al estado original.

### Proceso 4: Definición de un proceso múltiple

El proceso múltiple debe permitir anidar varios procesos a la vez. Al ejecutar un proceso múltiple se envía a ejecutar todos los procesos asociados al proceso múltiple.

### Manejo de resultados / errores en la Ejecución del Proceso

Al ejecutarse el proceso puede ocurrir:

* Que el proceso se ejecute correctamente.
* Que el proceso falle. En ese caso dependerá de lo que configure el Usuario:
  + Se envía un mail al Usuario avisando el fallo.
  + Se reintenta n veces el proceso. Si se ejecuta n veces en forma errónea queda en estado “error”.
  + No se realiza ninguna acción.
* Siempre se debe almacenar el resultado de Ejecución.

# Entrega 5

**Fecha de entrega: 23 de septiembre**

Esta entrega incluye:

* Modelado de datos
* Construcción de componente para servicios REST
* Diseño e implementación de interfaz de usuario.

### 5.1. Modelado de datos

Realizar el modelado de datos mediante un Diagrama de Entidad-Relación (DER) justificando las decisiones sobre el mismo. Para su construcción utilizar la técnica de mapeo objetos-relacional (ORM) justificando cada decisión tomada.

No es obligatorio realizar la persistencia sobre un motor de bases de datos. Esto será requerido en entregas siguientes.

### 5.2. Servicio REST

Deben generar un componente en el servidor que provea los siguientes servicios:

* Búsqueda de puntos de interés
  + recibe un criterio o una combinación de ellos,
  + devuelve la lista de puntos de interés
* Historial de consultas realizadas
  + se pasa el criterio de búsqueda (por rango de fechas / usuario)
  + devuelve la lista de consultas realizadas

### 5.3. Diseño e implementación de Interfaz de Usuario

Se debe realizar el diseño e implementación de las siguientes interfaces de usuario valorando la usabilidad y experiencia de usuario (UX).

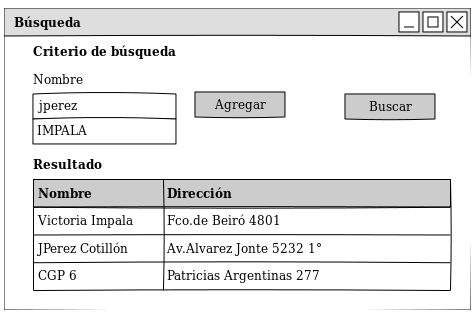
Se debe analizar si las interfaces presentarán un comportamiento diferente en caso de ser un “Usuario Administrador” o un “Usuario Terminal”.

#### 5.3.1. Pantalla de Login

Realice el diseño e implementación de la pantalla de Login. Justifique las decisiones de diseño tomadas.

#### 5.3.2. Búsqueda de POI

Se pide que diseñe la pantalla de búsqueda de puntos de interés, respetando el *layout* definido



En este ejemplo la búsqueda se hace por valor “jperez” o “IMPALA”. Se pueden agregar tantos valores como el usuario quiera: al presionar Agregar tiene que mostrarse un cuadro de diálogo para aceptar un texto e incorporarlo a la lista de nombres a buscar (siempre considerar que la búsqueda incluya todos los valores ingresados mediante un OR).

Al hacer click en alguno de los puntos de interés, se debe mostrar el detalle de la información propia de cada POI. **Nota:** no debe repetir “ideas” al desarrollar cada una de las pantallas.

Información a mostrar para una **línea de colectivo**:

* Icono
* Número de línea

Información a mostrar para un **CGP**:

* Icono
* Dirección
* Zona que le corresponde
* Lista de servicios que presta con su horario de atención

Información a mostrar para una **Sucursal de un Banco**:

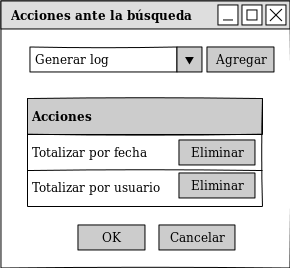
* Icono
* Dirección
* Zona que le corresponde
* Lista de servicios que presta

Información a mostrar para un **Local comercial**:

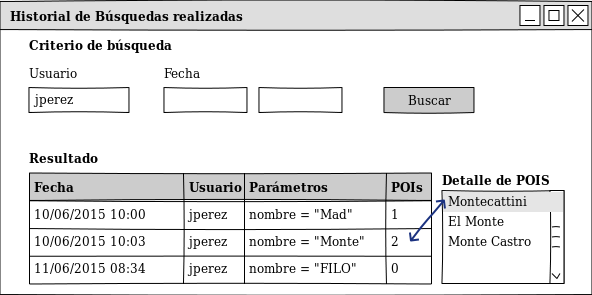
* Icono
* Dirección
* Nombre comercial o de fantasía
* Rubro

#### 5.3.3. Pantalla de Configuración de Acciones ante una Consulta

En esta pantalla se debe configurar las acciones que debe hacer el sistema ante una búsqueda, y asociarlas a un usuario. También se pide que valide que no se cargue dos veces la misma acción.



#### 5.3.4. Pantalla de Historial de Búsquedas Realizadas



La búsqueda se puede hacer:

* 1. por usuario exacto
  2. por rango de fechas entre las cuales se buscará consultas que se hayan hecho. Puede no ingresarse fecha desde o fecha hasta, en esos casos se trabajará con cota superior o inferior, respectivamente. Ej: si se ingresa fecha hasta 11/06/2015, se buscarán todas las consultas realizadas hasta el 11/06/2015

# Entrega 6: Persistencia a un medio relacional

**Fecha de entrega: 21 de octubre**

El objetivo de esta fase del proyecto consiste en persistir en un medio relacional todos los objetos de puntos de interés instanciados a partir de las clases del modelo. El esquema relacional deberá estar basado en el modelo de objetos del sistema y debe reflejar todas sus relaciones de herencia, composición y agregación.

**El sistema deberá persistir:**

* Puntos de Interés (PoI)
* Resultados de búsquedas (parámetros y conjunto de PoIs)
* Acciones de búsqueda
* Usuarios del sistema

**Requerimientos No Funcionales:**

Se pide contar con el soporte de un producto ORM con el fin de evitar la falta de concordancia (impedance mismatch) entre paradigmas.

**Comunicación del Diseño:**

* Diagrama de Entidad Relación (DER):

a.   Entidades,

b.   sus atributos y dominios (tipos de datos) y

c.    las relaciones entre entidades, incluyendo restricciones (constraints) para garantizar la consistencia del modelo.

* Explicar qué adaptaciones fueron necesarias en el modelo de objetos para conseguir persistencia relacional, si está normalizado y qué decisiones de diseño debió tomar, por ejemplo: desnormalizar, estrategia para herencia, objetos embebidos, cuáles objetos no debieron ser persistidos y otros.

**Conjunto de pruebas unitarias:**

Para validar este requerimiento, la entrega (release) deberá incluir el siguiente conjunto inicial de test unitarios:

1. Obtener un POI, modificar sus coordenadas geográficas, persistirlo, recuperarlo y verificar que las coordenadas sean las ingresadas en la última modificación.
2. Crear un nuevo PoI, persistirlo, recuperarlo, eliminarlo y al solicitar nuevamente su recuperación, la respuesta deberá ser que no existe (null).
3. Realizar una búsqueda, persistirla, recuperarla y verificar que corresponda al objeto de esa búsqueda e incluya referencias a los PoI.
4. Dar de alta un usuario, persistirlo, recuperarlo, realizar una modificación en el nombre de usuario, recuperarlo y verificar que el cambio esté presente.

1. <https://es.wikipedia.org/wiki/Punto_de_inter%C3%A9s> [↑](#footnote-ref-1)
2. Un proyecto de estas características podría fallar por diversos motivos humanos y técnicos, no relacionados al software en sí: roturas, robos, falta de conexión, problemas de energía, etc. Sin embargo, a fines del TP no serán considerados. [↑](#footnote-ref-2)
3. Como mínimo, si el texto de búsqueda está incluido en el nombre del servicio, debe encontrarlo. Si se desea realizar una búsqueda más sofisticada, se recomienda utilizar bibliotecas especiales que provean algoritmos como por ejemplo, [la distancia de Levenshtein](https://es.wikipedia.org/wiki/Distancia_de_Levenshtein) [↑](#footnote-ref-3)
4. Si bien este requerimiento se puede resolver mediante un motor de búsqueda de texto libre (búsqueda full text search, como Lucene/ElasticSearch o Postgre Full Text Search), se espera que se resuelva este requerimiento empleando exclusivamente el paradigma de objetos. [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)
6. Pueden utilizar estos servicios que fueron desarrollados por colegas de la cátedra:

   <http://demo3537367.mockable.io/trash/pois>  (devuelve el identificador de POI + fecha de baja)

   <http://demo3537367.mockable.io/trash/pois_bad>   (devuelve BadRequest + json en el body con la descripción, lo pueden usar para probar una url que falla) [↑](#footnote-ref-6)