UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Titulo: Git para el manejo de la legislación municipal
Autores: Cañibano, Tomás
Director: Molinari, Lía Hebe
Codirector: Luengo, Miguel
Asesor profesional: -
Carrera: Licenciatura en Sistemas

Resumen

Esta tesina analiza y propone una solución pragmática a los problemas de representación, manejo de cambios, creación, publicación y acceso de la legislación mediante la utilización de herramientas del ámbito de la industria del software. Específicamente Git, un sistema de versionado de código, y GitHub, el servicio de *hosting* de repositorios más grande y popular del mundo actualmente. Para llevar a cabo el análisis de la solución se tomó el caso del municipio de La Plata, haciendo foco particularmente en sus ordenanzas. Esta propuesta consta de 3 partes: la creación de un repositorio Git con la historia completa de la legislación municipal, la utilización de GitHub como plataforma colaborativa, y finalmente, el desarrollo de aplicaciones por parte de actores privados a partir de los datos del repositorio. Se va a enmarcar el desarrollo de la tesina dentro de los movimientos internacionales más importantes a nivel gubernamental: gobierno abierto, gobierno electrónico y datos abiertos, argumentando el por qué de apuntar hacia la implementación de sus principios y cómo el adoptar la solución aquí propuesta aportaría a la realización de ese objetivo de una manera eficiente y aplicable de forma general.

Palabras Claves

Gobierno Abierto, Gobierno Electrónico, Datos Abiertos, Versionado de código, Git, GitHub, Legislación, La Plata

Conclusiones

Si bien es factible el desarrollo de un sistema más enfocado a este caso de uso, la solución planteada sirve lo suficientemente bien para este propósito. Aprovechando herramientas usadas en un problema similar, se ahorra esfuerzo, riesgos y costos de desarrollo, y por lo tanto se facilita su implementación de forma general. La principal desventaja es que no se cuenta con un caso de uso concreto, del cual se podrían sacar conclusiones más fundamentadas y correcciones. Se espera que esta idea aporte a los pilares del gobierno abierto: transparencia, participación y colaboración.

Trabajos Realizados

- Investigación de gobierno abierto, gobierno electrónico y datos abiertos
- Estudio de la legislación del partido de La Plata
- Análisis de Git y GitHub
- Definición de una solución a los problemas del manejo de la legislación haciendo uso de los sistemas y conceptos antes analizados
- Descripción de los diferentes aspectos de la misma junto a la enumeración de las posibilidades que se abren con su

Trabajos Futuros

- Analizar la posibilidad de representar el proceso legislativo
- Desarrollar interfaz por sobre GitHub que simplifique el uso y aplicaciones que consuman los datos del repositorio
- Implementar la solución y analizar la experiencia, de forma de corregir defectos y definir un proceso de traspaso hacia este nuevo sistema

Fecha de la presentación: Marzo 2016

Universidad Nacional de La Plata

Facultad de Informática

Git para el manejo de la legislación municipal

Tesina de Licenciatura en Sistemas

Alumno: Tomás Cañibano

Director: Mg. Molinari, Lia Hebe

Codirector: Lic. Luengo, Miguel

Marzo 2016

Agradecimientos

A mi familia, amigos, país, y a aquellos sobre cuyas ideas u obras fue hecha esta tesina.

Abstract

Esta tesina analiza y propone una solución pragmática a los problemas de representación, manejo de cambios, creación, publicación y acceso de la legislación mediante la utilización de herramientas del ámbito de la industria del software. Específicamente Git, un sistema de versionado de código, y GitHub, el servicio de *hosting* de repositorios más grande y popular del mundo actualmente. Para llevar a cabo el análisis de la solución se tomó el caso del municipio de La Plata, haciendo foco particularmente en sus ordenanzas. Esta propuesta consta de 3 partes: la creación de un repositorio Git con la historia completa de la legislación municipal, la utilización de GitHub como plataforma colaborativa, y finalmente, el desarrollo de aplicaciones por parte de actores privados a partir de los datos del repositorio. Se va a enmarcar el desarrollo de la tesina dentro de los movimientos internacionales más importantes a nivel gubernamental: gobierno abierto, gobierno electrónico y datos abiertos, argumentando el por qué de apuntar hacia la implementación de sus principios y cómo el adoptar la solución aquí propuesta aportaría a la realización de ese objetivo de una manera eficiente y aplicable de forma general.

1. Introducción

En esta primer parte se va a dar una descripción general de la tesina. Esto incluye su motivación, objetivos y estructura.

Motivación

Cualquier sociedad moderna depende para su funcionamiento de leyes que se presumen conocidas por sus integrantes, de forma de que ninguno pueda alegar su desconocimiento como justificativo de alguna conducta que la infrinja. Esto se fundamenta en la publicación de todos los actos de gobierno, mediante la cual en teoría se informaría a todas las personas de las mismas. En la realidad esto presenta algunos inconvenientes. Por nombrar algunos de los principales, primero, la llamada **inflación legislativa**, que trata del volumen cada vez mayor de normas legales generadas por parte de las diferentes instituciones del estado, que resulta en la imposibilidad de conocerlas a todas, y por lo tanto en su desvalorización. La **publicación deficiente** es otro problema, ya que, en el caso que se haga, muchas veces es fuera de un tiempo adecuado y por medios tradicionales que no llegan a la gran parte de la población. [1] Finalmente la **dificultad para conocer la última norma vigente**, teniendo en cuenta que las leyes sufren modificaciones a lo largo del tiempo y no se suele contar con una versión consolidada de las mismas, que surge de aplicar sobre el texto original las modificaciones para obtener como resultado el texto vigente.

Esto es cierto también en el municipio de La Plata. Por empezar, el boletín municipal, donde se da conocimiento a la población de los actos realizados por el gobierno del partido, tiene un tiempo de publicación, que además de variable, es diferente al de la promulgación de las normas. [2] Esto, que se repite en otros partidos de la provincia de Buenos Aires y del país, genera situaciones donde esa misma norma en ese período de tiempo es inefectiva justamente por no ser de conocimiento público, como indican diferentes fallos. [1] Otra alternativa para conocer las normas, como es el digesto digital, un sitio web del municipio, es actualizado luego de la publicación del boletín con el contenido de éstas, lo que le otorga una ventaja relativa en tanto que no involucra una difusión más rápida. Por otro lado, se cuenta con un digesto, independiente del digesto digital, donde se guardan todas las normas en formato papel. Debido a esto la utilización del mismo tiene las dificultades y sigue los patrones tradicionales de cualquier biblioteca, por ejemplo el uso de índices para localizar legislación, y la dificultad de registrar cambios en las normas. Este es un problema recurrente con cualquier legislación. Ya que las ordenanzas pueden ser modificadas por ordenanzas posteriores, se cuenta por un lado con la ordenanza original, la cual no refleja el estado actual y real de la misma, y por otro lado con los cambios sobre la misma mencionados en el texto de otras ordenanzas. Pocas veces se produce un texto ordenado que “pase en limpio” a ésta, creando una nueva y derogando la anterior. [2] Esto sí es realizado por conveniencia en el digesto digital, pero provoca que se pierda la historia de la norma, ya que no se lleva un registro de los diferentes cambios que se aplican. Esto que a priori no tendría tanta importancia, cobra peso cuando se consideran

situaciones relativas a la labor legislativa o por ejemplo un juicio en donde se debe juzgar un hecho producido un tiempo atrás, donde las leyes eran diferentes a las actuales. Para esto es necesario conocer la legislación tal como era en ese instante, lo cual puede llegar a ser un proceso dificultoso si se hace mediante mecanismos tradicionales y sin ayuda de un sistema diseñado para esto que facilite la tarea.

Por otro lado, si bien el acceso libre a la información del gobierno está garantizada por la constitución, [3] las nuevas tecnologías de la información y comunicación (TIC) exigen nuevos enfoques para garantizar y potenciarlo, particularmente mediante el uso de internet. En muchos países del mundo ya se permite el acceso a los documentos oficiales a través de internet, mediante iniciativas a nivel nacional como regional para proveer un mejor acceso a la información legal, aunque en varios no se cuenta con mecanismos eficaces para el acceso a las leyes. [4] En el caso de los municipios de la provincia de Buenos Aires, está contemplada como necesaria la publicación en internet del boletín municipal. Esto se da dentro de un movimiento de carácter internacional hacia la apertura de los gobiernos, principalmente a través de la publicación de sus datos, pero también mediante la simplificación de los procesos mediante los cuales interactúa con sus ciudadanos. En un gran número de estas iniciativas juegan un rol principal las TIC. [5]

Dados los problemas actuales generales en el manejo de las normas gubernamentales y los nuevos enfoques tendientes a facilitar el acceso a los datos del gobierno y la interacción con el mismo, se torna necesario definir soluciones factibles que palien estas deficiencias. Esta factibilidad va a estar dada en gran medida por el esfuerzo requerido para implementar la solución, teniendo en cuenta que no todas las instituciones gubernamentales poseen la misma cantidad de recursos para asignar a este fin.

Objetivo

El objetivo de la tesina es proponer una solución pragmática al problema del manejo de la legislación utilizando dos sistemas propios del desarrollo de software, Git y GitHub, analizando el caso particular de aquella del municipio de La Plata, y abordando a la misma a través del marco de gobierno abierto, gobierno electrónico y datos abiertos. Esta solución deberá permitir representar la historia completa de la legislación, acceder a la información de ella en cualquier punto del tiempo, proveer un mecanismo de *feedback* y colaboración entre los ciudadanos y concejales, similar al presente en desarrollos *open-source* a través de plataformas como GitHub, y promover el desarrollo de aplicaciones por parte de terceros que hagan uso de los datos provistos por el repositorio. Todo esto se relaciona íntimamente con los 3 pilares del gobierno abierto: transparencia, participación y colaboración. *Transparencia*, ya que la legislación y el trabajo de cada concejal quedaría más accesible y visible para cualquier persona; *Participación*, ya que permitiría a la ciudadanía sugerir mejoras o *feedback* sobre la legislación vigente y futura; *Colaboración*, ya que el libre y fácil acceso a los datos permitiría el uso de los mismos de nuevas formas por parte de toda la sociedad, y porque el uso de una plataforma como GitHub puede fomentar el trabajo y comunicación entre concejales y

ciudadanos por igual, quienes de esta forma podrían involucrarse activamente en el proceso legislativo.

Estructura de la tesina

Parte 1 - Introducción: Esta primer parte sirve de introducción al trabajo y explica la motivación y objetivos detrás de su desarrollo.

Parte 2 - Gobierno Abierto: Se van a dar las definiciones de gobierno abierto y sus pilares, de gobierno electrónico y de datos abiertos, así como también sus características más importantes y ejemplos a nivel nacional e internacional.

Parte 3 - Municipio de La Plata: En esta parte se va a dar una descripción del municipio de La Plata y su sistema legislativo, junto a las problemáticas actuales tanto del mismo, como generales de los partidos.

Parte 4 - Git y GitHub: Va a empezar por desarrollarse lo que significa el versionado de código, para luego definir Git y enumerar sus características más salientes. Por otro lado se va dar una visión general de GitHub, de forma de poder conocer las funcionalidades provistas por la plataforma.

Parte 5 - Sistema propuesto: Aquí va a darse el contexto y estado del arte en cuanto a sistemas para manejo de la legislación. Una vez hecho esto se va a dar la descripción de la solución, sus fundamentos, los cuales a su vez van a ser desarrollados, y el alcance de la misma.

Parte 6 - Casos de referencia: Se cuenta con varios casos de uso similares en GitHub, los cuales van a describirse de forma general.

Parte 7 - Historia del repositorio: El primer aspecto a analizar del repositorio va a ser la historia del mismo, desde qué representará cada *commit* hasta cómo generar y actualizar la misma.

Parte 8 - Organización del repositorio: Como segundo aspecto se va a enumerar cómo puede estar organizado el repositorio, esto incluye los nombres de los archivos y la estructura de carpetas del mismo.

Parte 9 - Formato de los archivos: El aspecto final a analizar del repositorio va a ser el formato de los archivos con las ordenanzas, para esto se van a definir criterios para preseleccionarlos y estándares para comparar a algunos de los formatos posibles.

Parte 10 - Usos de GitHub: Se van a identificar y definir las diferentes posibilidades y dificultades que surgen del uso de GitHub como plataforma donde albergar el repositorio.

Parte 11 - Aplicaciones de terceros: En esta parte se van a enumerar algunas de las posibilidades que se abren para el desarrollo de aplicaciones a partir de los datos contenidos en el repositorio.

Parte 12 - Conclusión: Finalmente aquí se va a dar una conclusión en base a lo desarrollado en la tesina.

2. Gobierno Abierto

Todo sistema de software tiene un propósito dictado por una necesidad de una parte de la sociedad. A menudo estas necesidades, al igual que los sistemas que tratan de satisfacerlas, están contenidas dentro de un marco teórico que intenta explicar esa parte de la realidad. Para poder pensar y realizar el diseño de uno de estos sistemas y un mejor desarrollo, es importante conocer su marco, de forma de comprender el problema a resolver de una mejor manera, guiar las ideas y nutrirse de aquello ya hecho por otros.

Para el caso de esta tesina, la cual involucra una parte central del país como son las leyes, se van a analizar los principales movimientos e iniciativas gubernamentales actuales, como son *gobierno abierto*, *gobierno electrónico* y *datos abiertos*. Ya que gobierno abierto es la más general y la que mayor puntos de contacto tiene con las otras, es la que primero se nombra y la que se utiliza para designar a esta parte. Eso no quiere decir que las demás sean un subconjunto estricto de gobierno abierto, cada una posee características particulares y conforman un cuerpo de conocimiento amplio en sí mismas. Como se va a intentar justificar, el sistema propuesto puede relacionarse con todas ellas, y así mismo, realizar aportes para su mayor desarrollo donde sea que se implante.

¿Qué es Gobierno Abierto?

Si bien existen diversas definiciones, en su sentido más básico, es la idea de que los ciudadanos tienen el derecho de acceder a los documentos y procedimientos del gobierno para permitir un control efectivo sobre el mismo. [5] Aunque el término es relativamente reciente, siendo usado por primera vez en la década de 1950, y actualmente su significado se ha corrido del original para pasar a estar asociado fuertemente con las nuevas tecnologías de información, [6] sus principios forman parte de los valores democráticos desde hace mucho tiempo. [7]

Esta noción de que la ciudadanía tiene derecho a controlar y participar en el gobierno, si bien presente anteriormente en diferentes sociedades, tiene sus raíces formales en el iluminismo de los siglos XVIII y XIX, en el contexto del nacimiento de las primeras sociedades democráticas. El aumento en la transmisión de ideas en forma de libros, periódicos y ensayos, como consecuencia de las nuevas tecnologías de impresión, tuvo un papel central en el desencadenamiento de las revoluciones de Estados Unidos y Francia, a partir de las cuales se consolidaron derechos como la libertad de expresión, reunión y prensa en varios países occidentales. Esto formó la base de los que se consideran gobiernos abiertos, en contraposición a aquellos regidos por la doctrina absolutista donde primaba el secreto de estado. [8]

En el siglo XX, durante las décadas de los 50 y 60, surgió un movimiento de reformas gubernamentales, que incluyó a las llamadas leyes de libertad de información (*freedom of information laws*). Su objetivo era promover una cultura de gobierno transparente, accesible y responsable, para lo cual incluían elementos como el libre acceso a las discusiones,

deliberaciones y decisiones de los integrantes del gobierno o la posibilidad de dar testimonio, comentar o realizar un pedido. [8] Es en esta época donde se hace el primer uso conocido del término gobierno abierto. Se le atribuye a Wallace Parks, un miembro del subcomité para la información gubernamental, durante la década de 1950 en los Estados Unidos, en los debates previos a la promulgación del acta para la libertad de información (*Freedom of Information Act* o FOIA) en 1966. Parks creía que la información del Gobierno debía ser accesible por el público siempre, exceptuando casos limitados donde primaran otros derechos o intereses por sobre el interés del público en su libre acceso. De acuerdo a esta definición inicial, gobierno abierto sólo se trata de la libertad de información. [9] Leyes similares fueron promulgadas en diversos países posteriormente, como Dinamarca y Noruega en 1970, Francia y los países Bajos en 1978. En las décadas que siguieron el término fue usado menos frecuentemente, apareciendo esporádicamente en debates acerca de FOIA, como sinonimo de acceso público a información gubernamental previamente sin revelar, así como también en decisiones judiciales como referencia a transparencia gubernamental. [6]

No fue hasta la primer campaña presidencial de Barack Obama que el término empezó a ser usado ampliamente y con un significado diferente, mucho más ambicioso que el de Parks, ya que influenciado por las ideas centrales del movimiento *open source*, pasó a incluir la participación y colaboración activa de los ciudadanos en un amplio rango de propósitos, tomando a la tecnología como medio clave para llevarlos a cabo. [9] Posteriormente, en su primer día como presidente, emite un memorando sobre transparencia y gobierno abierto. En él, insta a la agencias y departamentos del poder ejecutivo a tomar acciones específicas, en un plazo de tiempo, para implementar los principios de *transparencia*, *participación* y *colaboración*, los cuales define como los pilares de un gobierno abierto. [10] De acuerdo a esto, el rol de un ciudadano dentro de un gobierno abierto no se limita al acceso de la información del mismo, sino que también abarca la posibilidad de por un lado participar aportando información, nuevas ideas y conocimiento, y por otro colaborar, pasando a tener un rol activo dentro de los diferentes procesos del gobierno

Es por esas razones que el término gobierno abierto se encuentra actualmente inherentemente relacionado a las TIC, y a menudo se lo usa como sinónimo del término Gov 2.0. Sin embargo éste se refiere más precisamente a la utilización de tecnologías abiertas y colaborativas para crear una plataforma a través de la cual el gobierno y los individuos puedan trabajar juntos para mejorar la transparencia y la eficiencia de los servicios del gobierno. Para esto, el gobierno hace uso de estándares abiertos y provee datos que usan los individuos para crear sitios, aplicaciones y demás para beneficio del público.

En conclusión, el significado del término gobierno abierto fue cambiando a través de los años, y aunque sigue teniendo el componente original de transparencia y rendición de cuentas de sus comienzos, se incluyen en el último tiempo más conceptos como la participación y colaboración por influencia de las TIC, las cuales están inherentemente asociadas en muchos usos del término.

Los 3 pilares

Más allá de la variedad de definiciones e interpretaciones de lo que significa el término gobierno abierto, [9] se suele decir que éste se basa en tres pilares, los cuales varían en alguno de ellos dependiendo de quién los define, aunque en general hay un cierto consenso sobre cuáles son. Transparencia y participación son los que primero se nombran y están presentes en casi cualquier lista. La mayoría suele incluir colaboración como el tercer pilar, aunque puede llegar a encontrarse la rendición de cuentas reemplazando a éste. [11] [12] A continuación se van a describir transparencia, participación y colaboración, entendiendo que son los comúnmente nombrados, [5] [7] [13] y los propuestos en el memorando de gobierno abierto emitido por el presidente Barack Obama. En parte estos conceptos son una influencia del movimiento *open source*, que es caracterizado como transparente, participativo y colaborativo, aunque igualmente estos son históricamente conceptos democráticos aplicados a procesos públicos como la votación, pero que ahora lo están siendo en contextos administrativos rutinarios dentro de los gobiernos. [13]

Es importante destacar el sentido de nombrar estos tres y en ese orden. Cada uno involucra un mayor compromiso por parte del público con el gobierno: con transparencia se obtiene visibilidad de lo que sucede dentro del mismo, luego con participación se permite el aporte del conocimiento que posee el público y finalmente con colaboración se incluyen a los diferentes actores que pasan a tener un rol activo dentro de los procedimientos del gobierno.

Transparencia

Para poder tener un gobierno responsable de sus actos, los ciudadanos deben saber cuáles son esos actos. Una democracia está basada en el principio de que la autoridad delegada en el gobierno se deriva del consentimiento de aquellos que son gobernados. [14] Para mantener esa base de consenso, la información juega un rol esencial, siendo necesaria para la formación de opinión, la evaluación de alternativas, hasta para la toma de decisiones. El término transparencia se refiere al acceso por parte del público a información actualizada, entendible, relevante, de calidad y veraz de las actividades gubernamentales. No se puede contar con un gobierno responsable y que responda a la voluntad pública sin información. En este sentido la transparencia describe hasta qué punto los integrantes del gobierno disponibilizan los datos y documentos que el público necesita para poder evaluar las acciones del mismo. [13]

Los mecanismos de acceso a esta información deben incluir su liberación en respuesta a un pedido así como también la publicación por iniciativa propia por parte de las instituciones. [12] Como caso emblemático se cuenta con las leyes de libertad de información como la FOIA, que provee a cualquier persona el derecho de acceder a información de nivel federal, debiendo cada agencia del gobierno de los Estados Unidos responder a ese pedido, excepto en casos especiales. [15]

El concepto de transparencia está íntimamente relacionado con el de rendición de cuentas, siendo la primera normalmente mencionada como una condición necesaria, pero no suficiente, para lograr la segunda. El concepto de rendición de cuentas es que aquellos con responsabilidades públicas deben rendir cuentas de sus acciones a la ciudadanía tomando

responsabilidad por ellas y explicandolas, y en consecuencia, los ciudadanos deberían ser capaces de influenciar la dirección del gobierno evaluando el progreso y asignando culpas. En consecuencia la rendición de cuentas está formada por dos componentes, una explicación de las actividades gubernamentales donde se justifiquen las acciones, y un mecanismo de sanciones con el cual el público pueda reaccionar a las mismas. [16] La transparencia permite una más efectiva rendición de cuentas por parte del gobierno, ya que se permite conocer qué es lo que está haciendo el gobierno y cómo, al echar luz sobre aquellos procesos hasta el momento oscuros para el resto de la sociedad. Los datos sobre el rendimiento de los servicios públicos permiten medir la efectividad de las políticas implementadas e informarla a la gente. Al poder supervisarse se genera por un lado un mayor incentivo para realizar las funciones gubernamentales correctamente, y además se facilita la detección y notificación de errores, ineficiencias y hechos como corrupción. Todo esto incrementa la efectividad del gobierno.

También, en relación a los demás pilares, el acceso a la información del gobierno permite una mayor y mejor participación pública. El tener datos confiables habilita al público a realizar aportes mejor informados, potenciando la toma de decisiones dentro del gobierno. Además pueden utilizarse de maneras no pensadas originalmente, creando nuevas interpretaciones de los mismos que tengan implicaciones de relevancia o agregándoles valor, por ejemplo creando aplicaciones que se alimenten de ellos. Finalmente, puede tener un impacto positivo en el desarrollo económico, no solo para aquellos que puedan hacer un uso directo de los datos para producir bienes y servicios a partir de ellos, sino que ya que se cuenta con más información del gobierno, esto lo hace más predecible, y se cuenta con una mayor consciencia de la situación actual. [5]

Por ende, el poder contar con información de calidad es un requisito para la transparencia gubernamental, y la transparencia es una herramienta para poder contar con un gobierno responsable. [17]

Participación

Este pilar se basa en que los derechos de los ciudadanos van más allá del acceso a la información, e incluyen el proveer aportes y *feedback*. Esto implica buscar activamente medios para incentivar a los ciudadanos a debatir sobre lo público y cómo aumentar y mejorar las oportunidades de participación. Además deben establecerse diversos espacios y mecanismos que permitan esta participación por iniciativa propia, así como también el permitir generar debates sobre diversos temas de interés donde se aporten opiniones e ideas. [12]

Las primeras formas de democracia en la antigua Grecia pueden definirse como democracias participativas. En estas, los ciudadanos discutiendo y deliberando como pares se involucraban en la toma de decisiones sobre los diferentes asuntos comunes. Este tipo de participación requiere conocer más a fondo las perspectivas de los demás y un lugar donde intercambiarlas. Con el paso del tiempo estos procesos se tornaron imposibles salvo para las comunidades más pequeñas, hasta que la llegada de las nuevas tecnologías permitieron a muchos creer en la posibilidad de una renovada participación cívica en los procesos políticos. [13]

La participación ciudadana aumenta la efectividad de los gobiernos, ya que se benefician del conocimiento ampliamente distribuido en la sociedad, de la capacidad de generar nuevas ideas y la habilidad de ejercer control sobre el funcionamiento estatal. Además, genera otros efectos positivos demostrados, como la reducción de la desconfianza en el gobierno, y dependiendo del tipo de mecanismo participativo usado, la obtención de nueva información valiosa o el aumento de la experiencia de los individuos en distintas áreas. [18]

Colaboración

La idea de este pilar es que la ciudadanía debería tener un rol activo en los procesos del gobierno. Va más allá de la inclusión de los aportes de los ciudadanos, para enfatizar sus roles formales en los procedimientos y la implementación de políticas del gobierno. [7] A diferencia de los anteriores pilares, la colaboración no fue asociada tradicionalmente con la teoría política democrática, sino que en parte responde a la influencia del movimiento *open source*, en el cual cumple un rol central. Se diferencia de otras prácticas participativas y deliberativas en que estas permiten el intercambio de ideas y puntos de vista pero están alejadas de la toma de decisiones, mientras que la colaboración implica reunir un grupo de ciudadanos experimentados para la definición de soluciones que van a ser implementadas. Esto se basa en la idea que se puede aumentar la efectividad de los gobiernos, sobretodo en la solución de problema complejos, utilizando personas de diferentes sectores de la sociedad para que aporten experiencia, imparcialidad, recursos, disciplina y tiempo para la solución de estos problema públicos. [13]

Gobierno Electrónico

A pesar de que las definiciones varían dependiendo de la fuente y no hay una aceptada universalmente, lo que lleva a ciertas críticas por la vaguedad en el uso del término, [19] [20] puede definirse a gobierno electrónico (*e-government* o EG) como el uso por parte del gobierno de las TIC, principalmente internet, para apoyar y mejorar el desarrollo de políticas y operaciones del gobierno, comprometer a los ciudadanos, y proveer servicios completos y a tiempo. [21]

La historia del término data de la década de 1990 cuando empieza a aparecer en la literatura académica y no académica, contrastando con el poco interés en los sistemas utilizados por el gobierno previamente y, por lo tanto, con lo escaso de las publicaciones al respecto. A partir de la aparición de internet, y a pesar de la lenta respuesta en las investigaciones relacionadas a gobierno electrónico, empiezan a surgir experiencias como el portal Firstgov.gov de Estados Unidos, o Europa.eu. El alto interés en una reforma del gobierno a través de las TIC empieza a fomentar investigaciones en el tema, siendo apoyada explícitamente por la Unión Europea a partir de 1998, al igual que por cada país perteneciente a ella individualmente, en Estados Unidos a través de la NSF[1] en 1998, junto a investigaciones

[1] *National Science Foundation*

de gobiernos de nivel local y estatal. Además hay que sumar otras colaboraciones menos institucionalizadas que a menudo se formaron entre agencias gubernamentales y consultores académicos y comerciales para llevar adelante proyectos de gobierno electrónico. A partir de allí empezaron a surgir conferencias a nivel global dedicadas al gobierno electrónico, que derivaron en foros donde se presentan los últimos resultados de investigaciones, y *journals* especialmente dedicados al tema. [21]

De acuerdo a la definición dada y a otras, pueden identificarse 3 áreas de transformación características de un gobierno electrónico, interna, externa y relacional. La parte interna se refiere a la mejora de la eficiencia de los procesos y funciones dentro del gobierno mediante la utilización de TIC. Estas interconectan las diferentes agencias y departamentos que lo componen y de esa manera se reducen los tiempos de procesamiento entre ellos. Esto se logra al permitir un mejor flujo de la información, eliminando cuellos de botella y procedimientos burocráticos, así como también dentro de ellos al reducir tiempos relacionados al uso de datos, costos asociados al manejo de información y mejorando la precisión y velocidad de las tareas. El área externa se relaciona con el aumento de la transparencia del gobierno dando mayor acceso a datos e información generados y recolectados por él, y con las oportunidades para colaborar entre diferentes instituciones gubernamentales o con el resto de la sociedad. Finalmente para la parte relacional, la adopción de TIC abarca cambios en las relaciones entre los ciudadanos y los gobiernos, y entre diferentes naciones, tanto en los procesos democráticos como en las estructuras del gobierno. [22]

A su vez otro componente siempre mencionado cuando se habla de gobierno electrónico son los grupos con los cuales se realizan las interacciones y transacciones electrónicas y que constituyen su red de relaciones. Estos son los ciudadanos, empresas, gobiernos y empleados. Surgen así los cuatro distintos bloques de un gobierno electrónico: [19] [20]

- Gobierno a ciudadano (*Government-to-Citizen* o G2C): Involucra las interacciones de los ciudadanos con el gobierno. Esto incluye interacciones relacionadas con el acceso a información y servicios públicos, como pago de impuestos o distintos trámites personales [19]. Fomenta la rendición de cuentas, mejora los servicios públicos, y permite y refuerza la participación en la comunidad.

- Gobierno a empresas (*Government-to-Business*, o G2B): Involucra las interacciones entre las agencias gubernamentales y las empresas privadas. Esto incluye iniciativas en la venta de servicios y productos, como aprovisionamiento electrónico (*e-procurement*) o la creación de un mercado electrónico para las compras y ventas del gobierno, y servicios electrónicos para las empresas como para el pago de impuestos o el registro de informes financieros.

- Gobierno a gobierno (*Government-to-Government*, o G2G): Involucra la comunicación entre las diferentes agencias u organizaciones gubernamentales. Esto incluye tanto comunicaciones dentro de cada una, como entre ellas, abarcando los niveles nacional, provincial, municipal o con organizaciones extranjeras. Esto tiene como objetivo la

mejora de la eficiencia y efectividad de los procesos del gobierno mediante la comunicación y coordinación *online* entre cada agencia, que les permite compartir recursos e información.

- Gobierno a empleado (*Government to Employee* o G2E): Se refiere a la relación entre el gobierno y sus empleados. Permite la colaboración entre los diferentes empleados en cualquier momento, mediante el intercambio de información y conocimiento, así como también el acceso a aplicaciones e información relevante, o la posibilidad de proveer *e-learning*.

Es común que a la hora de especificar los bloques, en el caso de G2E, se lo omita, se lo incluya dentro de G2G o se lo considere dentro del concepto de gobernanza electrónica (*e-governance*). [22] [19] Al igual que con el término gobierno electrónico, hay un gran rango de definiciones para gobernanza electrónica. En primer lugar hay que diferenciar gobierno de gobernanza, el primero se trata de la institución pública que tiene autoridad sobre un cierto territorio, mientras que el segundo es un concepto más amplio que se refiere a "los procesos e instituciones, tanto formales como informales, que guían y restringen los actividades colectivas de un grupo", no necesariamente dentro del gobierno ya que es aplicable a otras instituciones como empresas o ONGs. Gobernanza trata de procesos a largo plazo, objetivos y resultados mientras que gobierno trata de decisiones inmediatas, reglas y salidas. En la mayoría de los casos se entiende que el concepto de gobernanza electrónica incluye al de gobierno electrónico. [20] Mientras que gobierno electrónico trata básicamente de los procesos y estructuras necesarias para proveer servicios online a los ciudadanos, gobernanza electrónica es un término más amplio que abarca un rango de relaciones y redes en el gobierno, por ejemplo con el público, ONGs y entidades privadas, relacionadas con el uso e impacto de la aplicación de TIC. Se dice que gobernanza electrónica está más enfocada en los procesos democráticos, [23] ya que agrega al concepto de provisión de servicios de gobierno electrónico la mejora de las instituciones y procesos democráticos y una mayor participación en la toma de decisiones por parte del público. [24]

Es importante mencionar que para la implementación de gobierno electrónico o gobernanza electrónica, el uso de tecnología es una condición necesaria pero no suficiente. Se requiere también un análisis y rediseño de los procesos y estructuras del gobierno. Entre las barreras internas para la adopción de gobierno electrónico se pueden nombrar la falta de una infraestructura de IT adecuada, la necesidad de proveer seguridad, privacidad y confidencialidad de los datos personales, la falta de personal de IT calificado, cuestiones organizacionales como la cultura, los procesos de negocio y las estrategias de *management*, y los costos operacionales. [25] También se necesita superar barreras externas al gobierno, como la exclusión digital, o la necesidad de generar incentivos para los ciudadanos para adoptar el uso de los servicios electrónicos. [26]

La situación en Argentina

La Argentina, de acuerdo a la encuesta del 2014 de Naciones Unidas para el desarrollo de gobierno electrónico, ocupa el puesto 46 a nivel mundial, lo que marca un avance de 10 lugares con respecto a 2012. Esto lo clasifica como un país con alto índice de desarrollo gobierno electrónico, siendo superado a nivel de Sudamérica sólo por Uruguay y Chile. El ranking lo encabeza Corea del Sur, con países como Australia, Francia, Japón, Estados Unidos, Reino Unido integrando el grupo de líderes.

Entre las medidas tendientes al desarrollo del gobierno electrónico más importantes dentro del país se encuentra el decreto 378/2005 que aprueba los Lineamientos Estratégicos que deberán regir el Plan Nacional de Gobierno Electrónico y los Planes Sectoriales de Gobierno Electrónico para Organismos de la Administración Pública Nacional. Los principios rectores del mismo son: mejor servicio a los habitantes y al ciudadano, mejor gestión pública, reducción de costos, transparencia, participación, integración, apoyo y desarrollo, e integración a la economía mundial. Para poder cumplir con estos objetivos, el plan prevé el uso de internet, servicios web y tramitación electrónica, entre otras cosas, y la definición de un sistema de atención en línea, la implementación de la tramitación electrónica de expedientes para trámites internos del estado nacional, el establecimiento de políticas de seguridad, la definición de estándares tecnológicos para la interoperabilidad, la articulación entre organismos, y la creación de un sistema de autenticación único para la identificación de los habitantes, ciudadanos y usuarios que deban operar con el estado nacional.

Datos abiertos

Datos abiertos (*open data*) es la idea de que ciertos datos deberían ser accesibles, usables, modificables y compartibles por cualquiera, sujetos cada uno a lo sumo a medidas para preservar su procedencia y su carácter de abierto. [27] Según la *Open Definition*, para que una obra, definido como la pieza de conocimiento que se transfiere, datos en este caso, sea abierta, debe cumplir con varios requisitos como estar disponible mediante una licencia abierta, ser de libre acceso y proveerse en un formato abierto. [28] En el caso de datos del gobierno, se utiliza el término *open government data* (OGD). Se puede decir que está compuesto a su vez por dos elementos, datos abiertos, que se definió anteriormente, y datos gubernamentales (*government data*), que es cualquier dato o información generada por las instituciones públicas. También se cuentan con varias definiciones de cuando es abierto un dato del gobierno, una de las primeras es la definida durante una *Open Government Working Group Meeting*[2], en la cual se listan principios que se deben cumplir. Allí se indica que los datos deben ser completos, primarios, oportunos, accesibles, procesables por máquina, no discriminatorios, no propietarios y libres de licencia. [29]

El modelo tradicional utilizado para proveer estos datos es a través de páginas web gubernamentales que permiten el acceso a los mismos, aunque no es el único enfoque y probablemente tampoco el mejor. En vez de ser el encargado de toda la cadena que va desde el almacenamiento de los datos pertenecientes a cada institución hasta su presentación al ciudadano, el gobierno puede exponer los datos mediante una infraestructura libre, simple y

[2] Diciembre del 2007 en Sebastopol, California, Estados Unidos.

confiable y dejar que terceros, sean empresas, ONGs o ciudadanos, utilizando estos datos, sean los encargados de crear las aplicaciones que provean acceso interactivo a ellos. Es un cambio de prioridades, de los sitios web del gobierno, a proveer datos reusables. Una de las maneras de moverse del viejo esquema al nuevo, es exigir que los sitios del gobierno utilicen la misma infraestructura que se expone al público. Todo esto permite un mayor nivel de innovación del que se da con un sitio único, ya que se aumenta la diversidad de los medios a través de los cuales se acceden a los datos, cada uno de los cuales puede probar soluciones y experimentar con herramientas nuevas que exploren las mejores formas de sacarle valor a los mismos. Dada una amplia oferta se espera que se destaquen las mejores, y una suerte de "mano invisible" dada por el uso de los usuarios, corrija cualquier desviación indeseable o poco útil. Dada la incertidumbre de cuál es la mejor forma de presentar los datos, es deseable que sean actores privados en un mercado dinámico quienes lo descubran, a esperar que el gobierno termine eligiendo la mejor. [30]

Ejemplos

Como iniciativas de datos abiertos a nivel mundial se puede nombrar a Data.gov[3]. Es el sitio de datos abiertos del gobierno federal de los Estados Unidos, lanzado en Mayo del 2009. En él se puede encontrar datos, herramientas y recursos a nivel federal, estatal o local para por ejemplo investigar, construir aplicaciones o diseñar visualizaciones de datos. Sus datos son provistos por cientos de organizaciones, incluyendo agencias federales, exceptuando aquellos datos que no sean privados o restringidos por razones de seguridad. El código del sitio es *open source*, y está basado a su vez en dos proyectos *open source*, CKAN y Wordpress. Data.gov está hosteado en GitHub. Dentro de ella además hay links hacia muchas aplicaciones desarrolladas por el gobierno, la comunidad, individuos o empresas que utilizan los datos que el gobierno ofrece, y que van desde información sobre el clima, aplicaciones para comprar comida más saludable en el supermercado, o para explorar estadísticas de trabajos y salarios en todo el país. Sitios similares existen para otros países, por ejemplo Alemania[4], España[5] o el Reino Unido[6]. En Argentina uno de los ejemplos es Buenos Aires Data[7]. Este es el sitio de datos abiertos de la ciudad de Buenos Aires, y de manera similar a los demás sitios de datos de gobiernos anteriormente mencionados, se provee el acceso a diversos tipos de datos y a aplicaciones desarrolladas con los mismos. Actualmente se encuentra en la versión 2.0, conteniendo algunos cambios con respecto a las versiones anteriores, en cuanto a la forma de organizar los datos y su denominación y formatos. El catálogo de datos utiliza CKAN v2.0, y dependiendo del *dataset*, estos están disponibles en diferentes formatos, como csv o pdf. El sitio aún no permite el acceso a los datos a través de una API pública.

[3] http://www.data.gov/
[4] https://www.govdata.de/
[5] www.datos.gob.es
[6] http://data.gov.uk/
[7] http://data.buenosaires.gob.ar/

Yendo más allá del acceso a datos, se puede mencionar el caso de Police.uk.[8] Este sitio permite acceder a la información sobre crímenes y el sistema policial del Reino Unido. La idea es primero seleccionar el barrio en el cual vivimos. Una vez hecho esto se muestran datos como que fuerza es responsable de la seguridad, quienes conforman el equipo local de seguridad, dónde está la estación y cómo ponerse en contacto. También se muestra una lista de prioridades para ese barrio, conformada a partir de las opiniones de la comunidad. Quizás lo más interesante sea un mapa donde se muestran todos los crímenes reportados en la zona seleccionada, exceptuando aquellos de los cuáles no se conoce donde ocurrieron o se elija que sean anónimos, pudiendo filtrarlos por fecha y tipo de crimen. Además seleccionando una zona más específica, se puede acceder a los detalles de cada investigación de esos crímenes, pudiendo ver que tipo de crimen es y en qué estado se encuentra, por ejemplo si está aún bajo investigación o si se detuvo y condenó a alguien.

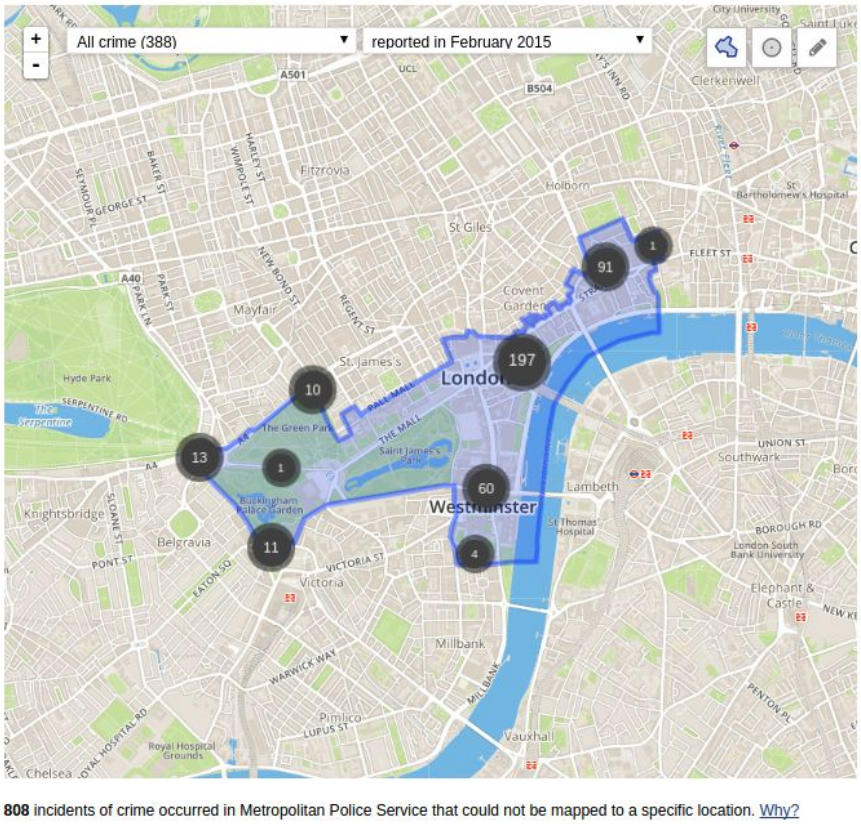

Imagen 2.1 - Mapa de los delitos en una zona de Londres, visualizado desde police.uk

Toda la información es provista por un sitio de datos abiertos,[9] que provee las opciones de acceder a través de una API pública, o descargar los datos en lote. Dentro del sitio se provee un listado de aplicaciones que utilizan estos datos, desarrolladas por el público.

[8] http://www.police.uk/

[9] https://data.police.uk/

3. Municipio de La Plata

Todo sistema de software, además de tener un marco teórico relacionado al problema a resolver, tiene injerencia sobre una parte de la realidad, el dominio del problema. Es importante conocerlo de forma de entender realmente el problema antes de diseñar la solución. Aún tratándose de un problema general, como es el manejo de la legislación, esta tesina está limitada en su alcance al caso del Municipio de La Plata. El propósito es presentar un caso concreto cercano, para evaluar la posible utilización del sistema y la capacidad de adaptación del mismo a este ámbito en particular. De esta forma se puede ganar conocimiento de aquellos problemas y beneficios potenciales que mediante un análisis global y general no habría sido posible, y se evita tener que especificar todos los casos probables de uso.

En esta parte se van a analizar las características particulares del municipio, centrándose en aquellos puntos relevantes a gobierno abierto y en todo aquello relacionado con sus normas, desde los diferentes tipos, hasta el proceso de creación de las mismas.

Datos Generales

La Plata es un partido perteneciente a la provincia de Buenos Aires, que contiene a la ciudad de La Plata, capital de la misma. Esta fue fundada en 1882 por el gobernador Dardo Rocha. A su vez, fue planificada por Pedro Benoit para cumplir con ese rol, presentando un trazado muy particular y característico fundado en concepciones higienistas y racionales por el cual es premiada en la exposición mundial de París de 1889 en las categoría "Ciudad del futuro" y "Mejor realización construida". Tiene forma de un cuadrado con calles paralelas numeradas consecutivamente, diagonales que la cruzan de norte a sur y de este a oeste, avenidas cada 6 cuadras y plazas en la intersección de esas avenidas, por nombrar algunos de sus aspectos más salientes. Entre sus logros también figuran ser la primer ciudad de América del Sur con servicio de alumbrado eléctrico y de tranvía eléctrico.

Según el último censo del 2010, el partido de La Plata tiene una población de 654,324 habitantes, lo que con sus 926 km^2 da una densidad poblacional de 585,2 habitantes por kilómetro cuadrado. [31] Esto lo sitúa como uno de los centros urbanos más grandes del país. De esa población, casi 2/3 poseen acceso a una computadora. [32] La ciudad cuenta con un gran prestigio a nivel educativo, teniendo numerosas instituciones educativas. Entre ellas, varias sedes de universidades, de las cuales la Universidad Nacional de la Plata (UNLP) es la más destacada y una de las más importantes de la Argentina, lo que atrae a diversos estudiantes de todas partes del país y países vecinos. Por esto, cuenta con una población de aproximadamente 150000 estudiantes. Asimismo, la Universidad cuenta con importantes centros de investigación en cada una de las facultades, donde se desarrollan estudios referidos a distintas actividades y problemáticas de la localidad y de la región. Otras universidades incluyen a la Universidad Tecnológica Nacional y la Universidad Católica de La Plata. Todo esto genera un ambiente propicio para el desarrollo de emprendimientos e industrias que se

benefician de esta reserva de talentos, entre las que se encuentran varias empresas tecnológicas que fueron fundadas, o poseen sedes en la ciudad.

Entre las experiencias relacionadas al gobierno abierto en el partido se destaca el presupuesto participativo, un tipo de democracia participativa donde la ciudadanía puede decidir cómo utilizar una parte del presupuesto público. Esto se realiza mediante la deliberación entre los ciudadanos, la formulación de propuestas, la elaboración de las mismas con ayuda de expertos, y finalmente una votación que decide aquellas a implementarse. La primer experiencia de presupuesto participativo se llevó a cabo en la ciudad de Porto Alegre en 1989, como intento de superar las graves situaciones de desigualdad entre sus habitantes. Se dividió la ciudad en 16 distritos en los cuales se organizan las asambleas, quienes identifican y votan las prioridades anualmente. Participa alrededor de un tercio de la población, los cuales provienen de diversos niveles económicos y políticos. Un estudio del banco mundial indica que la aplicación de este sistema llevó a mejoras en la infraestructura pública, como agua, cloacas o escuelas, un aumento en el presupuesto en educación y salud, así como también en el involucramiento de la ciudadanía. Entre el 15 y 25% del presupuesto se asigna en forma participativa. [33] A partir de este caso, se replicó en diversos lugares alrededor del mundo, como por ejemplo Francia, Italia, Alemania, España, Estados Unidos, Canadá, Perú o República Dominicana. En Argentina el primer caso se dio en Rosario en 2002. Luego siguieron San Fernando, Córdoba, San Carlos de Bariloche, Reconquista y La Plata. Esto llevó a la creación de la Red Argentina de Presupuestos Participativos.[10]

Las primeras experiencias realizadas en el partido de La Plata ocurrieron durante los años 1998 y 1999, aunque fueron limitadas en cuanto a la participación y amplitud. En 1998 se realizó una experiencia similar, con la creación de las Juntas Comunales, órganos cuyo objetivo era *"afianzar la integración entre la sociedad y su comuna"* relevando necesidades, formulando programas, y proponiendo y fiscalizando obras o distintas tareas. Entre los problemas encontrados se pueden nombrar la poca participación y por lo tanto representatividad, la falta de una asignación presupuestaria previa a los debates, y el hecho que la decisión final de qué proyectos elegir y cómo asignar el presupuesto quedaba en manos del poder ejecutivo municipal. Finalmente, a partir del año 2008 se retoma la idea del presupuesto participativo, por iniciativa del nuevo intendente de la ciudad. A diferencia de las anteriores veces, se realizó un llamado al público a participar y se asignaron partidas presupuestarias conocidas de antemano. [34]

Se realizan asambleas en 40 zonas durante 4 meses en donde se presenta la metodología, se debaten propuestas y finalmente se eligen aquellas que van a someterse a votación popular. El gobierno municipal cuenta con un presupuesto fijo y a partir del mismo, de las necesidades, los proyectos presentados y de las votaciones realizadas se realizan aquellos elegidos. [35] Actualmente (2015), el presupuesto participativo asignado es de 136 millones de pesos, lo que representa el 5,4% del presupuesto del municipio. [36] En cuanto a la aceptación, según una encuesta realizada por el municipio en 2010, hay un 28% que no conoce el presupuesto participativo, si bien hay un 57% de opinión positiva del 65% restante. Entre los puntos negativos, se identificaron inconvenientes en cuanto a población que queda sin cubrir,

[10] http://www.rapp.gov.ar/

los problemas administrativos, financieros y técnicos para cada proyecto y la incomprensión por parte de los vecinos del modo de funcionamiento, o de la distorsión de los mismos. [35]

Gobierno

La Argentina es un país federal dividido en 23 provincias más la ciudad autónoma de Buenos Aires. Cada una de ellas es responsable de definir el régimen municipal bajo el cual se han de dividir en departamentos y municipalidades, o en el caso de la provincia de Buenos Aires, en partidos. Las características generales de los gobiernos municipales de la provincia de Buenos Aires están especificadas en la Ley Orgánica de las Municipalidades (LOM)[11]. De acuerdo a ella la administración de los partidos está a cargo de una Municipalidad compuesta de un Departamento Ejecutivo, desempeñado por un ciudadano con el título de intendente, y un Departamento Deliberativo, desempeñado por ciudadanos con el título de concejal. Tanto el intendente como los concejales son elegidos por el voto por un periodo de 4 años, pudiendo ser reelectos, con el concejo renovándose cada 2 años por mitades.

El poder legislativo comunal en La Plata está a cargo del Concejo Deliberante, compuesto por 24 concejales. Un concejal es un representante político de un vecindario, cuyo objetivo es canalizar las demandas de sus representados en un cuerpo deliberante en el que se deciden las políticas locales, que se expresan mediante leyes. Además es el encargado de controlar la ejecución de la gestión del poder ejecutivo municipal. Por esto puede decirse que cumple a la vez los roles de representante del pueblo, legislador y controlador. Las facultades y atribuciones de un concejal, de acuerdo a los artículos 24 al 67 de la LOM, pueden dividirse en cuatro categorías: legislativas (creación y sanción de normas para la localidad), reglamentarias (como la radicación, habilitación y funcionamiento de los establecimientos comerciales e industriales, el trazado, apertura, rectificación, construcción y conservación de calles, caminos, puentes, túneles, plazas y paseos públicos, etc), presupuestarias y de control. Estas facultades son de carácter residual y por lo tanto no deben contradecir las facultades, resoluciones o legislación nacional y provincial. [1]

Legislación municipal

De acuerdo a la LOM los proyectos y solicitudes que pueden presentar los concejales pueden ser de las siguientes formas:

> *ARTICULO 77°: (Texto según Ley 13101) Las disposiciones que adopte el Concejo se denominarán:*
>
> *Ordenanza, si crea, reforma, suspende o deroga una regla general, cuyo cumplimiento compete a la Intendencia Municipal. Las Ordenanzas serán consideradas ley en sentido formal y material.*

[11] Decreto-ley 6769/58

Decreto, si tiene por objeto el rechazo de solicitudes particulares, la adopción de medidas relativas a la composición u organización interna del Concejo y en general, toda disposición de carácter imperativo, que no requiera promulgación del Departamento Ejecutivo.

Resolución, si tiene por objeto expresar una opinión del Concejo sobre cualquier asunto de carácter público o privado, o manifestar su voluntad de practicar algún acto en tiempo determinado.

Comunicación, si tiene por objeto contestar, recomendar, pedir o exponer algo.

De la LOM y de fallos judiciales que lo avalan, se puede afirmar que las ordenanzas son calificables como leyes en sentido formal y material, las cuales rigen en el territorio local del municipio para todos sus habitantes y sobre temas en el cual la municipalidad es competente. [1] Para el caso del municipio de La Plata el promedio aproximado es de 200 ordenanzas por año, aunque este último tiempo esta cantidad ha bajado a alrededor de 50. [2] También el poder ejecutivo puede enviar proyectos de ordenanzas para ser tratados en el concejo.

Expte. 54372

LA PLATA, 27 de junio de 2012

El Concejo Deliberante, en su Sesión Ordinaria N° 11 celebrada en el día de la fecha, ha sancionado la siguiente:

ORDENANZA 10908

ARTICULO 1°. Créase en el ámbito municipal el programa "un nacimiento, un árbol".

ARTICULO 2°. Por el mismo, se entregará un árbol a la familia con domicilio en el Partido de La Plata, como símbolo de una nueva vida, de desarrollo y esperanza para la Ciudad.

ARTICULO 3°. La plantación del árbol se efectuará, conservando la especie arbórea preponderante de la zona.

ARTICULO 4°. Facúltase al Departamento Ejecutivo, mediante la Dirección de Espacios verdes, a disponer que el árbol a ser entregado provenga del Municipio.

ARTICULO 5°. Se sugiere realizar plantaciones conjuntas los siguientes días: el Día del Medio Ambiente (5 de julio), el Día del Arbol (29 de agosto) y el Día de la Primavera (21 de septiembre).

ARTICULO 6°. De forma.

Un ejemplo de ordenanza

Por otro lado, para el caso de los decretos se trata más de un acto administrativo, normalmente atribuible al poder ejecutivo. En el caso de las municipalidades esta división de poderes no es tan clara, de allí que se hable de departamento y no poderes y el concejo deliberante pueda pronunciar decretos, denominación que suelen tener los actos del departamento ejecutivo, quien también en este caso puede emitirlos. [1]

Expte.59462

LA PLATA, 13 de mayo de 2015

El Concejo Deliberante, en su Sesión Ordinaria N° 6, celebrada en el día de la fecha, ha sancionado el siguiente:

DECRETO N° 25

ARTICULO 1°. Apruébase la Rendición de Cuentas del Ente Municipal La Plata, correspondientes al Ejercicio 2014, elevada por el Departamento Ejecutivo.

ARTICULO 2°. A efectos de cumplir con lo dispuesto en el inciso 5 del Artículo 165º de la Ley Orgánica de las municipalidades, elévense las presentes actuaciones al Departamento Ejecutivo para que sean remitidas al Honorable Tribunal de Cuentas de la Provincia de Buenos Aires, junto a la documentación correspondiente.

ARTICULO 3 °. De forma.

Un ejemplo de decreto

Las resoluciones suelen servir para fijar una posición respecto a un tema en particular, y son normalmente utilizados para rendir homenaje o declarar a alguien persona no grata. [1] La cantidad anual producida, en el partido de La Plata, es en promedio el doble que en el caso de las ordenanzas. [2]

Expte. 59764

LA PLATA, 20 de mayo de 2015

El Concejo Deliberante, en su Sesión Ordinaria N° 7, celebrada en el día de la fecha, ha sancionado la siguiente:

RESOLUCION N° 106

ARTÍCULO 1º: Solicitar al Departamento Ejecutivo estudie la posibilidad de gestionar ante las autoridades de Aguas Bonaerenses S.A. (ABSA) a fin de requerirle la ampliación de la red cloacal en las calles 605 a 609 de 13 a 16.

ARTÍCULO 2º: Solicitar al Departamento Ejecutivo estudie la posibilidad de gestionar ante las autoridades de Aguas Bonaerenses S.A. (ABSA) a fin de requerirle la normalización del servicio de la red de colectoras cloacales ubicada en:

-Diagonal 618 y 82 de la localidad de Villa Elvira.
-Calle 4 y 77 Nº 2245 de la localidad de Villa Elvira.

ARTÍCULO 3°: Solicitar al Departamento Ejecutivo estudie la posibilidad de gestionar ante las autoridades de Aguas Argentinas S.A. (ABSA) a fin de requerirle solucione el desborde cloacal existe en calle 34 y 139.

ARTÍCULO 4º: Solicitar al Departamento Ejecutivo estudie la posibilidad de gestionar ante las autoridades de Aguas Argentinas S.A. (ABSA) a fin de requerirle la reparación de la pérdida de agua en:

Calle 2 entre 80 bis y 81 de la localidad de Villa Elvira.
Calle 1 bis entre 77 y 78.

ARTICULO 5°: De forma.

Un ejemplo de resolución

Por último, una comunicación es el medio mediante el cual el concejo se dirige a otros organismos o particulares, para por ejemplo pedir información o instar a ciertas medidas. Ya que cabe la posibilidad de que se exprese sobre temas fuera de su competencia, es importante evaluar el uso que se hace de esta forma de manifestación. [1] Son muy poco frecuentes en comparación a los demás tipos, del orden de 1 cada 3 años aproximadamente. [2]

Proceso legislativo

El primer paso en la creación de alguno de estos tipos de normas es la llamada etapa pre-legislativa, que abarca la preparación del proyecto a presentar posteriormente. En cuanto al origen de la idea, usualmente los concejales tienen asesores que estudian un tema, o esta puede surgir de instituciones o vecinos, aunque estos casos son poco frecuentes y suelen ocurrir sólo con temas de trascendencia. [101] Pueden identificarse en la etapa de preparación del proyecto dos planos: de fondo o hecho y de forma o redacción. El primero se refiere a identificar el objeto sobre el cual se va a legislar, definiendo su objetivo, y a partir de este, el tipo de proyecto que se va a elaborar. Por ejemplo, si el problema es que hay una mala aplicación de una ordenanza por el departamento ejecutivo, se hará una comunicación indicando esto, si en cambio es un problema de la ordenanza en sí, se deberá realizar una modificación sobre la misma. [37] También se analiza la razonabilidad del mismo, de forma de asegurarse que se estaría resolviendo una problemática real y de una manera factible. El otro aspecto es la redacción del proyecto. La LOM no se explaya mucho en este tema, sólo mencionando que las ordenanzas y decretos deberán ser concisos y preceptivos, sin mencionar a las comunicaciones y resoluciones. [1] En este punto es recomendable buscar antecedentes locales y de otros municipios sobre ordenanzas similares, como así también ayuda en el ámbito público y privado, en donde hubiera conocimiento sobre el área a legislar. Esto hace necesario que existan facilidades y métodos confiables a la hora de buscar legislación, lo cual no es el caso generalmente, por ejemplo dadas las deficiencias en los digestos. [101] Un aspecto contemplado en la ordenanza general 267 en su artículo 124, para la sanción de actos de carácter general, es el sometimiento a información pública: “La iniciativa podrá ser sometida a información pública por disposición y plazo que señale el Departamento Ejecutivo. Igualmente, se podrá requerir informes a sociedades o personas ajenas a la administración municipal.”. Esto habilita a que un concejal remita a estas entidades el proyecto para ser analizado, de forma de tener un proyecto con mejores argumentos a la hora de presentarlo a debate. [37]

El siguiente paso involucra la presentación del proyecto dentro de una sesión del concejo, en donde se decide a qué comisión debe enviarse. Este paso por una comisión es obligatorio salvo en contadas excepciones donde no se necesita el dictamen o despacho de la comisión. Las comisiones son las encargadas de discutir y lograr que los proyectos sean elevados a su tratamiento en pleno con un dictamen previo, que indica la existencia o no de consenso entre los diferentes actores dentro de esa comisión. [1] Una vez hecho esto, en una sesión determinada del concejo, se va a incluir ese proyecto dentro del orden día, y cada concejal va a votar por afirmativo o negativo la sanción de la misma. Si es sancionada ésta pasa al departamento ejecutivo, quien tiene la potestad de vetar, vetar parcialmente o promulgar la norma. Finalmente es también el ejecutivo el encargado de dar publicidad y difusión a la misma a través del boletín oficial del municipio, de forma de que se asegure su eficacia. [2] [101]

Problemas en la creación, manejo y publicidad de las normas

Más allá de las particularidades del municipio de La Plata, hay ciertas problemáticas recurrentes en el resto de la provincia de Buenos Aires, con respecto a la legislación, que vale la pena analizar y que también son aplicables a este. Se puede decir que el problema general más relevante para nuestro caso es la falta de conocimiento de la normas por parte del gran conjunto de la población. Este problema surge a partir de diferentes causas.

La primera de ellas se refiere a los problemas relativos a la normatividad emanada de los concejos deliberantes. La falta de ordenamiento del texto de las ordenanzas se hace evidente al momento de trabajar en modificaciones a una de ellas o al intentar conocer la ordenanza vigente. Debido a que las ordenanzas (y las leyes en general) sufren modificaciones a lo largo del tiempo a causa de otras posteriores, se termina teniendo una versión original y cambios diversos en diferentes ordenanzas. Lo que normalmente no se realiza es un llamado texto ordenado, en el cual se aplican esos cambios sobre la ordenanza inicial para producir un texto que representa la ordenanza vigente. [37] [2] Esto se agrava si pensamos en los errores en la redacción de las normas desde el punto de vista técnico, en cuanto a ambigüedades y vaguedades en el texto. Otro problema es la falta de sistematización de las normas, en cuanto a compilaciones y su acceso a través de sistemas informáticos, que dificultan conocer sobre qué temas se legisló o no. Esto genera que a la hora de elaborar un nuevo proyecto no se mida adecuadamente las vinculaciones con normas anteriores, de forma de poder encontrar contradicciones entre ellas, ya que ningún concejal puede llegar a recordar absolutamente todo lo sancionado. Es por eso que se vuelve necesario iniciar procesos codificatorios que permitan ordenar y depurar la legislación, sumado a una sistematización, sobretodo en aquellos municipios donde la codificación no es factible, de forma de poder acceder más fácilmente a la información. [1]

El caso de La Plata no es excepción, aunque igualmente el municipio cuenta con 6 códigos: Código de Ordenamiento Urbano y Territorial, Código de Uso del Espacio Público, Código de Edificación, Código Faltas Municipal, Código Tributario y Código de Nocturnidad, y se han producido textos ordenados de los mismos. En cuanto sistematización, además de un

digesto tradicional, se cuenta con un digesto digital accesible desde la web, pero que no posee la totalidad de las normas y su historia, [2] y presenta errores y poco mantenimiento. Esto dificulta la tarea del concejal a la hora de buscar información. [101]

Es importante mencionar una problemática actual que no es particular de los municipios, sino que es general a toda legislación: la llamada "inflación legislativa". Esta se define como el aumento en la sanción de normas y en su complejidad en todos los niveles del estado, y en consecuencia en su desvalorización, y si bien puede justificarse por el incremento en la complejidad de la sociedad, no siempre es proporcional a este ya que responde a la creación innecesaria de una norma. En el ámbito municipal se presenta cuando se aprueban ordenanzas por motivos diferentes a responder a una necesidad concreta, como por presiones de ciertos sectores, o que entran en contradicción con normas anteriores. [1]

La otra problemática es la de la publicidad de las leyes municipales, debido a la carencia o deficiencia en la generación de boletines o publicaciones con lo sucedido en las sesiones, con las ordenanzas sancionadas, o los actos administrativos dictados por el poder ejecutivo. Estos son indicados como necesarios por el artículo 108 de la LOM, que fuera modificado en el año 2012 por la ley 14.491, el cual además exige a los mismos ciertos requisitos de periodicidad y contenido. Ellos deben ser editados al menos una vez por mes y deben contener todas las disposiciones del concejo con la totalidad de su contenido. Dado que la publicación en un boletín tradicional no asegura su conocimiento por parte de la sociedad, en tanto que son pocas personas lo que los leen, se torna necesario realizarlo también por otros medios, particularmente mediante las TIC. Es por eso que además se establece la obligatoriedad de su publicidad en internet, así como también en lugares públicos. [38] Más allá de lo dispuesto por este artículo, hay casos de incumplimiento del mismo. [39] Si bien la no publicidad de una norma no implica su inconstitucionalidad, sí ocasiona su inefectividad, según la Suprema Corte de Justicia y diversos fallos. Esto termina por depender del departamento ejecutivo que es el que se encarga de aprobar todas las leyes, y por lo tanto es el responsable de publicarlas. [1]

En el caso de municipio de La Plata, se cuenta con un boletín oficial, el cual es también accesible a través de internet. Es común que no contenga la totalidad de las normas y que su publicación varíe en cuanto a su periodicidad. Esto trae inconvenientes a la hora de conocer qué es lo que está vigente, ya que por ejemplo una ordenanza puede estar promulgada pero no publicada. [2]

La falta de publicidad, junto a los defectos técnicos en la elaboración de la legislación y la inflación legislativa, atentan contra el principio jurídico de que nadie puede alegar desconocimiento de la ley como justificativo para una conducta que la viole, el cual es una de las bases del sistema legal. A la necesidad legal por parte de un ciudadano de conocer las leyes, también se suman las necesidades políticas relacionadas con la difusión de los actos del gobierno y la posibilidad de ejercer contralores adecuados. Dadas las condiciones actuales, en las cuales puede afirmarse que ni siquiera los abogados especializados en un tema pueden conocer la totalidad de normas existentes, este principio se ha convertido en una ficción jurídica. [1]

4. Git y GitHub

El punto central de esta tesina es el manejo digital de los cambios sobre la legislación a lo largo del tiempo. Si bien muchas soluciones se han desarrollado para este problema, el enfoque elegido hace uso de un sistema de versionado de código. De la misma manera, aún habiendo muchas alternativas, se elige a Git para esta función entre las diferentes opciones.

En esta parte se va a justificar esta elección, proveyendo un pantallazo general de lo que es el versionado de código y describiendo aquellas características generales de Git que lo hacen destacarse sobre el resto. También se van a tratar otras cuestiones ya no particulares de Git en sí mismo, como por ejemplo GitHub, una plataforma colaborativa que permite el *hosting* de repositorios.

Control de versiones

El cambio es un aspecto fundamental de los datos. Para ciertos tipos de datos, como código fuente o documentos, es importante llevar un registro de los mismos. Un sistema de control de versiones (*version control system* o VCS)[12] es una herramienta que maneja y lleva un seguimiento de los diferentes cambios en los datos, cumpliendo al mismo tiempo varios roles dentro del ámbito del desarrollo de software. Por un lado, el de herramienta de colaboración, coordinando las modificaciones realizadas por varias personas sobre los mismos archivos, y permitiendo compartir los datos entre diferentes nodos para que cada uno tenga la última versión de los mismos. Por otro lado, permiten ver la evolución de los datos a través del tiempo con todos sus estados intermedios, y recuperar la información de un estado en particular. Por ejemplo en caso de que haya algún error, puede recuperarse la última versión válida y continuar trabajando a partir de ella. Algo tan importante como guardar los cambios es guardar las razones tras ellos, por eso un VCS asocia para cada uno la información sobre quién lo realizó, cuando y porqué, en la forma de un usuario, una fecha y un mensaje explicando el cambio, respectivamente. [40]

Si bien cada VCS tiene características particulares, hay conceptos que se repiten en cada uno. Todos los sistemas guardan esta información en un lugar llamado repositorio, el cual contiene toda la historia de los datos y la información asociada a cada versión. Cada uno de estos estados de un archivo o proyecto es llamado *commit.* [13] La primer acción a realizar sobre un VCS es la de obtener los datos desde el repositorio, para esto se realiza una acción llamada *checkout* por la que obtenemos una copia local, llamada *working copy*. El *working directory*[14] es donde se realizan los cambios al conjunto de datos para luego persistirlos en el repositorio mediante un *commit*. [41] En la mayoría de los sistemas cada *commit* está acompañado de un mensaje explicando los cambios realizados, junto al nombre de usuario y la fecha. Además se le asigna un número único de *commit*, que dependiendo del VCS puede ser un número

[12] También llamado de otras formas, como *source code manager* (SCM) o *revision control system* (RCS)
[13] También llamado de otras formas, como *revision*, *version* o *changeset*.
[14] También llamado de otras formas, como *working tree*, *working copy*, *sandbox* o *checkout*.

secuencial, al azar o un hash de los cambios. [40] En base a donde se encuentre el repositorio pueden clasificarse a los VCSs en: locales, centralizados o distribuidos.

En los VCS locales la persona realizando los cambios y el repositorio están en la misma máquina, éstos corresponden a los primeros VCS. Ejemplos de VCS locales son SCCS o RCS.

Los VCS centralizados (*centralized* VCS, o CVCS) siguen una arquitectura cliente-servidor, en donde se cuenta con un solo repositorio central en un servidor sobre el cual los clientes leen y escriben, y al cual acceden a través de una red desde cualquier lado de esta. Un cliente puede realizar un *checkout* de una versión de un archivo particular o el repositorio completo, obteniendo una *working copy,* la cual carece de la historia. Luego de editar los archivos, se realiza un *commit* sobre el repositorio, creando una nueva versión en el mismo. [42] Este modelo ofrece las ventajas por sobre los repositorios locales de permitir tener un mayor control sobre las operaciones que se realizan sobre el repositorio, una mayor visibilidad sobre lo que están trabajando las demás personas y una mayor facilidad para la administración, en comparación a administrar varias bases de datos locales. [43] Ejemplos de CVCSs son CVS, SVN y Perforce.

Los VCS distribuidos (*decentralized* o *distributed* VCS*,* o DVCS) no necesitan de un repositorio central, debido a que cada persona tiene una copia de todos los datos en un repositorio local. Ya que se cuenta con toda la historia se suele utilizar algún método de compresión para reducir el tamaño del repositorio. El tener el repositorio local permite trabajar de forma *offline*, y además, como cada operación se realiza localmente, son más rápidas que aquellas realizadas por sobre una red. Para colaborar con diferentes personas, aunque no es estrictamente necesario, se suelen utilizar repositorios remotos, usualmente configurando un repositorio que actúe de central, el cual se clona para obtener una copia local y hacia el cual se envían y se obtienen los cambios. [42] Otra de las ventajas por sobre los CVCSs o VCSs locales es que en estos hay un solo punto de falla, ya que si el servidor falla o se corrompe y no se tomaron medidas provisorias se puede llegar a perder todos los datos, mientras que en el caso de los DVCSs cada persona tiene una copia completa del repositorio, pudiendo restaurarse a partir de cualquiera. [43] Ejemplos de de DVCSs son Bazaar, Git y Mercurial.

Hay diferentes maneras en las que un repositorio puede guardar las diferentes versiones de un archivo, pudiéndose diferenciar en aquellas basadas en *snapshots*, en *deltas* (también llamados *changesets* o *diffs)*, o *weaves*.

En el caso basado en *snapshots* cada versión de los datos es guardada completamente, independientemente de las versiones anteriores, lo que hace que la recuperación de los mismos sea asintóticamente constante. Igualmente para ahorrar espacio usualmente se utilizan algoritmos para comprimir los datos. Git y Bazaar usan este modelo.

Para el caso de que se usen *deltas* lo que se guardan son las diferencias entre dos versiones de los datos, basándose en que en general dos archivos no tienen muchas diferencias entre dos versiones consecutivas. Este método implica que para obtener cierta versión deba utilizarse una versión base y aplicar los diferentes *deltas* hasta llegar a la misma. [40] Una manera de implementarlo es con *forward deltas*, en los que se guardan una versión de los datos y los *deltas* sobre las versiones posteriores, que deben aplicarse para obtener la versión actual. También pueden usarse *reverse deltas*, en los que se guarda la versión más nueva y los *deltas* sobre las versiones anteriores, aumentando generalmente el rendimiento ya

que la versiones mas nuevas son accedidas más frecuentemente que las viejas. También está la opción de los *skip deltas*, que calculan la diferencia con otras versiones diferentes a la anterior, para producir *deltas* menores o necesitar menos cambios para reconstruir un archivo. Mercurial, CVS y Subversion (SVN) usan *deltas* para guardar la historia de los archivos.

Con *weaves* todas las versiones de un archivo se guardan en un solo archivo, con cada bloque conteniendo metadata que indica a qué versión pertenece. Esto permite obtener cualquier versión a partir de una lectura secuencial del mismo. Versiones viejas de Bazaar usaban este modelo. [41]

En el último tiempo los dos sistemas más utilizados son SVN y Git, [44] [45] [46] ambos representativos de dos modelos de sistemas de versionado diferentes, el centralizado y el distribuido.

¿Qué es Git?

Git es un DVCS inicialmente diseñado y desarrollado en 2005 por Linus Torvalds para el desarrollo del kernel de Linux. Git es software libre, distribuido bajo los términos de la GNU General Public License version 2 (GPLv2). Actualmente es el VCS más utilizado en el desarrollo de software. [47] La idea surgió a partir de que el sistema de versionado propietario utilizado desde el 2002 por los desarrolladores del kernel de Linux, Bitkeeper, dejó de ser ofrecido gratuitamente a la comunidad. A partir de la necesidad de reemplazarlo y la ausencia de algún sistema que cumpliese con los requisitos que se esperaban, la comunidad de desarrollo del Linux, y en particular Linus Torvalds, decidió desarrollar su propio sistema. Estos requisitos eran: evitar la forma de trabajo de los CVCSs, tomándolo como ejemplo de lo que no había que hacer, soportar un flujo de trabajo distribuido como BitKeeper, [48] tener garantías contra corrupción de los datos, accidentales o maliciosas [49] y buen rendimiento, por ejemplo en la aplicación de parches, [50]. Se plantearon además los siguientes objetivos que debía cumplir, basándose en algunas lecciones aprendidas con el uso de BitKeeper: [43] velocidad, diseño simple, fuerte soporte para desarrollos no lineales, totalmente distribuido y capaz de manejar eficientemente grandes proyectos como el kernel de Linux.

El desarrollo empezó el 3 de abril del 2005, saliendo la versión 1.0 el 21 de Diciembre de ese mismo año. El diseño estuvo influenciado por BitKeeper y Monotone, siendo su objetivo el ser un sistema de versionado de bajo nivel, sobre el cual pudieran desarrollarse otras aplicaciones para actuar como interfaces. A pesar de esto Git terminó siendo un sistema de versionado completo en sí mismo y con un diseño único, aunque reteniendo la facilidad de uso y siguiendo cumpliendo los objetivos iniciales. [51]

Estructura del repositorio

En su forma más básica, Git es un *store* de datos del tipo clave-valor. Esto significa que puede almacenarse cualquier tipo de contenido y Git permitirá acceder al mismo mediante una clave en cualquier momento posterior. [43] Toda la información y metadatos de un repositorio

están situados en un directorio `.git/` oculto en la raíz del directorio designado como repositorio.

Git utiliza el modelo de *snapshots* para representar los cambios en los datos. Para esto internamente maneja 4 tipos de objetos: *blob* (*binary large object*), *tree*, *commit* y *tag* los cuales guarda en el directorio `.git/objects/`. Todos estos objetos son inmutables y tienen como nombre un código SHA1 calculado a partir de sus contenidos. [42] Cada versión de un archivo es representado por un *blob*, este contiene los datos del archivo pero ningún otro tipo de metadatos sobre el mismo. Un *tree* representa un nivel de un directorio, el cual puede apuntar a cualquier número de *blobs* u otros *trees*, por lo cual puede formar una estructura recursiva con la cual representar cualquier sistema de archivos. Guarda además metadata sobre los archivos, como por ejemplo sus nombres. Un *commit* guarda la metadata de cada cambio producido en el repositorio, por ejemplo el nombre de quien hizo el *commit* y la fecha, junto a un puntero a un *tree* que representa el estado del repositorio cuando se realizó ese *commit*. Cada *commit*, excepto el primero, tiene un puntero a su *commit* padre, o a sus *commits* padres en caso de haberse producido mediante un *merge*, es decir la unión de dos *commits* diferentes. Un *tag* es un alias que se le asigna a un objeto, usualmente un *commit*, de forma de poder referenciarlo por ese alias y no por su nombre. [52]

Integridad de datos

El modelo de datos utilizado por Git asegura la integridad criptográfica de los mismos. Cada archivo y commit es procesado con la función de *hash* SHA-1. SHA-1, por *secure hash algorithm*, es una función criptográfica de *hash* diseñada por la Agencia Nacional de Seguridad (NSA) de los Estados Unidos y publicada en 1995. Produce un valor de 160 bits, generalmente representado como un número hexadecimal de 40 caracteres (0-9 y a-f). [53] Esta es la función más utilizada de la familia de algoritmos SHA, que van del SHA-0 a SHA-3, a pesar que en los últimos años surgieron movimientos para deprecar certificados SSL basados en SHA-1 por parte de organizaciones como Microsoft, Google o Mozilla, debido a ciertas vulnerabilidades encontradas. [54]

Git genera una clave mediante esta función a partir de los contenidos de los objetos, que utiliza de manera extensiva en la implementación de la mayoría de sus funcionalidades, por ejemplo para acceder a los datos. Ya que cada *commit* tiene una referencia a un *tree*, el cual tiene a su vez referencias a los *trees* y *blobs* de ese *snapshot*, y una referencia a sus *commits* padres, quienes presentan la misma estructura, es imposible cambiar algún archivo, fecha, mensaje de commit o algún otro dato sin cambiar la clave de todo después de eso. Esto asegura que los datos que se obtienen son los mismos que se pusieron y que no se cambió nada de la historia del proyecto. [55]

Branching y merging

Un *branch* en un VCS representa una línea de trabajo paralela a la línea de trabajo principal, referida en varios VCS como *trunk*, con la cual habrá de unirse eventualmente en

caso de que se desee incorporar ese desarrollo. En Git un *branch* es sólo un objeto liviano que apunta a un objeto *commit* particular. Por este motivo son fácilmente manipulables y la creación y eliminación de nuevos branches es una operación muy poco costosa, a diferencia de otros sistemas donde estas operaciones implican por ejemplo la duplicación de directorios. [43] Esto permite que Git tenga un gran soporte para desarrollos no lineales e invita a utilizar *branches* en una amplia gama de situaciones. Permite cambios de contexto más ágiles, ya que se puede estar trabajando en un *branch*, cambiar a otro a arreglar un *bug*, y luego volver al inicial, aplicar la solución al *bug* y seguir trabajando. Se pueden tener *branches* con código para diferentes tipos de entornos, o crear nuevos *branches* para cada nueva funcionalidad a desarrollar, borrandolos cuando ese trabajo sea integrado con el desarrollo principal o al ver que debe descartarse porque por ejemplo la nueva funcionalidad ya no es necesaria. [55]

Estas integraciones de diferentes líneas de desarrollo se realizan mediante un *merge*. *Merging* es el proceso de incorporar cambios de diferentes fuentes sobre un mismo archivo o proyecto. Dependiendo del CVS se siguen diferentes estrategias para resolver los conflictos. En el caso de Git se utiliza un algoritmo recursivo que realiza un *three-way merge*, que es básicamente aplicar los cambios introducidos por ambas versiones sobre su ancestro común. [41] Esto genera un nuevo objeto *commit* que apunta a sus dos *commits* padres. También permite otras alternativas como por ejemplo realizar un *rebase*, que aplica los *commits* sucesivamente en el *branch* sobre el cual se quiere realizar el *merge*, dando como resultado una historia lineal. [42]

Workflow Distribuido

Como se mencionó anteriormente, Git es un DVCS. Esto quiere decir que a diferencia de un CVCS, donde se cuenta con un solo repositorio central sobre el cual se realizan las operaciones, cada persona que obtenga una copia del repositorio va a tener toda la historia del proyecto de manera local. Esto implica que cualquiera va a tener un backup completo del proyecto, agregando mayor tolerancia a fallos, ya que en caso de corrupción de datos en el servidor central, estos pueden reemplazarse con los datos que se tienen localmente. Además deja de ser necesaria la comunicación a través de la red con un repositorio para realizar cada operación, ya que se realizan localmente y la sincronización con otro repositorio puede posponerse. Esto hace que la gran mayoría de las operaciones sean mucho más rápidas al evitarse la latencia de red. Sumando esto a su sistema de branching, se abren una gran variedad de flujos de trabajo: [55]

- Centralizado: Este es el caso más parecido a un CVCS, donde todos se sincronizan con un mismo servidor que actúa de central. Como Git no permite hacer un *push* si otra persona ya realizó un *push* desde la última vez que se trajeron cambios, este esquema funciona sin problemas.

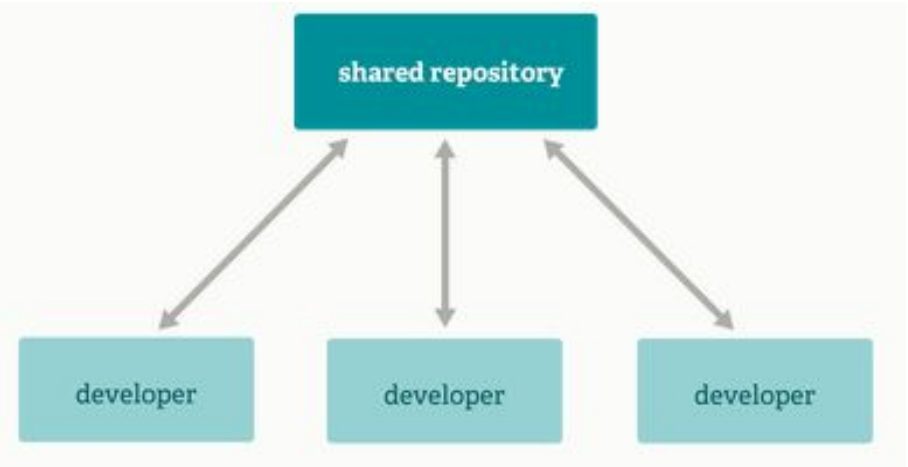

Imagen 4.1 - Flujo de trabajo centralizado [43]

- Gestor de integración: Otro patrón común involucra tener a una persona encargada de realizar los commits a un repositorio "bendecido". Los demás desarrolladores clonan desde ese repositorio, envían su trabajo a sus propios repositorios remotos e indican al gestor que traiga esos cambios y los aplique. Este modelo es muy común dentro de proyectos *open source* o repositorios alojados en sitios como GitHub, donde suele llamarse modelo *fork & pull*. Cada persona que quiera desarrollar una funcionalidad en un proyecto debe primero realizar un *fork* del mismo. Esto genera una copia personal del proyecto oficial. Una vez hecho esto, se lo clona y se empieza a trabajar sobre el mismo, idealmente en un *branch* aparte, el cual se envía al *fork*. Cuando se decide que la funcionalidad está lo suficientemente madura, se crea un *pull request*[15] sobre el repositorio oficial, lo que inicia una conversación con revisión de código sobre esos cambios a aplicar, donde se discute, corrige y llegado el caso se sigue desarrollando hasta que el pedido es aceptado o rechazado. Si es aceptado, el desarrollo contenido en ese *branch* es integrado dentro del proyecto oficial. Permite a cualquier persona colaborar en cualquier proyecto y a aquellos responsables del mismo ordenar esas contribuciones y elegir cuáles se aplican sobre el proyecto oficial, sin necesidad de agregar colaboradores para otorgarles permisos de escritura. [43]

[15] Como se lo llama en GitHub, otros sistemas pueden utilizar otras denominaciones pero el concepto es el mismo

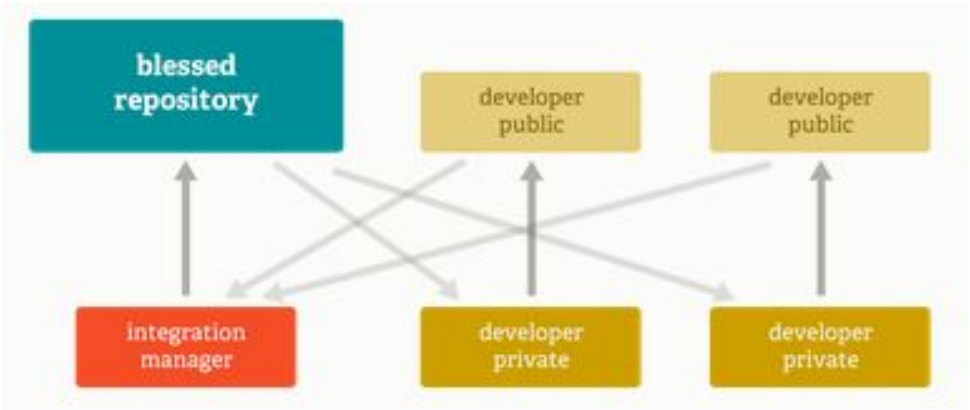

Imagen 4.2 - Flujo de trabajo con gestor de integración [43]

- Dictador y tenientes: Para proyectos más masivos se puede usar un esquema como el del kernel de Linux. En este modelo, algunas personas llamadas tenientes, están a cargo de un subsistema específico del proyecto y juntan todos los cambios relacionados a ese subsistema. Otro integrador, el dictador, puede traer cambios sólo desde sus tenientes y luego enviarlos hacia el repositorio "bendecido" del cual todos clonan nuevamente.

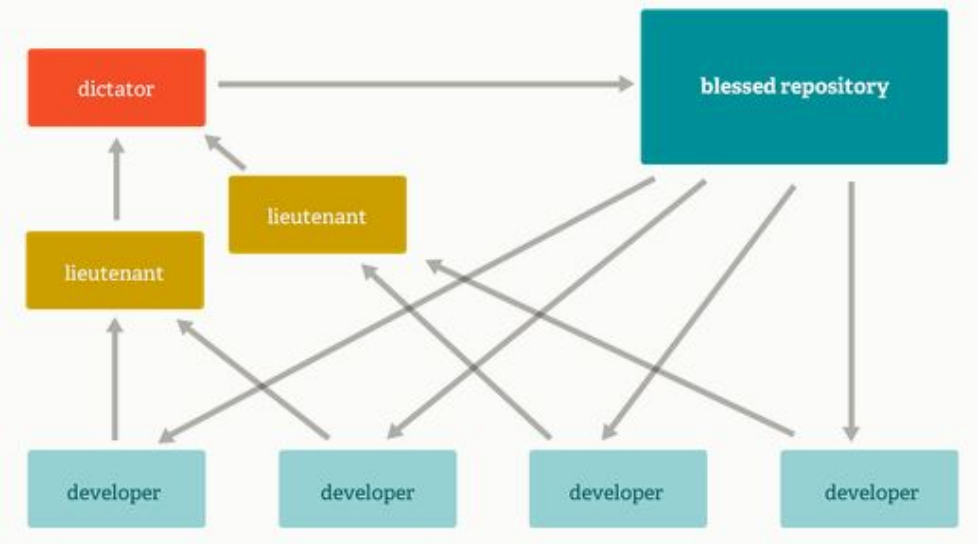

Imagen 4.3 - Flujo de trabajo con dictador y tenientes [43]

Protocolos de comunicación

Git puede usar cuatro protocolos para transferir estos datos desde y hacia los repositorios: local, HTTP, SSH y Git.

El protocolo local es el más básico. En éste el repositorio remoto está en otro directorio o disco y se usa cuando todas las personas del equipo tiene acceso a un sistema de archivos compartido. Tiene la ventaja de que se usan los permisos de los archivos y el acceso a la red

ya existentes. El problema es que es más difícil de configurar comparado con otros protocolos, en el caso de que se quiera acceder de diferentes lugares, y es más lento por ejemplo comparado al acceso por SSH sobre el mismo servidor. Sólo es rápido si se tiene un acceso rápido a los datos.

Git también puede comunicarse sobre HTTP de dos formas. La primera simple y en general de sólo lectura, denominada *dumb HTTP*, permite el acceso a los archivos del repositorio como archivos normales en un servidor web. Luego se introdujo el protocolo *smart HTTP*, que permite negociar los datos inteligentemente de una manera similar a SSH. Este se usa sobre puertos HTTP estándar, y puede usar varios mecanismos de autenticación, como usuario/contraseña, que facilita su uso por parte de usuarios sin muchos conocimientos, en comparación a SSH. Es probablemente el más popular ya que puede usarse para proveer acceso anónimo como el protocolo Git, o autenticado y encriptado como SSH, todo con una misma URL. También pueden configurarse los repositorios para accederse en modo de solo lectura sobre HTTPS, de manera de encriptar el contenido, o hacer que los clientes usen certificados SSL. La contra es que puede ser más complicado de configurar sobre algunos servidores en comparación a SSH. Más allá de eso hay pocas ventajas de los demás protocolos por sobre *smart HTTP.*

La siguiente forma es a través de Secure Shell (SSH). SSH es un protocolo de red para iniciar sesiones de shell sobre máquinas remotas de manera encriptada. Esto permite correr comandos sobre una máquina estableciendo un canal seguro sobre una red insegura, conectando un cliente SSH con un servidor SSH. [56] Este protocolo de transporte para Git es común en el caso de que se haga un *hosting* propio, ya que el acceso SSH a los servidores ya está configurado en la mayoría de los lugares, o es fácil de realizar. La mayor ventaja es que es un protocolo autenticado fácil de configurar, de uso muy común y eficiente, ya que compacta la información lo más posible antes de realizar la transferencia. La contra es que no se puede permitir el acceso anónimo al repositorio incluso para sólo lectura, ya que se debe tener acceso a la máquina sobre SSH, lo que no lo hace adecuado para proyectos *open source*.

Finalmente está el protocolo Git. Este es un *daemon*, un programa que corre como un proceso en *background*, que escucha en un puerto dedicado y que provee un servicio similar a SSH, pero sin autenticación. Al no haber seguridad, el repositorio está disponible para cualquier persona, por lo que generalmente éste no permite realizar un *push*. Es generalmente el protocolo de red más rápido ya que usa los mismos mecanismos de transferencia de datos que SSH, pero sin el costo de autenticación y encriptación. Por esto es la opción más recomendable en caso de tener un proyecto público con alto nivel de tráfico, que permita el acceso de sólo lectura sin ningún tipo de autenticación. [43]

Staging Area

Git implementa un concepto llamado *staging area* o *index*, un área intermedia donde se pueden formatear y revisar los *commits* antes de realizarlos. Es posible de esta forma enviar sólo algunos archivos a esta área, y posteriormente realizar su *commit*, sin el resto de los archivos modificados en el directorio de trabajo. Además se pueden agregar sólo porciones de

un archivo, por lo que los cambios realizados sobre el mismo pueden ser divididos entre varios *commits*.

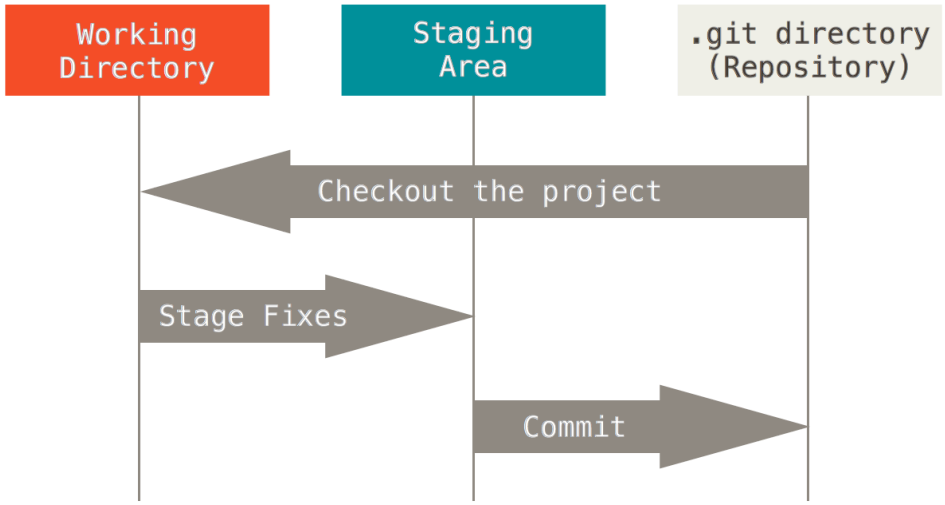

Imagen 4.4 - Flujo de trabajo en Git [43]

Como se ve en la imagen 4.4 el flujo de trabajo básico en Git consiste en obtener una versión del proyecto desde el repositorio y realizar cambios en algunos archivos para luego agregarlos al *staging area*. Esto se realiza mediante el comando `git add`. Finalmente se crea el *commit* y se lo persiste en el repositorio con lo que se haya agregado.

Más allá de su objetivo de facilitar el trabajo de los desarrolladores, sobretodo cuando se deben ordenar los *commits* a realizar, el *staging area* ha recibido algunas críticas por considerar que agrega complejidad innecesaria al modelo conceptual de Git y hace más difícil su uso. [57]

Eficiencia

La velocidad y el rendimiento fueron unos de los objetivos primordiales de Linus Torvalds en el diseño de Git, ya que debía manejar un proyecto muy grande como el kernel de Linux. Prácticamente todas las operaciones son locales, dándole una gran ventaja contra CVCSs como SVN, que necesitan de una conexión con un servidor remoto para prácticamente todas las operaciones. Además Git está hecho en C, por lo que evita el overhead asociado a lenguajes de programación de más alto nivel.

Las siguientes estadísticas corresponden a pruebas entre Git y SVN utilizando el repositorio de Ruby y realizando o intentando realizar las mismas operaciones sobre los dos repositorios, ya que no todos los comandos son equivalentes.

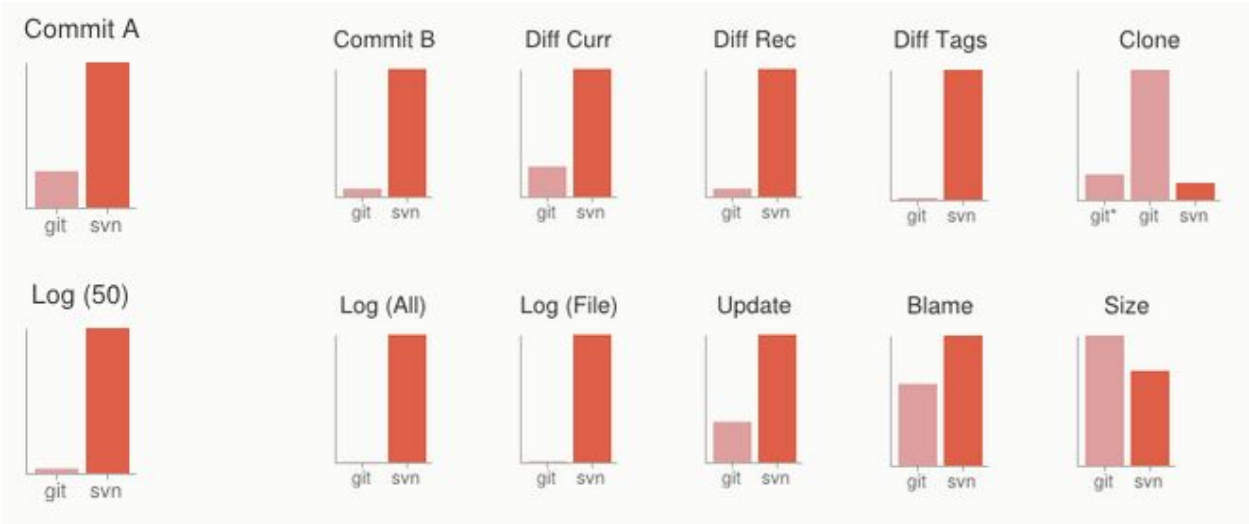

Imagen 4.5 - Comparación entre Git y SVN [55]

Operation		Git	SVN	
Commit Files (A)	Add, commit and push 113 modified files (2164+, 2259-)	0.64	2.60	4x
Commit Images (B)	Add, commit and push 1000 1k images	1.53	24.70	16x
Diff Current	Diff 187 changed files (1664+, 4859-) against last commit	0.25	1.09	4x
Diff Recent	Diff against 4 commits back (269 changed/3609+,6898-)	0.25	3.99	16x
Diff Tags	Diff two tags against each other (v1.9.1.0/v1.9.3.0)	1.17	83.57	71x
Log (50)	Log of the last 50 commits (19k of output)	0.01	0.38	31x
Log (All)	Log of all commits (26,056 commits - 9.4M of output)	0.52	169.20	325x
Log (File)	Log of the history of a single file (array.c - 483 revs)	0.60	82.84	138x
Update	Pull of Commit A scenario (113 files changed, 2164+, 2259-)	0.90	2.82	3x
Blame	Line annotation of a single file (array.c)	1.91	3.04	1x

Imagen 4.6 - Comparación con los tiempos en segundos entre diferentes operaciones [55]

Este escenario de prueba favorece a SVN, ya que se trata de un servidor con buenas prestaciones (ancho de banda de 80MB/s) y sin carga. Casi todos sus tiempos hubieran sido peores si esa conexión fuese más lenta porque depende de la comunicación con el servidor para realizar las operaciones. En cambio Git, al ser distribuido, puede realizar la mayoría de las operaciones localmente, y por lo tanto a pesar de eso, Git es una o dos órdenes de magnitud más rápido que SVN. Sólo en el *clone* inicial del repositorio Git es más lento, ya que descarga todo el historial del repositorio, no sólo la última versión, aunque para ser una operación que sólo se hace una vez no es tan relevante. Otra muestra de la eficiencia es que el tamaño de los datos en el cliente es similar, a pesar de que Git tiene cada versión de cada archivo para toda la historia del proyecto. [55]

Git también supera a otros DVCSs como Mercurial y Bazaar en cuanto a la eficiencia en prácticamente todos los aspectos, como por ejemplo tamaño del repositorio o tiempos de ejecución de comandos como *clone*, *log*, *status* y *diff*. [41]

Libre y Open Source

Git está distribuido bajo la licencia *open source* GPLv2. Es una de las licencias *open source* más utilizadas, aunque por la prohibición de agregar restricciones al software distribuido con esta licencia se puede dificultar su combinación con software bajo otra licencia no compatible, ya que no se permitiría su distribución legal. [58] Esta licencia permite:

- Copiar y distribuir el código fuente del software sin modificar
- Modificar el código fuente del programa y distribuirlo
- Distribuir versiones compiladas del programa, tanto modificadas como no, siempre y cuando: todas las copias lleven una nota de copyright y exclusión de garantía, todas las copias modificadas estén distribuidas bajo GPL y todas las versiones compiladas del programa estén acompañadas del código fuente, o una opción viable para hacer disponible el código fuente.

¿Qué es GitHub?

Para poder colaborar en Git, si bien no es técnicamente necesario ya que se puede realizar *push* y *pull* desde el repositorio de cualquier persona, es usual contar con un repositorio común remoto que provea un punto sobre el cual combinar los diferentes desarrollos y desde donde se obtengan la versiones más actualizadas de los mismos. Esto facilita la colaboración entre los diferentes desarrolladores y permite que cualquiera pueda acceder independientemente del estado de los repositorios particulares. Las diferentes formas de colaborar en un proyecto fueron vistas anteriormente en los *workflows* distribuidos. Hay varias formas de implementar este esquema de colaboración, de las cuales GitHub es la opción más utilizada actualmente.

GitHub es un sitio de hosting de código colaborativo, construido sobre Git. Es el mayor host de repositorios Git, con más de 10 millones de usuarios y más de 24 millones de repositorios actualmente. [59] Estos repositorios pueden ser públicos, utilizados típicamente por proyectos open source y que sólo requieren una cuenta básica, o privados, los cuales requieren en este caso suscribirse a plan pago. A diferencia de Git, que es una herramienta de línea de comandos, GitHub provee una interfaz gráfica web y de escritorio, además de integración móvil. [60]

Su diseño está centrado fuertemente en lo social. Por ejemplo, permite seguir repositorios (*watch*) o usuarios (*follow*), de manera similar a otras redes sociales, para mantenerse actualizado sobre los mismos. Esto incluye, en el caso de un usuario, las últimas acciones en el sitio, y para un repositorio, los últimos cambios sobre el mismo como *commits* e *issues* nuevos. Además da a cada usuario un perfil configurable con información personal, como su nombre, imagen o email, y donde se muestran sus actividades más recientes. [61] Es

posible también descubrir nuevos proyectos usando la funcionalidad *explore*, o compartir *snippets* usando la funcionalidad *Gist*. [62] Esto le da a la plataforma un gran nivel de transparencia en cuanto a las acciones de los demás usuarios y de los proyectos. A su vez tiene implicancias en la forma de colaborar de ellos, ya que les permite tomar conocimiento de diversa información útil y a partir de ella realizar inferencias en cuanto a la calidad de trabajo, el compromiso y la relevancia personal y de la comunidad. Por ejemplo, para los proyectos, pueden inferirse a partir del volumen y actualidad de las acciones el nivel de mantenimiento del mismo y su importancia o popularidad y, junto a la historia general, puede entenderse la estructura del proyecto y los diferentes roles. A partir de la información de los *commits* e información relacionada como comentarios o *issues* puede verse la intencionalidad detrás de cada acción. En cuanto a los usuarios, por ejemplo, la historia de sus actividades permiten inferir su nivel de competencia o las necesidades funcionales en relación a un proyecto en particular, así como también la información sobre los seguidores puede ser interpretada como una señal de status. [63] De acuerdo a estas conclusiones realizadas a partir de las impresiones que se forma cada usuario es que se puede decidir qué seguir o en donde contribuir. [61] Todo esto fomenta la colaboración, el aprendizaje y el manejo de la reputación en la comunidad.

Una de las funcionalidades más reconocidas de GitHub, alrededor de la cual está basado su flujo de trabajo colaborativo, son los *Pull Request*. Como se describió anteriormente, son un mecanismo para manejar la realización de cambios tanto por parte de personas pertenecientes a un proyecto o por terceros, agregando una etapa de revisión de código en el flujo de trabajo. En esta, se genera una conversación sobre esos cambios y en la cual generalmente se realizan más commits basados en el feedback recibido, para eventualmente aplicarlos sobre la línea de desarrollo principal cuando se llega a un consenso. Hay básicamente dos formas de colaborar en un proyecto en GitHub, o bien se utiliza un modelo de repositorio compartido (*shared repository model*), donde se hace un *clone* del proyecto y se modifica ese repositorio directamente, o si se quiere colaborar en un proyecto sobre el cual no se tienen permisos de *push,* se utiliza un modelo de *fork and pull*, donde se realiza un *fork* del proyecto. Se crea de esta manera uno paralelo, al cual se suben los cambios idealmente en un *branch* especial, para posteriormente iniciar un *pull request* sobre el repositorio oficial. Los *pull requests* pueden ser utilizados también por el modelo de repositorio compartido, como un mecanismo para realizar revisión de código. [62] Este modelo es flexible, disminuye riesgos y tiempos en la incorporación de cambios y aumenta las oportunidades para el compromiso de la comunidad. [61]

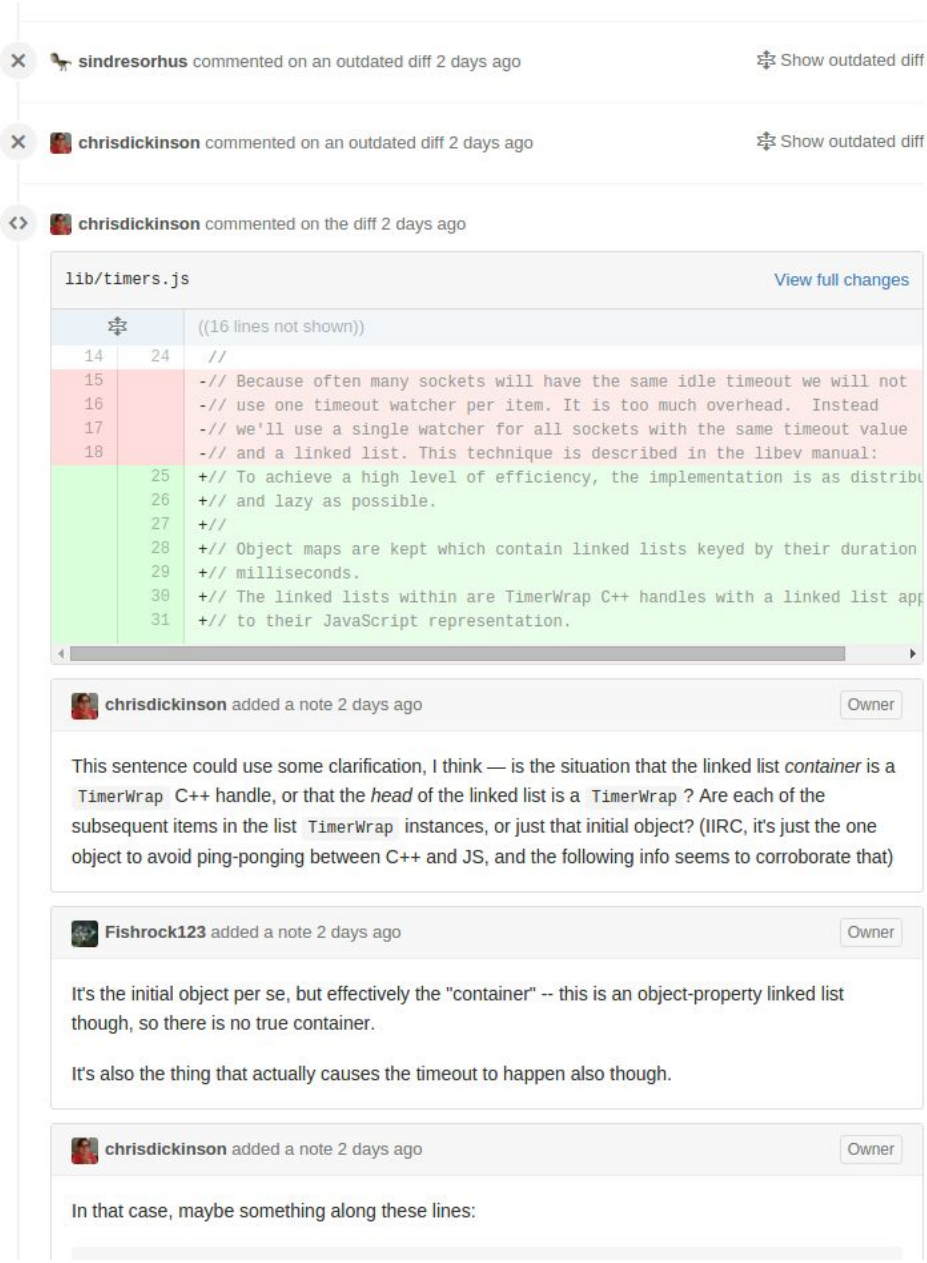

Imagen 4.7 - Comentarios sobre un fragmento de código de un pull request

GitHub Además ofrece todas las características de manejo y control de versiones de código de Git, junto a otras adicionales. Cuenta con *tracking* de *issues* integrado, lo que facilita el manejo de *issues* proveyendo varias funcionalidades útiles, por ejemplo filtrado por varios criterios como por etiquetas o issues cerrados, ordenamiento por fecha o número de comentarios, *milestones*, etiquetas para aplicar a los *issues* y la capacidad de cerrar *issues* a través de los mensajes de los *commits*. Permite también organizar los diferentes equipos de una organización, desde la capacidad de crear proyectos para ella hasta configurar para cada persona permisos de sólo lectura, de lectura-escritura o de administración para cada repositorio existente. [64]

Alternativas

Si bien GitHub es la opción más popular actualmente, hay otras alternativas posibles a la hora de colaborar sobre un repositorio Git y que vale tener presentes, de las cuales éstas son algunas de las más utilizadas:

- Servidor propio: Es posible configurar un repositorio Git para que corra dentro de un servidor propio. Esto se hace primero creando un repositorio *bare*, que es un repositorio que no tiene un directorio de trabajo. Luego sólo queda copiar ese repositorio a un servidor, teniendo cada usuario permisos para clonar el repositorio si tienen acceso SSH al servidor y permisos de lectura sobre el directorio en el que se encuentra, así como también la posibilidad de realizar *push* si tienen permisos de escritura sobre el mismo. Git provee varias alternativas a la hora de definir los accesos. Además es posible activar una interfaz web simple y configurable llamada GitWeb que permite navegar entre las revisiones y archivos de los repositorios de manera análoga a GitHub. [43]

- GitLab: Esta es una alternativa más completa que GitWeb para el caso que se quiera usar un repositorio propio. GitLab es un manejador web de repositorios Git escrito por Dmitriy Zaporozhets en Ruby. Originalmente un proyecto *open source* gratuito distribuido bajo la licencia MIT, [65] fue dividido en 2013 en 2 proyectos diferentes, GitLab CE (*Community Edition*) y GitLab EE (*Enterprise Edition*), aún con la misma licencia, hasta que a partir del 2014 la versión GitLab EE pasaría a tener una licencia propietaria, conteniendo características no presentes en la versión de la comunidad. De manera similar a GitHub, GitLab CE posee las siguientes características y funcionalidades: [66]

    - Open source: Está licenciado bajo la licencia MIT. Esta se originó en el Massachusetts Institute of Technology, y es una licencia de software libre permisiva, que implica que el software puede ser redistribuido dentro de software libre o propietario. [67] El proyecto cuenta con más de 700 contribuidores

    - *Merge requests*: Es el equivalente a los *pull requests* de GitHub, y proveen una funcionalidad similar, como por ejemplo la posibilidad de incluir comentarios en cualquier línea del código

    - Navegador de directorios: Similar al de GitHub, permite ver la última vez que se actualizó cada archivo, por quién y en qué *commit*

    - Manejo de proyectos, usuarios y grupos

    - Otras funcionalidades, como *issue tracking, wikis, feed* con la actividad más reciente y *code snippets*

- GitLab.com: Es una alternativa similar a GitHub en la cual se provee el *hosting* gratuito del repositorio. Utiliza la versión GitLab EE, por lo que cuenta con mayor funcionalidad frente a la alternativa anterior. Una ventaja frente a GitHub es que permite un número ilimitado de repositorios privados, más allá que a grandes rasgos es similar a este y también ofrece servicios pagos. [66]

- Bitbucket: Es un servicio web de *hosting* de proyectos que utilizan tanto Mercurial como Git como VCS. Más allá de también ofrecer planes pagos, ofrece gratuitamente repositorios privados ilimitados. Al igual que las otras alternativas cuenta con *pull requests*, control de permisos y una interfaz para explorar los repositorios.

5. Overview del sistema propuesto

En las partes anteriores se describieron las bases sobre las cuales se va a realizar la definición general de la solución propuesta. Se vio el marco teórico de gobierno abierto, gobierno electrónico y datos abiertos, el ámbito del sistema, es decir el Municipio de La Plata y las características de su legislación, y la herramienta a utilizar, el sistema de versionado Git.

En esta parte se van a analizar y justificar los distintos aspectos de esta definición, nombrando otros sistemas ya diseñados para el manejo de la legislación, describiendo el sistema propuesto y finalmente explicando los 3 pasos o partes del mismo: el uso de Git para representar la legislación, el de GitHub para aumentar la transparencia, participación y colaboración en el proceso legislativo, y la generación de aplicaciones que se basen en los datos y funcionalidades de la plataforma.

Contexto

Debido a las necesidades de diferentes actores con respecto a la legislación, entre los cuales están abogados, legisladores y ciudadanos, varios sistemas informáticos y formatos se han desarrollado específicamente para las leyes. Estas necesidades van desde las profesionales, para que esas personas puedan realizar sus funciones, hasta las de los ciudadanos, para conocer qué leyes están vigentes de manera de por ejemplo conocer sus derechos o llevar a cabo negocios.

Salvo alguna excepciones, los gobiernos deben publicar sus leyes y así darlas a conocer al público como requisito para su obligatoriedad. Tradicionalmente esto se hace a través de publicaciones, por ejemplo el boletín oficial en Argentina[16], las cuales son publicadas periódicamente, y tras lo cual pasando un período que varía en cada lugar, las leyes allí enumeradas entran en vigor. Actualmente mediante el uso de internet se pueden solucionar varios problemas de la distribución tradicional, permitiendo a cualquier persona con acceso a internet entrar en conocimiento de cada ley instantáneamente. Tal es así que en Francia una ley por defecto entra en vigencia luego de ser publicada en su gaceta electrónica. Más allá de esta cuestión de legalidad, varios sitios de diferentes gobiernos permiten a cualquier persona acceder a su legislación a través de la red. [68]

Dependendiendo de cada sistema en particular, varía a qué tanta información se puede acceder sobre las leyes, ya que, a diferencia diferencia de otros documentos como por ejemplo fallos judiciales, la legislación cambia a través del tiempo. Estos cambios se producen por medio de enmiendas, que realizan modificaciones a determinadas leyes, o a causa de alguna ley nueva que además de su contenido particular, puede involucrar pequeños cambios en otras leyes. Generalmente estos cambios no son retrospectivos, es decir, no modifican como era una ley en un tiempo pasado, sino que los cambios son efectivos a partir de su realización. Esto lleva a que surjan varias versiones de una misma ley, las cuales tienen un rango de validez

[16] También llamado en otros lugares diario oficial, boletín oficial, gaceta oficial, registro oficial o periódico oficial.

determinado dependiendo de cuando las enmiendas sobre ella se hayan efectivizado. [69] De esta característica surgen las diferencias en los sistemas a la hora de definir el acceso a las leyes.

En la mayoría de los casos a lo que se accede es a la última versión consolidada de cada ley. Una versión consolidada se forma tomando el texto original de la ley y aplicando los sucesivos cambios que se hayan producido en el mismo a causa de enmiendas u otras leyes. Esta es una parte importante de la publicidad de las leyes, ya que al haber por un lado una ley y por otro muchas enmiendas que no están incorporadas al texto original, se pierde la noción de cuál es el texto vigente de la ley. La consolidación permite conocer su estado preciso y actualizado, favoreciendo la accesibilidad a la misma. [70] Por otro lado, hay sistemas que permiten el acceso a una versión consolidada correspondiente a un período de tiempo particular, lo que se conoce como *point-in-time access*. Un uso concreto de esta funcionalidad se da por parte de los abogados, quienes para para preparar un caso necesitan conocer el estado de las leyes en el momento que se produjeron los hechos, para lo cual deben poder navegar entre la historia de la legislación. Es decir, no sólo es importante contar con la última versión consolidada de la ley, sino también con las versiones anteriores. [71] Igualmente esto no implica que no sea importante o que no se permita el acceso a la legislación tal como fue publicada.

Entre los diferentes sistemas diseñados para el manejo de la legislación que tuvieron en cuenta estas características se puede nombrar primeramente a EnAct del gobierno de Tasmania. El sistema surge en 2004 tras un rediseño del sistema anterior, en el cual se pasa a usar una base de datos de texto no relacional llamada TeraText, la cual almacena los textos de las leyes en formato SGML. Está establecido que esta base de datos contiene la legislación autorizada vigente en el estado. Haciendo uso de ella, EnAct permite un acceso point-in-time a las versiones consolidadas de cada ley. Para lograr esto lo que hace es dividir a cada legislación en fragmentos, que se corresponden con una sección de una ley, a los cuales les agrega fechas que indican para qué período de tiempo está en vigencia. Cuando se realizan enmiendas se crean nuevas versiones de los fragmentos afectados. De esta forma dada una fecha, se puede consolidar la ley obteniendo los fragmentos válidos y combinándolos para formar la versión de ese periodo. Además EnAct permite mediante la edición de una versión consolidada de una ley generar el texto de la enmienda correspondiente a esos cambios. Toda esta información está disponible para ser consultada a través del sitio de la legislación de Tasmania.[17]

Otro sistema que tiene en cuenta la historia de la legislación es el del sitio Legislation.gov.uk el cual permite acceder a la base de datos de las leyes del Reino Unido. Contiene toda la legislación desde 1988, y gran parte de antes de esa fecha, en su mayoría en formato consolidado, aunque quedan cambios posteriores al 2002 por aplicar aproximadamente a la mitad de la legislación. En este sitio se publica toda nueva legislación, casi en simultáneo con la versión impresa. Otra funcionalidad que provee es la búsqueda de cambios en la legislación desde el 2002, ya que los demás ya están aplicados sobre el texto original y no están disponibles para buscarlos. Esto permite listarlos en los casos que estén disponibles, y

[17] http://www.thelaw.tas.gov.au/

ver el texto consolidado y las diferentes versiones a través de una línea de tiempo que sirve de interfaz para un acceso *point-in-time*, junto a *links* hacia la legislación que produjo esos cambios. Además provee una API que permite recuperar la información de cada ley en diferentes formatos.[18]

Lo que estos sistemas dejan entrever es el interés de modelar las leyes y sus cambios a través del tiempo y de permitir su acceso por parte de la ciudadanía. La dificultad en el modelado surge a raíz de varios motivos:

- La particularidad de representar la variabilidad temporal de una ley.

- La necesidad de proveer consolidaciones para contar con las versiones válidas y actualizadas de las leyes de forma que sean más accesibles y entendibles, así como también su registro histórico.

- El enorme volumen de la legislación generada a lo largo del tiempo que complica el proceso de digitalización y consolidación.

- La necesidad de incluir el manejo de la legislación digital dentro de los procesos legislativos ya establecidos, para que posea la información más actual en sintonía con la publicación oficial, o directamente sirva de referencia como ley oficial.

En la mayoría de los casos se tratan de sistemas propietarios que no permiten utilizarlos o modificarlos libremente. Esto reduce su adopción generalizada al no poder aplicarse directamente y dificulta cualquier posible modificación de los mismos para adaptarse a las circunstancias particulares del sistema legal en el que pudiera implantarse. Otra limitación es la dependencia en lenguajes como SGML o XML para definir la estructura de las leyes y proveer metadatos que permitan llevar el registro de las mismas. Si el sistema está construido con un formato especial en mente se pierde versatilidad tanto si se quiere modificar la forma en que se representan las leyes en el sistema en uso, como utilizarlo en casos donde otro formato sería más adecuado.

Fundamentos y descripción general

La solución propuesta por esta tesina para el manejo de la legislación municipal se fundamenta en tres ideas cuya validez se va a intentar justificar haciendo uso de los conceptos explicados en las partes anteriores:

- Al igual que el código de software, la legislación, incluyendo su historia, puede representarse de manera natural y eficiente mediante el sistema de versionado Git.

[18] www.legislation.gov.uk

- Haciendo uso de plataformas colaborativas como GitHub para el *hosting* de la legislación, y a su vez, de los flujos de trabajo que se hacen posibles a través de ellas, se puede mejorar el proceso legislativo aumentando la transparencia, participación y colaboración dentro del mismo.

- A partir de los datos de la legislación y la API de esa plataforma que permite el acceso a sus datos y funcionalidades, pueden desarrollarse aplicaciones por parte del estado y, por sobre todo, de cualquier tercero, que agreguen valor a los mismos o sirvan de interfaz alternativa.

Cada hipótesis se corresponde con una parte o paso para la implementación del sistema. Primeramente, el digitalizar la legislación con un repositorio Git, que implica definir la historia del repositorio, un formato para los archivos, los cuales representarán a cada ley, y la estructura del mismo. De esta forma, actualizando el repositorio con cada cambio que se produzca sobre la legislación, se puede contar con una herramienta para proveer acceso *point-in-time*.

El segundo paso es utilizar una plataforma colaborativa en la cual se aloje el repositorio anteriormente definido. Para esto se va a elegir GitHub. Este paso implica definir el alcance formal que se le va a dar a la utilización de la plataforma, que va desde la simple publicación de la legislación la cual es actualizada periódicamente, su uso por parte de las instituciones y aquellos encargados de legislar, hasta la participación y colaboración de los ciudadanos en el proceso legislativo mediante la generación de *feedback,* sugerencias y propuestas.

La última parte que completa la idea general son las posibilidades que se abren a través de la opción de obtener el repositorio con los datos y acceder a la API de la plataforma, para el desarrollo de aplicaciones que hagan uso de esos datos y las funcionalidades de la misma, como los *pull request*. Estas, si bien pueden ser desarrolladas por parte del gobierno, deberían naturalmente provenir mayoritariamente de terceros, sean empresas, organizaciones sin fines de lucro o individuos, las cuales deberían tener las mismas posibilidades de acceso a los datos. Debido a que las posibilidades que se abren son incontables, se van a analizar algunas de las ideas más inmediatas. Así puede ejemplificarse lo interesante y potente de este esquema, que se basa en la separación de roles en el acceso a datos gubernamentales expuesto en *Government Data and the Invisible Hand*. Allí se propone un esquema que sitúa al gobierno como proveedor de datos y a los actores privados como los encargados de llevarlos hacia los ciudadanos a través de las herramientas desarrolladas por ellos mismos. [30] Todo esto permitiría:

- Habilitar el acceso libre a los datos de la legislación. Esto facilitaría el uso de los mismos por los ciudadanos y aquellos con necesidad de conocer la historia de la legislación como abogados o concejales, y la creación de nuevas aplicaciones que hagan uso de los datos

- Aumentar la transparencia del proceso legislativo

- Indirectamente aportar a la inclusión digital porque invita a que todo aquel que tenga la iniciativa utilice y ejercite las nuevas tecnologías de información y comunicaciones

- Permitir un cierto control por parte de la ciudadanía sobre aquellos a quienes se delegó el gobierno del municipio al poder ver los proyectos en los cuáles están trabajando actualmente

- Facilitar la participación y colaboración entre los ciudadanos y el Gobierno, y entre los mismos concejales, haciendo uso de las facilidades que proveen estas tecnologías.

Alcance

Como ya se mencionó en la introducción, se propone utilizar el sistema de versionado Git sobre la legislación municipal de La Plata, lo que implica conocer las características particulares de ella. El alcance elegido, en cuanto a que se elige un municipio, tiene como justificativo que es importante pensar el sistema para un caso concreto. Ya que un municipio es la entidad administrativa más pequeña, gran parte del cuerpo de legislación perteneciente al mismo va a referirse a cuestiones locales, más directas y cercanas que una ley federal que aplica a todo un país, y por lo tanto, aquellas leyes locales van a ser generalmente menos complejas. Así es más factible que haya un mayor interés por parte de las personas de conocer qué leyes municipales rigen para su lugar de residencia. Por otro lado también, de que se sienta una mayor cercanía en cuanto a sus representantes municipales y por lo tanto se quiera conocer sus proyectos y funciones. Y finalmente, que sea más fácil aportar sugerencias para realizar modificaciones o generar nueva legislación, ya que se tratan de cuestiones de menor alcance y por lo tanto más simples, accesibles al entendimiento de una persona y más cercanas. De igual forma, la utilización de un sistema a nivel municipal tiene un menor impacto que uno a un nivel superior, tanto en cuanto a lo territorial, poblacional e institucional. Ya que se trata de un enfoque nuevo para el problema de la legislación es razonable experimentar con su uso en un alcance limitado.

Debido a que se trata de un municipio de la provincia de Buenos Aires, la voluntad del concejo deliberante puede ser canalizada de 4 formas posibles: ordenanzas, decretos, resoluciones y comunicaciones. De todas ellas, debido a su importancia y carácter de ley local, [1] se va a centrar el análisis en las ordenanzas para todas las cuestiones del repositorio, ya que de ellas es que surge la necesidad de registrar de alguna manera los cambios sobre la legislación a lo largo del tiempo. Esto no implica necesariamente que no se pueda, o no se vayan a representar decretos, resoluciones y comunicaciones dentro del mismo, sólo que en el caso de estos el nivel de complejidad es menor y por lo tanto puede derivarse en gran parte del análisis realizado para la ordenanzas. Si bien el alcance de esta tesina se limita a la legislación perteneciente al municipio de La Plata, los principios e ideas generales son extrapolables a otros contextos, ya sean otros municipios, gobiernos provinciales o federales, u otras organizaciones o instituciones que tengan reglas o leyes que las regulen.

Git para representar a la legislación

La primer y más importante cuestión y a definir es si Git es apto para representar a la legislación con sus particularidades. Para esto primero hay que partir de que el código y la legislación son análogos en ciertos aspectos. Ambos son grandes conjuntos de texto divididos en partes más pequeñas, que sufren pequeños cambios a lo largo del tiempo, mientras la mayor parte permanece igual. Pero por otro lado, al igual que en caso del código, se producen distintas versiones de las normas, como ya se mencionó en la sección anterior. Por ejemplo, a partir de una norma A_1 válida a partir de una fecha dada, que es modificada en alguno de su artículos por otra norma B posterior, se produce una versión A_2 diferente de la original, llamada versión consolidada, formada a partir de A_1 y la aplicación de los cambios introducidos por B. Ambas son importantes, A_2 porque representa la norma vigente actualmente y por lo tanto es aquella que hay que cumplir, y A_1 porque, por ejemplo, en el caso de un juicio que tenga que expedirse sobre un hecho cometido anteriormente a la entrada en vigencia de A_2, va a ser la versión que estaba en vigencia en ese momento la cual debe tenerse en cuenta, osea A_1. Es por esto que es necesario poder acceder a todas las versiones, que en el caso de un sistema con este propósito significa proveer acceso *point-in-time*. [72]

Este problema es también análogo al que sucede en el desarrollo de software, donde es necesario llevar un registro de los cambios en el código mediante un VCS. El hecho de que la legislación y el código estén compuestos por texto no traería mayores inconvenientes para este análisis, ya que Git está diseñado para manejar contenido en forma de texto. Como se explicó, generalmente el enfoque elegido para diseñar un sistema que provea acceso *point-in-time* es el de dividir el texto de cada ley en fragmentos a los cuales se les asocian fechas que indican el rango de tiempo en el cual son válidos. De esta forma, al producirse un cambio en alguna de esas partes en una fecha dada, se indica para el registro del fragmento a reemplazarse esa fecha de fin, y se crea un nuevo registro de los nuevos datos con esa fecha de inicio. Así se minimiza la redundancia de guardar el texto completo de las versiones de cada ley y se permite rearmar las diferentes versiones realizando búsquedas sobre estos fragmentos de acuerdo a las fechas en ellos indicadas. Esta búsqueda es computacionalmente compleja, y además vuelve al sistema dependiente del formato utilizado para representar las leyes, usualmente SGML y XML.

El paradigma bajo el cual se representaría en Git contrasta con el utilizado normalmente, y tiene que ver con la forma en que conceptualmente maneja sus datos. En vez de versionar cada ley mediante la ley completa o sus fragmentos, lo que se guardaría serían los *snapshots* de toda la legislación en un momento dado. Es un contraste similar al de Git con otros sistemas de versionado de código como SVN en los cuales los que se guardan son los *deltas* entre los diferentes archivos. Si bien es cierto que Git periódicamente empaqueta sus objetos y aplica *deltas* para utilizar menos el espacio, esto se hace a bajo nivel y por una cuestión de eficiencia, por lo que se entiende que su modelo de objetos y comandos conceptualmente se basa en *snapshots*.

Este es un enfoque más natural para la representación de las leyes, ya que si bien puede pensarse a alguna ley individualmente, forman parte de un sistema legal, y por lo tanto están conectadas entre sí. Es este sistema en su conjunto el que rige por sobre la sociedad en un período dado, y son sus diferentes estados los que se necesita conocer. Más aún, a veces los cambios que se dan en la legislación involucran varios cambios menores sobre diferentes leyes.

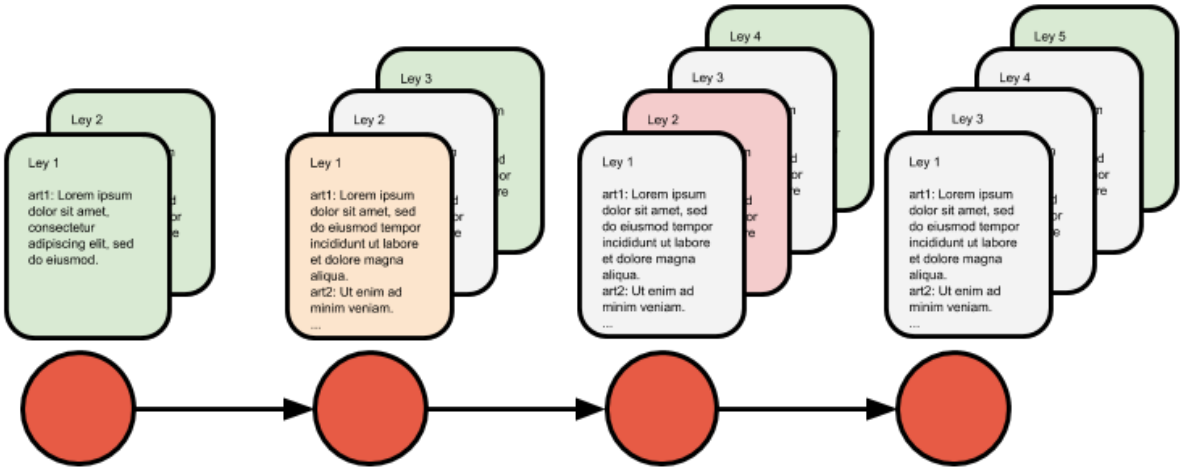

Imagen 5.1 - Diagrama conceptual del repositorio Git de una legislación con 4 commits representando sus diferentes estados a lo largo del tiempo. En cada uno se agregan, modifican o derogan leyes.

Siempre que surgiese una nueva ley o modificación a leyes existentes se crearía un nuevo *commit* que representaría ese nuevo estado de la legislación. Esto debería coincidir idealmente con su tiempo de entrada en vigencia, por ejemplo cuando aparecieran en la publicación oficial, lo que implicaría agregar este paso de actualización del repositorio en la parte del proceso que corresponda. De esta forma, para conocer el estado de la legislación en un período dado, bastará con posicionarse en el *commit* adecuado. Esto facilita cualquier tipo de análisis que se quiera hacer sobre la legislación pasada, y es algo que en otros sistemas no es posible o simple, ya que el enfoque está en conocer la versión de una ley en particular. Así por ejemplo, un abogado que necesitase conocer las leyes vigentes en el momento de un hecho tendría acceso instantáneo al conjunto de leyes válidas para ese período, y no debería tener que realizar búsquedas individuales para cada una de ellas. Dado que las ordenanzas son leyes [1] este esquema es aplicable a las mismas, y por lo tanto el caso elegido del municipio de La Plata es válido para poder realizar un análisis concreto de la idea aquí expresada.

La utilización de Git además puede solucionar gran parte de los problemas mencionados para los sistemas tradicionales, dado que en cuanto a las cuestiones generales ofrece varios beneficios por sobre ellos, muchos de los cuales ya fueron explicados en la sección de Git:

- Está disponible en todos los sistemas operativos principales.

- Es *open source*, lo que significa que puede utilizarse casi sin restricciones y llegar a modificarse en caso de que sea necesario adaptarse a alguna necesidad particular para su uso con la legislación.

- Más allá de esas posibles modificaciones, como se vio anteriormente, Git tiene muchos puntos de extensión y configuración que pueden bastar para la mayoría de los casos.

- Es el sistema de versionado más utilizado en el mundo por lo cual, hay innumerable información y herramientas que facilitan su aprendizaje y uso. De igual forma se asegura su desarrollo, mejora continua, y eventualmente las facilidades necesarias para migrar hacia otro futuro sistema.

- Es más eficiente y performante, en casi todos los aspectos, que otros sistemas de versionado.

- Es independiente del formato de los archivos del repositorio. Mientras sea un formato de texto se pueden obtener todos los beneficios del control de versionado. Esto da un mayor grado de libertad a la hora de elegir uno.

La forma en la que la legislación es representada, el formato utilizado para cada ley, qué datos se agregan al archivo, cómo se estructura el directorio, qué datos se indican en los mensajes de *commits* y quienes y cuando se realizan, todo esto, es independiente de la utilización de Git. Estas cuestiones van a analizarse para las ordenanzas en las siguientes partes de la tesina. En resumen, esta parte que involucra el manejo de la legislación en Git permitiría:

- Acceder a la legislación actual y pasada
- Resguardar los datos. Al estar replicados en cada repositorio personal se evita tener un "*single-point of failure*"
- Reutilizar toda esta información de la legislación clonando el repositorio
- Saber cuándo y qué cambios se produjeron sobre la legislación
- Actualizar todos los repositorios cuando se introdujesen cambios en la legislación
- Asegurar la integridad de los datos mediante el código *sha-1* y firmas de *commits* o *tags*

Una plataforma sobre la cual interactuar con el repositorio

Una vez definido cada aspecto del repositorio y registrado en el mismo cada estado de la legislación, debe precisarse la forma en que se va a permitir el acceso y el uso del mismo por parte del resto de la sociedad. El acceso a un repositorio Git puede proveerse de varias formas, puede configurarse un servidor Git, una aplicación web como GitLab, o utilizarse un sitio de *hosting* externo como GitHub o GitLab.com. Estos últimos ofrecen ciertos beneficios por sobre los demás, ya que son fáciles y rápidos de configurar, no implican un gasto de mantenimiento, y facilitan la colaboración con el resto de la comunidad al tener mucha visibilidad. [43] Además proveen funcionalidad y herramientas, algunas externas, que potencian su uso. Se va a tomar el caso de GitHub a través de la tesis, ya que es el *host* de código más popular del mundo, y si bien tiene ciertas ventajas por sobre otras opciones, no se descarta el uso otra aplicación que provea funcionalidad similar, como GitLab.com.

El impacto de GitHub como plataforma colaborativa en el desarrollo, sobre todo en el ámbito de proyectos *open source*, ha sido tan grande que han surgido numerosos proyectos e ideas desde otras disciplinas para utilizarlo como herramienta. Estos van desde las investigaciones y publicaciones científicas [73] [74], la educación [62], y otros proyectos entre los cuales se incluyen aquellos relacionados con la legislación. [75] La conclusión que se puede sacar de todos estos casos es que, a pesar de que existen dificultades y problemas que resolver para que su uso sea óptimo, la plataforma que provee GitHub posee un gran potencial para su utilización en otros proyectos fuera del desarrollo del software. Esto se da principalmente por su filosofía abierta enfocada en la transparencia y colaboración junto a un diseño social y herramientas como los *pull request*. Es esta forma de trabajo la que es aplicable a otros ámbitos y la que posiblemente guíe los futuros cambios a producirse en GitHub, de forma de que se facilite un uso más general. Esto es algo que ha sido remarcado por sus fundadores. [76]

El primer beneficio inmediato de su uso para nuestro caso, sería el de proveer un punto de acceso confiable a la información de la legislación. El publicar un repositorio Git en GitHub es trivial, y a partir de allí es posible para cualquiera inspeccionarlo, realizar un *fork* del mismo o descargarlo. Puede pensarse al órgano encargado de realizar las publicaciones oficiales como aquel con los permisos para realizar las modificaciones sobre el repositorio. Así se contaría con una versión digital, mucho más eficiente y accesible que la material, además de con un resguardo del repositorio. Se permitiría la sincronización con el repositorio oficial a aquellos que realizaron un *fork* o descargaron el repositorio, para cuando se realicen cambios sobre el mismo. Esto facilitaría la reutilización de los datos al proveerles una fuente de información actualizada de fácil acceso. Para el caso de alguien al que sólo le interesa inspeccionar la legislación y su historia, la interfaz de GitHub es adecuada para esa tarea al ser simple e intuitiva, al menos mucho más que la de Git, para alguien sin conocimientos.

Un segundo paso sería empezar a hacer uso del resto de las funcionalidades y del diseño social de GitHub incorporando a los concejales y la diferentes entidades, como las comisiones, a la plataforma, para poder conocer los proyectos futuros y los que se están tratando para de esa forma tener un mayor detalle del proceso legislativo. Un modelo

colaborativo posible podría ser uno donde cada uno de ellos tuviera un *fork* del repositorio oficial con la legislación, en el cual subieran los proyectos en los que estuvieran trabajando, actualizandolos convenientemente. Esto requeriría un mayor nivel de involucramiento por parte de las instituciones en el uso del sistema, pero por otro lado podría mejorar la forma de trabajo de y entre los diferentes concejales, abriendo nuevos canales de colaboración entre ellos. Además de esta forma se llegaría a un nivel mayor de transparencia, ya no sólo de la legislación presente y pasada, sino de la que está desarrollándose, ya que cualquier persona podría ver en qué y cómo está trabajando cada concejal. También esto deja abierta la posibilidad de pensar un esquema que represente el flujo del proceso legislativo, desde la elaboración del proyecto, su paso por las cámaras y las comisiones y su eventual aprobación, agregando un elemento más a la información de la legislación.

El último aspecto a analizar es la incorporación de *feedback* de los ciudadanos a través de la plataforma. Este *feedback* puede darse de varias formas, de las cuales las dos principales en GitHub son los *issues* y *pull requests*, que ya fueron descriptos anteriormente. A través de *pull requests* se permite la colaboración en la construcción de los proyectos en los que estén trabajando los concejales. Contando con un *fork* del repositorio de ese concejal, y realizando la modificación que se quiere sugerir, puede crearse un *pull request* sobre el repositorio original. Así, cada persona con conocimientos o interés sobre el tema que ese proyecto abarque puede realizar su aporte y propuesta, la cual puede ser tomada en cuenta e incorporada. También puede darse proponiendo cambios a legislación vigente o legislación nueva por fuera de los proyectos que estén elaborando los concejales. Nuevamente, estas propuestas pueden o no ser tomadas por alguno de ellos. Así se establecería para la sociedad un canal de participación y colaboración en el proceso legislativo. Cabe destacar que además de esta relación entre el gobierno y los ciudadanos, pueden darse casos de colaboración entre las personas por ejemplo para redactar un nuevo texto de una ley, que son también facilitados por la plataforma.

Una plataforma online para acceder y trabajar sobre el repositorio es vital para poder obtener todos los beneficios posibles del uso de Git sobre la legislación. En resumen esto posibilitará:

- Acceder a los datos de manera online sin necesidad de descargar el repositorio
- Usar para el trabajo legislativo diario de los concejales, permitiendo a los ciudadanos ver en que están trabajando, y facilitando y abriendo nuevos canales para la colaboración entre los concejales
- Permitir la participación de la ciudadanía, aportando *feedback* sobre la legislación actual y los proyectos en los cuales trabajan los concejales
- Permitir la colaboración, para que cada ciudadano pueda trabajar activamente en la creación de nuevas propuestas de normas entre ellos y junto a los concejales

- Hacer uso de las herramientas que provee la plataforma
- Abrir el camino a la utilización de herramientas relacionadas existentes o a desarrollarse, que faciliten el uso de la plataforma o hagan uso de los datos

Con estos tres aspectos, la publicación en GitHub, su uso por parte de los concejales, y la generación de *feedback,* se aportaría a los 3 pilares de gobierno abierto: transparencia, participación y colaboración. Todo esto enmarcado en una iniciativa que puede considerarse de datos abiertos ya que se permite el acceso digital a la información gubernamental, y de gobierno electrónico ya que se abre una canal de comunicación haciendo uso de las TIC entre las personas y las instituciones. De cualquier forma que se lo enmarque, el objetivo es proveer más información de la que se pueda estar accediendo normalmente y fomentar la participación y colaboración de la sociedad en el proceso legislativo.

Aplicaciones de terceros

Un patrón recurrente en el desarrollo de software es la separación entre datos y presentación, que permite el desarrollo independiente de ambas partes y una mayor extensibilidad, lo que facilita ese desarrollo. En el caso de datos gubernamentales, a la hora de publicarlos *online*, tradicionalmente se elige un esquema donde un sitio del gobierno es el encargado de proveer acceso a ellos. Toda interacción debe pasar necesariamente por este. Esto conlleva varios inconvenientes, principalmente por la falta de actualización y adaptación a la nuevas tecnologías, la excesiva burocracia y normativa que restringe el rango de acción de los sitios a implementarse, y la imposibilidad de prever o proporcionar todo el rango posible de interacciones con los datos o las mejores maneras de hacerlo. En contraposición, y siguiendo el patrón de separación entre datos e interacción, puede pensarse una arquitectura donde se considera al gobierno principalmente como un proveedor de datos. Su rol en este caso es el de proveer una infraestructura simple, pública y confiable que exponga los datos subyacentes, quedando del lado de actores privados llevar la información hacia los ciudadanos. Estos pueden implementar soluciones, adaptarse a los cambios tecnológicos y explorar las diferentes posibilidades de interacción de una manera más eficiente que el gobierno. [30]

Son innegables y han sido enumerados los beneficios que aporta el uso de GitHub al desarrollo de software. Pero, si bien ha habido experiencias en las cuales se lo utiliza para otras actividades como la creación de libros, artículos o legislación, hay barreras iniciales que son difíciles de superar para aquellos usuarios no técnicos en comparación con otras plataformas colaborativas. Son los principios subyacentes a GitHub como la transparencia, colaboración, versionado y ciertas funcionalidades como *pull requests* los que se ven como positivos y útiles para todo el otro rango de usos fuera del software. Es por eso que puede ser beneficioso el contar con una capa de abstracción por sobre aquellos aspectos complejos que

dificultan su uso. [75] Afortunadamente, GitHub provee una API robusta que provee un acceso a prácticamente cualquier información, como por ejemplo:

- *Commits, branches, pull requests, issues*, étc. de cualquier repositorio
- Crear y realizar el *merge* de *pull requests*
- Información de los usuarios
- Modificar los objetos internos de Git mediante la Git Database API
- Acceder a notificaciones y eventos

Es de esta forma que lo expresado en cuanto a la forma de proveer datos gubernamentales puede llevarse a cabo. A través de esta API se puede acceder a los datos del repositorio, es decir, a los datos de la legislación. Así puede ser posible desarrollar una aplicación web que sirva de interfaz simplificada que facilite el uso de las funcionalidades requeridas por parte de los ciudadanos, legisladores y la institución encargada de actualizar el repositorio, como por ejemplo los *pull requests*. También se abren posibilidades para el desarrollo de aplicaciones que tengan otros fines, como una que muestre datos estadísticos, o entrecruce los datos de la legislación con otros, u otras posibilidades que son difíciles de prever, pero que este modelo soporta y motiva. A su vez, de esta forma no se excluye el uso de GitHub, ya que se puede acceder a toda la información de la legislación y colaborar de cualquiera de las dos maneras.

Si bien esta interfaz puede ser desarrollada desde el gobierno municipal, el hecho de contar con una API pública habilita a cualquier otra persona o institución desarrollar la propia. Así, por más que haya una aplicación oficial inicial, si hay diversas aplicaciones para la misma tarea es posible elegir aquella que mejor diseñada esté, o que provea alguna funcionalidad novedosa. Se evita el modelo clásico gubernamental antes mencionado de una sola aplicación para acceder a los datos, que en la mayoría de los casos lleva a diversos problemas si, como suele suceder, sufre poco mantenimiento o hay una falta de innovación. Es importante para esto que la aplicación oficial utilice la misma API que se provee al resto de los actores, de forma de tener una competición justa, y evitar que la aplicación oficial se nutra de datos e información no accesible al resto de las personas. Si se sigue este modelo se crea un ambiente propicio para la innovación lo que a su vez lleva a mejores aplicaciones, y por lo tanto a mayores beneficios para los usuarios. [30] No sólo a través de una API pueden proveerse los datos para desarrollar aplicaciones independientes, una alternativa es ofrecer los datos en lote. En ciertos casos es más eficiente contar con los datos que realizar consultas a través de una API cada vez que se quiera acceder a los mismos. [5] La descarga del repositorio puede satisfacer este tipo de necesidad.

De esta forma se permitiría:

- Facilitar el uso por parte de la mayoría de las personas
- Abstraerse de aquellos detalles y funcionalidades de GitHub que no encajarían en el trabajo sobre la legislación
- Desarrollar y experimentar con nueva funcionalidad que pueda llegar a ser útil o a agregar valor a los datos

Comparación con el sistema actual

Actualmente, a través de la página del concejo deliberante, se puede acceder a un digesto digital que permite consultar las ordenanzas, ordenanzas generales[19] y decretos vigentes en el partido de La Plata. Esta es la funcionalidad principal que provee, y a pesar de no contar con la totalidad de la legislación municipal como posee el digesto tradicional, sí posee la mayoría de las más relevantes y actuales, aproximadamente 5000. [2] El sitio almacena sus datos en una base de datos Access, y está realizado con ASP. El contenido del sitio se generó a partir de una carga inicial con las normas vigentes realizada hace aproximadamente 10 años, sobre las cuales se fueron agregando las ordenanzas promulgadas, eliminando aquellas derogadas, y consolidando el texto de aquellas que fueron modificadas por otras ordenanzas, indicando con un vínculo en ese caso la ordenanza que lo produjo y para el caso de artículos derogados marcandolos además en rojo. De esta forma se genera un texto ordenado para las ordenanzas. Esta actualización se realiza mensualmente luego de la publicación en el boletín oficial mediante un programa en el que se ingresan todos los datos. [101]

[19] Así se denominan aquellas ordenanzas promulgadas durante la dictadura militar. De las aproximadamente 300 originales sólo quedan unas 40 en vigencia

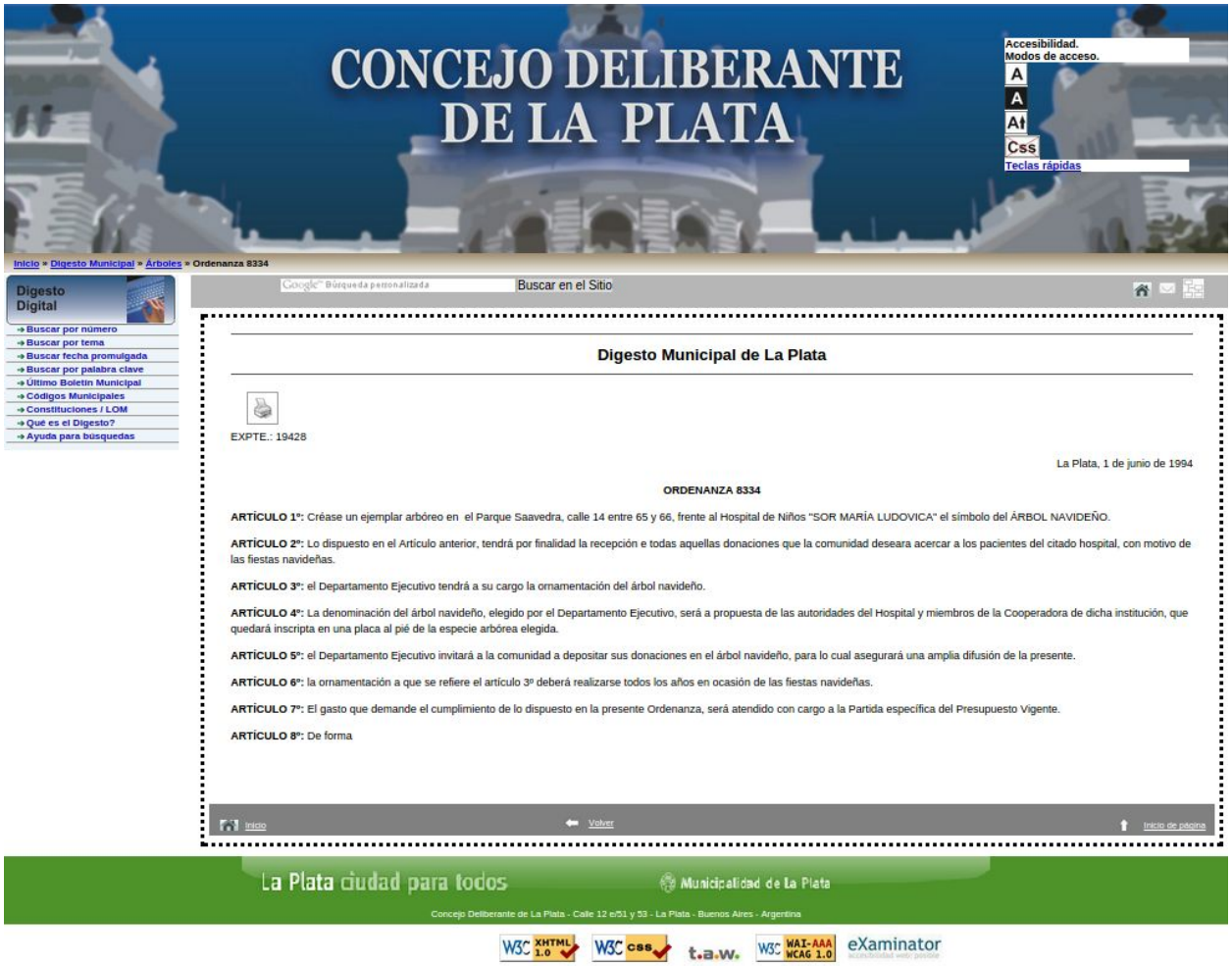

Imagen 5.2 - Una ordenanza vista en el digesto digital

Estas se visualizan como se muestra en la imagen 5.2, con su número y número de expediente, su fecha y el contenido de la misma. En la imagen 5.3 se puede ver un ejemplo de ordenanza donde han sido modificados algunos de sus artículos. Se indica para cada uno el número y un *link* a la norma que produjo esa modificación, y para el caso de artículos derogados, estos aparecen pintados de rojo, mostrándose como en el caso anterior una nota al inicio de la ordenanza con la ordenanza o decreto que lo ha producido y un *link* a ella. A su vez, en los artículos pertenecientes a la ordenanza que modifica se incluye un *link* a aquella que está siendo modificada.

ARTÍCULO 346: El Departamento Ejecutivo deberá remitir anualmente al Concejo Deliberante, un informe evaluativo de las actuaciones municipales en relación al cuerpo normativo vigente en materia de Planeamiento Territorial, referido a los desajustes territoriales en función de la dinámica de crecimiento y la verificación del cumplimiento de los objetivos planteados en la fundamentación de la presente Ordenanza.

ARTÍCULO 347: Si el Departamento Ejecutivo, considera que de las conclusiones emanadas del informe elaborado en los términos especificados en el artículo anterior, surge la necesidad de efectuar modificaciones a la Ordenanza, deberá elevar al Concejo Deliberante las propuestas normativas correspondientes para su tratamiento.

CAPITULO 3. ACCESO A LOS ACTOS URBANÍSTICOS.

ARTÍCULO 348: Se garantiza a toda persona interesada el libre acceso a los expedientes que tramitan en el Municipio referidos a los actos y normas urbanísticos que regula el presente Código.

ARTÍCULO 349: A efectos del artículo precedente la petición deberá hacerse ante la oficina en la que el expediente se encuentre o ante la autoridad de aplicación de la presente ordenanza, por escrito, especificando nombre y domicilio.

ARTÍCULO 350: La petición formulada deberá ser contestada en el plazo de 10 días. La respuesta contendrá el período por el que se pone a disposición del interesado el expediente requerido, debiendo notificarse fehacientemente dicha decisión.

Registros de Oposición

ARTÍCULO 351: El Departamento Ejecutivo reglamentará los Registros de Oposición requeridos por la presente y por las normas provinciales de aplicación.

DISPOSICIONES ADMINISTRATIVAS

ARTÍCULO 352: Las normas del Título VIII –INTERVENCIONES Y PROCEDIMIENTOS DE APROBACION- entrarán en vigencia a partir de su publicación en el Boletín Municipal.

ARTÍCULO 353: Las normas que fijan las alturas de referencia determinadas en los Títulos V y VII, entrarán en vigencia a partir de la sanción de la presente Ordenanza.

ARTÍCULO 354 (Texto según Ordenanza 10896**):**El Departamento Ejecutivo realizará dentro del término de un año a partir de la Convalidación del Poder Ejecutivo Provincial y la entrada en vigencia del presente Artículo, un estudio pormenorizado de las manzanas, y grupo de ellas que conformen un área característica de todo el casco fundacional, estableciendo y determinando cual es la figuración, vista y densidad actual.
Una vez concluido este estudio, el Departamento Ejecutivo dentro de los siguientes noventa días enviará al Concejo Deliberante un proyecto de Ordenanza estableciendo una densidad máxima por manzana del casco urbano.
Asimismo, el Departamento Ejecutivo propenderá un estudio de prefiguración de cada cuadra de la Ciudad, a los efectos de modelizar y prefigurar su condición en relación con el espacio libre público , reglamentando las pautas de crecimiento y desarrollo a futuro, mediante la incorporación al Código de Ordenamiento Territorial y Uso del Suelo de indicadores, planos límites y demás restricciones a las parcelas que las integran, como así también a las calles, ramblas, plazas y plazoletas que las integran.

Imagen 5.3 - Parte del código de ordenamiento urbano donde se ve un ejemplo de artículos modificados y derogados

El sitio además permite realizar las búsquedas de diversas formas: por número y tipo de de norma (ordenanza, ordenanza general o decreto), por temas (se debe seleccionar de una lista de 287 temas, que listará todas las ordenanzas pertenecientes al elegido), por fecha de promulgación (devuelve un listado con todas las ordenanzas promulgadas entre la fecha introducida y la fecha actual), por palabras claves o también puede usarse una barra de búsqueda de Google, que buscará las palabras solicitadas dentro de todos los textos del sitio. En todas las consultas las normas legales aparecen listadas desde la más reciente.

Además de permitir acceder a las ordenanzas, se provee el último boletín oficial, que es un listado con links a las ordenanzas que lo componen. También los diferentes códigos municipales (de Ordenamiento Urbano y Territorial, de Uso del Espacio Público, de Edificación,

de Faltas Municipal, Tributario y de Nocturnidad) y las constituciones nacional y provincial junto con la ley orgánica de las municipalidades. De estos códigos sólo está accesible actualmente[20] el Tributario, dando los demás errores cuando se quiere acceder a los mismos.

Si bien tiene aspectos positivos, como proporcionar textos consolidados de las ordenanzas, vínculos a legislación adicional y opciones de búsqueda, lo cual es suficiente información para la mayoría de los casos de uso, posee los mismos problemas antes mencionados resultantes de la falta de consideración del aspecto temporal de las normas. Al no tenerlo en cuenta, la información de la legislación queda condensada dentro de los límites que permite una sola versión. Cada vez que se produce un cambio, este es aplicado sobre lo que ya se tiene, perdiendo el estado anterior. Esto lleva a que toda la información que en el caso de usar Git quedaría distribuida en los diferentes *commits* y las diferencias entre ellos, deba ser indicada en el texto de cada ordenanza, como por ejemplo la información de que un artículo fue modificado o derogado. Además, se pierden todos los estados intermedios de la legislación, por lo cual la reconstrucción de un estado particular se vuelve dificultoso o imposible.

Un aspecto que no está presente tampoco es la capacidad de participación y colaboración por parte de cualquier ciudadano. Si bien podrían incluirse los proyectos de los distintos concejales, esto no equivale a las posibilidades que abren con el uso de una plataforma como GitHub, donde hay un control más fino de los cambios, mayores herramientas de *feedback* y participación, que abren nuevos flujos de trabajo y colaboración.

Otro aspecto a considerar es la arquitectura elegida, que corresponde al esquema tradicional donde todas las capas del sistema son controladas y desarrolladas por el gobierno mismo. Esto se contrapone con el esquema de datos abiertos antes enunciado, donde cualquier actor privado puede generar su aplicación construyendo sobre los datos provistos . De esta otra forma se aportaría mayor dinamismo a búsqueda de formas de visualizar esta información y permitiría un mayor grado de innovación en este aspecto, multiplicando los usos y agregando valor a los datos, todo esto facilitando el uso de tecnologías más modernas.

[20] Noviembre 2015

6. Casos de referencia

Siempre es recomendable cuando se está en la búsqueda de una solución a un problema ver qué soluciones fueron encontradas por otros para ese, u otros similares. Si bien este tipo de solución para el problema de la representación de la legislación no ha sido implementada oficialmente hasta la fecha, hay varias experiencias por parte de individuos que siguen en parte el enfoque de esta tesina.

En esta parte se van a enumerar y describir algunos casos a tener en cuenta que utilizaron en parte este enfoque para diversos conjuntos de leyes en varios países. Esta enumeración permite tener un contexto de las soluciones aplicadas, las razones detrás del diseño de las mismas, y sus resultados. A partir de esto se va a poseer una base experimental para poder realizar el diseño del sistema de una manera más fundamentada.

Open Congress

Open Congress[21] es un sitio perteneciente a la Sunlight Foundation[22] que permite seguir fácilmente la legislación en el Congreso de los Estados Unidos durante todo el proceso legislativo, y además provee información de los votos en las cámaras y de los senadores y representantes. Open Congress es un sitio *open source* y está hosteado en GitHub[23]. Todos los datos de los cuales se nutre son de fuentes oficiales y son actualizados diariamente, permitiendo a quien quiera acceder a los mismos a través de una API.

Al ingresar al sitio se proveen varias alternativas, se puede buscar por una ley específica, un tópico, por senador y representante, o por comité. Al acceder a una ley se muestra quien fue el que la propuso, el estado actual dentro del proceso, datos de las últimas votaciones y el texto completo de la misma, mostrando las diferentes versiones por las que fue pasando dentro del Congreso. Para cada legislador se muestran sus datos personales, las leyes sponsoreadas por él y aquellas hechas en conjunto, registro de votos junto a datos de interés sobre los mismos, cómo con qué otras personas suele votar o el porcentaje de acompañamiento en los votos para con el partido. Además se ofrecen datos de los contribuidores para su campaña, pudiendo saber qué tanto dinero recibió de algún grupo de interés, de manera de revelar alguna relación entre esas contribuciones y los votos realizados por él. Entre otra funcionalidad el sitio permite registrarse y comentar, participar en discusiones con otros usuarios o seguir una ley para recibir actualizaciones sobre su estado.

Open Law

[21] https://www.opencongress.org/
[22] http://sunlightfoundation.com/
[23] https://github.com/opencongress/opencongress

Open Law [24] es una iniciativa de la ciudad de San Francisco para publicar sus leyes en formatos digitales que las hagan más accesibles y reusables por cualquier aplicación. La idea es que este sea el primer paso para que surjan estas aplicaciones que proporcionen un mejor entendimiento y uso de los datos de maneras creativas, como por ejemplo para realizar búsquedas más efectivas o descubrir leyes obsoletas. Esto se alinea con la política de datos abiertos de la ciudad. [77] Las leyes se proveen a través de varias fuentes:

- Datos en lote desde la OpenGov Foundation en formato JSON [25]
- API de la OpenGov Foundation
- Los códigos municipales oficiales desde la American Legal Publishing [26]
- Archivos de texto sin formato desde GitHub

Al entrar al repositorio se pueden ver varios archivos con un nombre representativo del tema que engloba, con extensiones *.txt* y *.rtf*. Al no estar formateados, las leyes que contienen no tienen una estructura definida por lo que son difíciles de leer, sobretodo cuando se quieren ver los cambios entre las diferentes versiones

Bundes-Git

Bundes-Git, que significa Git federal, es un repositorio Git de todas las leyes y regulaciones alemanas federales hosteado en GitHub.[27] Fue creado en 2011 por Stefan Wehrmeyer, un ingeniero de software que, entre otras cosas, trabaja para la Open Knowledge Foundation Germany. La idea es que si bien los ciudadanos alemanes pueden acceder a una versión actual de los datos de manera online, el proceso legislativo, la evolución y los cambios sobre las leyes no son fácilmente accesibles, ya que estas sólo son publicadas en su versión más reciente y sus cambios no están disponibles en un formato legible por máquina. Por eso, aplicando Git, se puede acceder a esta funcionalidad. Los datos son sacados mediante scripts en Python, en formato XML, del portal legislativo oficial manejado por el gobierno alemán, luego transformados a Markdown y finalmente enviados en un commit al repositorio en un formato estandarizado. Se eligieron esas tecnologías por los siguientes motivos: [78]

- Git: Ya que es el sistema distribuido de control de versionado más popular en este momento.

[24] https://github.com/SFMOCI/openlaw
[25] http://www.sanfranciscocode.org/api-key/
[26] http://www.amlegal.com/library/ca/sfrancisco.shtml
[27] https://github.com/bundestag/gesetze

- GitHub: Ya que es el host de Git más popular y tiene algunas herramientas útiles como Pull Requests y GitHub Pages.

- Markdown: Las leyes no contienen semántica legible por máquina. Otros formatos como XML contienen mucha sintaxis innecesaria, dificultan la lectura y escritura por parte de las personas y hace que las comparaciones entre los archivos sean difíciles de leer. Markdown es un formato de texto intuitivo, legible y escribible por las personas sin necesidad de herramientas adicionales, lo que se adapta a las leyes, que necesitan un formateo mínimo. Además es formateable por máquina y puede ser convertido a otros formatos como HTML.

Según su creador, que ha dejado de trabajar en el proyecto, como herramienta realmente no llegó muy lejos, si bien algunos activistas digitales y el German Pirate Party realizaron algunos *pull requests* y notificaron *issues*, quizás por falta de una mejor interfaz. Igualmente tuvo cierta repercusión en artículos, sitios [79] [80] [81], y en congresos [82] [83]. Además posiblemente, si bien la idea tiene potencial, debería mejorar en algunos aspectos como la interfaz.

Código civil francés

De manera similar al caso de Alemania, el código civil de Francia está disponible en un repositorio en GitHub a partir de este año 2015,[28] por iniciativa de un desarrollador llamado Steeve Morin. La ventaja de este repositorio frente al que se encuentra como datos abiertos en una página del gobierno, [84] es que no sólo permite ver el código actual y sus estados anteriores, sino también ver fácilmente los cambios entre ellos, que incluyen toda la historia de cambios en las leyes desde el siglo XIX, marcados como *commits* en los diferentes artículos. [85] Esto se traduce en una forma más clara de ver los cambios.

```
     À l’article 165 du même code, le mot : « devant » est remplacé
par les mots : « lors d’une cérémonie républicaine par[29]
```

[28] https://github.com/steeve/france.code-civil
[29] Traducción: En el artículo 165 del código, la palabra "antes" es reemplazada por los términos "en una ceremonia republicana por".

Imagen 6.1 - Diff visto en GitHub de un artículo del código civil almacenado en el repositorio Legifrance

Este repositorio fue creado accediendo a cada artículo en Legifrance, una entidad del gobierno francés responsable de publicar textos legales online, y luego creando un archivo JSON de todos ellos, con el libro o sección al cual pertenecen y guardando para cada artículo sus versiones (fechas). Luego con un script y usando el JSON resultante, se crearon los archivos Markdown y corrieron los comandos git necesarios para guardar la información en el repositorio. Este es un proceso propenso a errores, ya que los archivos pueden tener errores de formateo. [86] Entre otros inconvenientes, por el formato utilizado por Git para las fechas, aquellas anteriores a 01/01/1970 tienen ese valor con segundos incrementales, aunque en el mensaje del commit está la fecha real.

Taiwan

Este repositorio contiene las leyes de Taiwán,[30] las cuales son obtenidas mediante scripts de una página del gobierno y transformadas a 3 archivos JSON. Un archivo que contiene la ley en sí, otro con su historia de cambios y otro con la historia de su movimiento por los diferentes comités. Además se provee una versión Markdown de cada ley. El repositorio está dividido en diferentes carpetas que contienen las leyes para un cierto tema, como leyes administrativas de educación, tráfico, etc.

Japón

Al igual que los demás repositorios, este repositorio[31] nace a partir de las leyes de un sitio gubernamental.[32] En este caso las leyes y regulaciones del ministerio de asuntos internos y comunicaciones se guardan en el repositorio de dos formas, en un archivo de texto plano sin formato y como documentos html. No posee ningún tipo de jerarquía para los archivos, lo cual dificulta la navegación por el mismo.

[30] https://github.com/victorhsieh/tw-law-corpus
[31] https://github.com/riywo/law.e-gov.go.jp
[32] https://github.com/g0v/twlaw

7. Historia del repositorio

El primer punto a definir del diseño del repositorio es cómo se va a representar la historia de la legislación municipal. Es quizás el aspecto más general y uno de los más importantes para que el sistema sea usable. Este análisis abarca la lógica por la cuál se van a crear nuevos *commits*, su forma, el tipo de historia (lineal, con *branches, etc*) y cómo se podría realizar el traspaso del sistema actual hacia un repositorio Git.

¿Qué representa cada commit?

Para el caso del repositorio principal, el objetivo de cada *commit* es el de representar la legislación municipal válida durante un período dado de tiempo. Esto abarca a todas las ordenanzas del municipio, como ya se dijo, debido al modelo de *snapshots* de Git. Ya que la legislación cambia a través del tiempo, se va a contar con varios *commits* dentro de la historia del repositorio. Va a haber uno para cada uno de estos estados, los cuales van a estar conectados entre sí de acuerdo al modelo utilizado por Git, en el cual se tiene una referencia al *commit* padre. Van a conformar una historia lineal, es decir, como sucedería si se realizaran los *commits* sobre un mismo *branch*[33], ya que el conjunto de legislación válida para un rango de tiempo siempre es uno, por lo cual se representa sobre una sola línea de desarrollo. Este es el caso más básico donde sólo se utiliza Git para versionar la legislación.

Para el caso de GitHub, donde además del repositorio oficial con la legislación se tienen otros usuarios trabajando y realizando aportes sobre sus repositorios propios, el concepto de *commit* es diferente. Por ejemplo para un concejal trabajando sobre su repositorio en un *branch* específico, cada *commit* representa una propuesta de cambios sobre la legislación municipal. Más aún, si esa propuesta se va mejorando y por lo tanto se realizan más *commits* incorporando esas mejoras, lo que quedan son los diferentes etapas por las que fue pasando esa propuesta. Así, cada *commit* por fuera del *branch* principal del repositorio oficial conforma la historia del trabajo sobre una propuesta.

Por lo tanto, para que el repositorio con la historia de la legislación pueda ser usable, debe haber una línea de *commits* que representen esa historia con sus cambios. Cada *commit* sobre el repositorio principal debe contener las ordenanzas válidas para un rango de tiempo que va entre su fecha y la del próximo *commit*. Sería ilógico y engañoso contar con un *commit* en el cual haya ordenanzas que jamás fueron válidas. Si se piensa en contar con el repositorio para permitir el acceso a esos datos solamente, esto puede ser fácilmente realizado siempre y cuando se realicen los *commits* oportunamente y de forma correcta sobre un mismo *branch*. El problema de la representación surgiría al involucrar diferentes líneas de desarrollo con el objetivo de representar el proceso legislativo. En ese caso habría que definir la forma en que se combinarán estos diferentes *branches* dentro de la historia oficial.

[33] Ver imagen 6.1

Formato de un commit

Un objeto *commit* en Git posee la siguiente información:

- *Tree:* El objeto Git que contiene las referencias para todo el directorio de ese *commit*. De poco interés para el uso habitual, más que nada es usado internamente por Git.

- *Parent:* La referencia (sha-1) al *commit* padre.

- *Author:* El nombre, email y fecha del usuario que originalmente creó ese *commit*.

- *Committer:* El nombre, email y fecha del último usuario que aplicó ese commit. Puede llegar a ser diferente del *Author*, por ejemplo en el caso de un *rebase*.

- *Commit message:* El mensaje del *commit.*

De toda esta información, las más importantes para nuestro análisis son las fechas y el mensaje. El autor para el caso del repositorio principal no va a ser muy útil, ya que va a corresponder a la institución encargada de actualizar la historia, el digesto por ejemplo. En los casos que variaría serían en los repositorios de los concejales, donde puede haber *commits* de ellos o de otras personas, o en el caso de elegir representar el proceso legislativo en el repositorio principal, lo cual está fuera de este análisis.

Puede utilizarse la fecha del *commit* para contener la información de validez de las ordenanzas, lo que posibilita fácilmente realizar búsquedas temporales. Git ya provee esta funcionalidad mediante `git log --since` o `--after` y `git log --until` o `--before`, pudiendo especificar fechas del estilo `1 week ago` o `1/1/2010`. Se utiliza la fecha del *commiter* para estos comandos. [43] Ya que estas fechas pueden ser configuradas a la hora de realizar un *commit*, es posible representar la historia de la legislación sin la necesidad de realizarlo en el momento adecuado. Esto puede lograrse al momento de realizar el *commit* agregando `GIT_COMMITTER_DATE="Wed Feb 10 10:00 2005 +0100"` o `GIT_AUTHOR_DATE="Fri Jan 11 22:59 2006 -0200"`. Para el caso que se quiera sólo indicar el author-date, también se puede usar el comando `--date="Fri Jan 11 22:59 2006 -0200"`.

El mayor problema para nuestro caso, es que actualmente sólo permiten representar fechas a partir de 1970 por el formato usado internamente, Unix *time*, que mide el tiempo como el número de segundos desde el Unix *epoch* (1970-01-01). Si bien para la mayoría de los casos puede bastar, esto limita las posibilidades de representación de legislación anterior a esa fecha. Mientras Git siga utilizando este formato se pueden pensar estas alternativas a tener toda la información de las fechas dentro de los *commits*:

- Representar toda la historia de la legislación a partir de 1970, modelando las fechas dentro de cada *commit*. El *commit* inicial va a contener la legislación en el estado en que se encontraba hasta 1970.

- Representar toda la historia de la legislación, pero para aquellas fechas anteriores a 1970, codificar la fecha u *offset* con respecto a 1970 como parte de los demás dígitos no utilizados de la misma.

- Incluir la información de la fecha dentro del mensaje del *commit*. Esto permite representar toda la legislación, definiendo las fechas de manera homogénea. Por otro lado, no se puede hacer uso de las funcionalidades ya provistas por Git que hacen uso de las fechas de los *commits.*

El mejor enfoque bajo estas circunstancias parece ser una mezcla de las dos últimas, similar a lo hecho en el repositorio del código civil francés. Esto involucraría incluir las fechas dentro de los mensajes de los *commits* y para aquella legislación posterior a 1970 también configurarlos dentro del *commit* para que las fechas coincidan. Además para aquellas ordenanzas anteriores a 1970, para poder conservar el orden entre *commits*, se pueden asignar segundos incrementales, u alguna forma de codificación de las fechas dentro de los dígitos no usados para marcar la validez de una ordenanza. Así, se puede contar con la historia completa de la legislación, realizar consultas fácilmente para la historia más reciente, y definir alguna forma de realizar búsquedas mediante la información de las fechas o los mensajes de los *commits.* Esto último puede llegar a ser realizado mediante el uso del comando `git log --grep` que toma una expresión regular y la ejecuta contra los mensajes de los *commits*, o programáticamente, por ejemplo convirtiendo las fechas en los mensajes para un manejo más fácil o ya teniendo los rangos de fechas calculados para cada *commit*. Independientemente de la solución utilizada, esta funcionalidad debería ser transparente al usuario y quedar como un detalle de implementación dentro de una aplicación que sirva de interfaz.

Comúnmente el objetivo del mensaje en los *commits* es explicar y proveer un resumen de los cambios introducidos por el mismo. Para nuestro caso puede pensarse de la misma manera. Una primera opción es en cada mensaje incluir por ejemplo el nombre de las nuevas ordenanzas. Como alternativa se puede incluir el fundamento para cada ordenanza en particular. Allí se detallan los motivos que llevaron a la promulgación de esta ordenanza y se hace referencia al expediente correspondiente. Como puntos negativos se puede nombrar que esta información sólo está disponible en el boletín, no así en el digesto, por lo que para ordenanzas más viejas puede que esto no esté disponible o sea dificil de conseguir, y además esta forma de detallar los cambios es engorrosa en caso de tener un *commit* con más de una ordenanza. Ya que es información importante se puede incluir un *link* hacia un lugar donde estén disponibles. De cualquier forma, así se contaría con una justificación más entendible de los cambios que se introdujeron, a comparación de un *diff* entre dos *commits* sucesivos.

Migración hacia el nuevo sistema

El proceso de migración involucra dos grandes pasos, la digitalización del total de la legislación y la generación de la historia del repositorio. Ambos, por su complejidad y dependencia en tareas manuales, son propensos a errores. Por lo tanto es necesario definir un esquema en el cual estos errores se reduzcan o eliminen antes de empezar a utilizar el sistema completamente por parte de los concejales y/o como punto de obtención de *feedback.* La forma más directa y segura de hacerlo es hacerlo en dos fases que respeten los pasos antes nombrados.

La primera es la digitalización de la legislación. Esto involucra cargar los contenidos de cada ordenanza en un archivo para cada una, con el formato elegido para este propósito. Esto debería realizarse para el caso de los decretos, resoluciones y comunicaciones si se elige incluirlos también. Ya sea manualmente o de alguna manera automática es de esperar que surjan errores en esta etapa y por lo tanto alguna norma termine con su texto alterado. Por lo tanto para esta tarea puede utilizarse un repositorio en GitHub donde se vayan subiendo a medida que se cargan. Serviría como medio de coordinación y colaboración entre aquellos encargados de hacer esta digitalización y además se pueden aprovechar aportes de cualquier persona para corregir esos posibles errores. El resultado de esta tarea debería ser un repositorio con el conjunto de las ordenanzas municipales en su estado original en un formato determinado. Este es valioso para los siguientes pasos y en sí mismo, por lo que debería seguir manteniéndose luego de terminada esta digitalización inicial.

Dado el gran volumen de normas, esta tarea podría facilitarse utilizando la información guardada en el digesto online. A pesar de que no contiene la totalidad de las normas, ya que sólo contiene las vigentes, el volumen es grande y debería reducir el trabajo a realizar. [2] El problema es que las ordenanzas en él guardadas han sido modificadas para reflejar los cambios sobre las mismas, marcando en ellas las ordenanzas que lo realizaron. [101] Para este paso sería necesaria una herramienta que analice si la ordenanza fue modificada, en cuyo caso debería marcarse como modificada y evitar su procesamiento, y que en caso contrario realice la traducción entre los datos del digesto y el formato deseado. Ese formato idealmente debería identificar para esa ordenanza la fecha de entrada en validez de la misma. Tal proceso podría ser difícil de realizar, debido a que en caso de haber algunas donde no estén marcados estos cambios se perdería la consistencia del repositorio. Es por eso que habría que analizar el obtener los datos originales, o realizar la digitalización manual nuevamente de ser posible.

La segunda parte es la generación de la historia en el repositorio. Ya que en el paso anterior se identificó la fecha por la cual se van a ordenar las ordenanzas, puede llegar a automatizarse en cierto punto esta parte, seleccionando secuencialmente las ordenanzas a agregarse para cada *commit*. En cuanto al grado de granularidad de los *commits*, lo más correcto es realizar uno para todas las ordenanzas que hayan entrado en validez en el mismo día. Si bien pueden llegar a realizarse *commits* individuales, lo cuál facilitaría la identificación de los cambios y haría a cada *commit* más granular, también aumentaría en gran medida la historia del repositorio. Esto puede dificultar la navegación del mismo y además no representaría fielmente el estado real de la legislación, ya que las ordenanzas son promulgadas

en un día determinado sin distinción para todas aquellas que lo sean en el mismo día. Los pasos a seguir para realizar cada *commit* son:

1. Seleccionar las ordenanzas para la fecha correspondiente.

2. Modificar o eliminar las ordenanzas que en el texto se indique que haya que modificar o derogar.

3. Agregar los cambios al *staging area*.

4. Realizar el *commit* con su mensaje.

Idealmente, como ya se dijo, los nuevos commits en el repositorio deberían realizarse, o aparentar al menos haber sido realizados en la fecha en la cual la legislación haya entrado en vigencia. Para el caso que se esté armando la historia o que no se lo realice la actualización por nueva legislación el mismo día, puede parametrizarse al momento de realizar el commit como ya se vio.

Debido a que esta tarea de generar los *commits* con la historia es propensa a errores, tanto por los textos de las ordenanzas en sí, como por el hecho de representar cuando se produjeron los cambios en la legislación, es conveniente esperar antes de empezar a usar el repositorio. Eso no significa esperar a tener una versión estable de la historia de la legislación para publicarla en GitHub. Puede llegar a ser beneficioso para el desarrollo de la misma hacerlo de forma de aprovechar por ejemplo el reporte de *issues* sobre el texto de alguna ordenanza. Así se estaría aceptando la filosofía detrás de la plataforma desde el comienzo mismo del repositorio.

Corrección de errores

Una característica particular de Git es la forma en que los *commits* se referencian formando un árbol que representa la historia del repositorio. Además, cada uno de ellos tiene un código sha-1 que se forma a partir de sus contenidos. Ya que la referencia a los *commits* anteriores está "incluida" dentro de este código, si se cambia el código de algún *commit* anterior o de cualquier otro objeto como un *blob* o *tree*, el código de los *commits* siguientes va a ser diferente. Esto genera problemas si hay otras personas que trabajaron sobre la versión vieja, ya que a la hora de incluir ese trabajo el árbol es otro y hay que realizar un trabajo extra para solucionar esta disparidad. [43]

En el caso de que se quiera realizar una corrección sobre un *commit* antiguo a causa de un mensaje de *commit* mal hecho, un error en el texto de una ordenanza o por una decisión de cambiar el formato de los mensajes, Git provee el comando `git rebase -i` que permite realizar modificaciones sobre un rango de commits especificado, incluyendo modificar el mensaje del *commit*. Si se quiere corregir un error tipográfico en una ordenanza y que esto se vea reflejado en la historia posterior, puede utilizarse `git filter-branch` que permite

reescribir la historia aplicando filtros sobre cada *commit*. Así un cambio realizado sobre el primer *commit* se refleja en toda la historia posterior. Hay que tener en cuenta que toda la historia va a ser reescrita y cualquier trabajo hecho por otra persona en base a la historia vieja va a tener que ser aplicado sobre la nueva historia, de forma de sincronizarse con el repositorio. Esto puede realizarse mediante el comando `git rebase` sobre *master* para cada *branch* particular que tenga ese usuario.

Por estos motivos la reescritura de la historia es poco recomendada para los casos que se haya compartido esa historia. Para evitar esto es necesario proveer algún tipo de precaución como indicando que puede sufrir modificaciones y no se debería realizar trabajo sobre el mismo, y al mismo tiempo intentar realizar todas estas correcciones al inicio de la generación del repositorio y no una vez que este esté siendo usado. Si llegado el momento es conveniente reescribir la historia, puede llegar a ser necesaria alguna herramienta que facilite esta actualización para todas las personas (concejales, ciudadanos, etc.) que tengan un clon o *fork* del repositorio.

8. Organización del repositorio

Todas las operaciones que realiza Git son sobre un repositorio, los cuales están compuestos por un conjunto de carpetas y archivos. En cualquier proyecto de software se suele especificar la estructura del repositorio, de forma de tener organizados los diferentes recursos del mismo. Nuestro caso no es excepción, por lo que va a ser necesario considerar de qué forma va a organizarse el repositorio.

En esta parte de va a definir cuál va a ser la estructura interna del repositorio, en cuanto a la jerarquía de carpetas, nombres de las mismas y de los archivos con las ordenanzas.

Nombres de los archivos

Si bien puede pensarse un esquema en el cual se tenga un archivo con varias ordenanzas, lo más simple y conveniente es contar con un archivo para cada una. Así se identifica claramente donde están, y en caso de que haya modificaciones, donde hacerlas. Además cualquier información que se vea a través de Git va a ser más comprensible si se puede identificar a qué archivo y ordenanza se hace referencia.

Para esto, va a ser necesario definir una manera de nombrarlos, pudiendo basarnos en la forma utilizada por la legislación. Luego de la creación del digesto en 1936, en 1938 por la ordenanza 534 se estableció el sistema de numeración de las ordenanzas, la cual indica:

> *ARTÍCULO 1º: A partir del 1º de mayo de 1938 todas las Ordenanzas, sancionadas por el Honorable Concejo Deliberante, serán designadas con una numeración correlativa, sin distinción de años ni período alguno de tiempo.*
>
> *ARTÍCULO 2º: Para evitar confusiones con la numeración de las sancionadas en años anteriores, a partir de la fecha indicada por el artículo 1º se comenzará con la Ordenanza número quinientos (500).*

Es recomendable por lo tanto utilizar dicho número para la identificación de cada una, junto a la extensión del tipo de archivo. Ya que sólo estamos considerando el caso de las ordenanzas, es redundante agregar el prefijo "ordenanza". Si se da el caso de que se agregue otro tipo de documento como decretos o comunicaciones, si se los separa en otras carpetas no va a haber problema de identificación. Por lo tanto el nombre de un archivo con una ordenanza quedaría como `númeroDeOrdenanza.formato`.

Jerarquía de carpetas

Como se mencionó, este análisis alcanza a las ordenanzas, por lo cual las diferentes jerarquías van a considerarse para este tipo de norma solamente. En caso de incluir por

ejemplo decretos, se puede seguir un esquema similar, pero dentro de una carpeta especial para los mismos.

Por número de ordenanza

Cada ordenanza recibe un número correlativo. Una forma de organizarlas, similar a la forma usada en Bundes-Git (en ese caso es con letras ya que las leyes tienen nombres, pero es análogo), es crear carpetas que engloben ciertos rangos de ordenanzas. Esto facilita el acceso a una ordenanza en caso de conocer su valor particular, en este caso su número:

```
.
├─── 500-999
│       ├─── ordenanza 500
│       ├─── ordenanza 501
│       ...
├─── 1000-1499
│       ├─── ordenanza 1000
│       ├─── ordenanza 1001
│       ...
├─── 1500-1999
│       ├─── ordenanza 1500
│       ├─── ordenanza 1501
│       ...
```

Obviamente los rangos de números de ordenanzas de las carpetas deberían ser lo suficientemente granulares como para facilitar la búsqueda dentro de cada una, por lo que una estructura jerárquica puede ser más adecuada, de acuerdo a una cierta cantidad que se crea indicada, como 50 para este ejemplo:

```
.
├─── 500-999
│       ├─── 500
│       ...
├─── 1000-1499
│       ├─── 1000
│       ...
├─── 1500-1999
│       ├─── 1500
│       │       ├─── ordenanza 1500
│       │       ├─── ordenanza 1501
│       │       ...
│       ├─── 1550
│       │       ├─── ordenanza 1550
```

```
│       │       ├──── ordenanza 1551
│       │       ...
│       ├──── 1950
│       │       ├──── ordenanza 1950
│       │       ├──── ordenanza 1951
└──── 2000-2499
```

El problema de este enfoque es que limita el uso por parte de alguien que desconoce específicamente que ordenanza buscar, no permitiendo encontrar fácilmente ordenanzas relacionadas a un cierto tema.

Por fecha

Esta organización sería similar al enfoque por número de ordenanza, pero en vez de mostrar rangos de números, serían por fecha de entrada en validez, por ejemplo por años:

```
.
├──── 1936
│       ├──── ordenanza 500
│       ...
├──── 1937
│       ├──── ordenanza 574
│       ...
├──── ...
└──── 2015
        ├──── ordenanza 11238
        └──── ordenanza 11237
```

Al igual que con el rango de números, puede pensarse en una estructura jerárquica donde se subdivida en meses.

Esto permite localizar fácilmente una ordenanza en caso de conocer la fecha, o buscar las ordenanzas emitidas en cierto año o mes. La ventaja de esta organización y aquella por números es que la estructura es totalmente estable. Cada vez que se inicia un año o mes nuevo se crea una carpeta y se van agregando las ordenanzas. Una vez finalizado ese período esa carpeta va a contener las mismas ordenanzas indefinidamente.

Por tema

Como alternativa a la organización por número o fecha se podría pensar en una agrupación por tema, similar a la utilizada en el repositorio con las leyes de Taiwán y a la actual en el digesto online. En el repositorio las ordenanzas están incluidas dentro de alguna categoría particular, como puede ser "Antenas" o "Transporte", no habiendo ninguna división

más allá de estas. Actualmente hay 287 temas por los cuales buscar dentro del digesto, que varían en la cantidad de ordenanzas que contienen. Hay muchos con pocas ordenanzas, o ninguna, como en el caso del tema "Zona no nuclear". Esto indica que las categorías tienden a ser específicas y por lo tanto contienen un número reducido de ordenanzas cada una, aunque hay casos como el de "Dominio municipal - Restricciones" en el que el número de ordenanzas aumenta a 33. Esto también se puede corroborar viendo que en el caso de transporte se encuentran los siguientes temas:

- TRANSPORTE
- TRANSPORTE - BOLETO ESCOLAR, SECUNDARIO, UNIVERSITARIO
- TRANSPORTE - CUADRO TARIFARIO
- TRANSPORTE - EFLUENTES / RESIDUOS ORGÁNICOS, ETC
- TRANSPORTE - ESCOLAR
- TRANSPORTE - MENORES
- TRANSPORTE - PARADAS
- TRANSPORTE - SUSTANCIAS ALIMENTICIAS
- TRANSPORTE DE PASAJEROS (SUT)

Lo que se podría ver como una subcategorización del tema "transporte", que se repite con otros temas como "calles" o "tránsito".

Para el repositorio Git esto podría pensarse de otra manera, agregando para cada una de estas subcategorías una carpeta con todas sus ordenanzas. Estas carpetas estarían contenidas dentro de "transporte", al mismo nivel que las ordenanzas que pertenecen a ella. Esto eliminaría la necesidad del prefijo "transporte", "calles" o "tránsito" para cada subcarpeta ya que estaría especificado por la carpeta padre. Además daría una estructura más jerárquica reduciendo la cantidad de temas del nivel superior:

```
.
└─── transporte
       ├─── ordenanza de transporte 1
       ├─── ordenanza de transporte 2
       ├─── ordenanza de transporte 3
       ├─── ...
       ├─── boleto escolar, secundario, universitario
       │      ├─── ordenanza de boleto escolar, secundario, universitario 1
```

```
│       ├─── ordenanza de boleto escolar, secundario, universitario 2
│       └─── ...
├─── cuadro tarifario
├─── de pasajeros (SUT)
├─── efluentes, residuos organicos, etc
├─── escolar
├─── menores
├─── paradas
└─── sustancias alimenticias
```

Idealmente se debería encontrar una categoría para las ordenanzas genéricas de “transporte”, de forma de tener una organización más estandarizada.

A diferencia de los demás enfoques antes mencionados, es muy probable que surjan momentos en los que haya que crear temas nuevos, quizás debiendo mover ordenanzas ya existentes. En cuanto al control de versionado, agregar un tema nuevo para nuevas ordenanzas no supone ningún problema, pero en el caso que surja la necesidad de agregar una subcategoría que contenga ordenanzas anteriores, y por ende halla que moverlas de sus ubicaciones actuales ¿Que pasaría con toda la historia de las mismas? Como ya se mencionó, Git no registra archivos sino contenido. Esto significa que es lo suficientemente inteligente como para manejar renombres o movimientos de archivos o contenidos y poder conectar la historia de un contenido cuando se necesite consultar esa información. Todo esto lo realiza sin manejar el concepto de renombrado internamente. [87] Para poder ver la historia de un archivo incluyendo cualquier tipo de renombre que pudo haber sufrido, Git provee un parámetro para el comando `git log` que provee esta funcionalidad: `git log --follow` *`filename`*.

9. Formato de los archivos

Cada ordenanza va a estar contenida dentro de un archivo individual. Dentro del mismo, sus contenidos van a estar definidos y estructurados de una manera particular, la cual va a estar dada por el formato que se utilice.

En esta parte se van a analizar diferentes formatos posibles, primero enumerando los criterios para su preselección y los estándares usados para compararlos. Para cada uno se van a describir sus características generales, beneficios y desventajas, así como también va a darse un ejemplo de su uso sobre una ordenanza. Finalmente se va a dar una conclusión en base a lo analizado anteriormente.

Datos y estructura general de una ordenanza

El primer paso antes de definir el formato debe ser analizar y describir la estructura de las ordenanzas a guardar en los archivos. Ya que la LOM no explicita cuál debe ser la estructura de una ordenanza, [1] se van a tomar los textos de aquellas pertenecientes al partido de La Plata. La recomendación y aparentemente la convención en cuanto a la elaboración de ordenanzas en la provincia de Buenos Aires indican que estas van a contar con: un título donde se indica la fecha, número, temática y demás elementos primordiales; un preámbulo donde se incluye el visto y considerando que indican los antecedentes, el contexto y las razones que justifican el proyecto; y finalmente el articulado con su contenido, numeración, subinscripciones y posibles anexos. De acuerdo a esto y al análisis de varias de ellas a través del digesto online, se van a considerar las siguientes partes que se repiten en la mayoría. Para cada una se provee un texto de ejemplo:

Parte de la ordenanza	Ejemplo
Número de expediente	Expte. 54372
Fecha	LA PLATA, 27 de junio de 2012
Frase introductoria	El Concejo Deliberante, en su Sesión Ordinaria N° 11 celebrada en el día de la fecha, ha sancionado la siguiente:
Titulo	ORDENANZA 10908
Nombre de artículo	ARTICULO 1°.-
Texto de artículo	Créase en el ámbito municipal el programa "un nacimiento, un árbol".
Referencia a otro artículo y ordenanza dentro del artículo	La ampliación mencionada en el Artículo precedente se encuadra en el **Artículo 189º de la Ordenanza 10703**

Subdivisiones dentro del artículo	A. En cuanto a montos de acuerdo con las características de cada usuario (doméstico, comercial e industrial). B. En cuanto al número de cuotas y descuentos de pronto pago, de conformidad con diferentes de pago. Asimismo, se faculta al Departamento Ejecutivo para: C. Iniciar el cobro de cuotas con hasta DOS (2) meses de anticipación previo al inicio de obras, que será depositadas en Cuenta Especial a efectos de facilitar un avance normal del pago de la Certificación de obras.
Subdivisiones con divisiones dentro del artículo	ARTICULO 4º: Los indicadores básicos que deberá construir el Observatorio de Calidad de Vida, a los que podrán sumarse aquellos otros que se identifiquen como necesarios, serán: SALUD: 1º - Evolución de la mortalidad infantil, localizando los casos en el territorio, siguiendo ... 2º - Evolución de las enfermedades inmunoprevenibles localizando los casos en el territorio, siguiendo su evolución ... 3º - Evolución de la desnutrición infantil medida en jardines ... 4º - Evolución de los niveles de contaminación ambiental: sólida, líquida ... EDUCACION: 5º - Evolución de los niveles de merma de matrícula en cada colegio primario y secundario, en el tiempo oficial del ciclo, identificando los establecimientos con niveles críticos, su evolución ... 6º - Evolución de los niveles de repitencia en cada colegio ... 7º - Seguimiento de la demanda real y potencial ... SEGURIDAD: 8º - Evolución de los niveles de seguridad ... ...
Anexos	Cada uno con un número y un título. El texto de los anexos está compuesto por listados

Se cuenta además con los códigos municipales: de Ordenamiento Urbano y Territorial, de Uso del Espacio Público, de Edificación, de Faltas Municipal, Tributario y de Nocturnidad, que si bien son considerados como ordenanzas [2] poseen una estructura particular que se basa en los siguientes elementos:

Partes	Los códigos se dividen en partes compuestas de títulos, cada uno con un número y un nombre
Títulos	Cada título está compuesto de artículos

Vale aclarar nuevamente que este análisis no es completo ni abarcativo del total de partes posible de las ordenanzas del municipio. Puede que haya elementos no encontrados que sean particulares de alguna, o en un futuro se agregue alguna parte de otro tipo. La idea es proveer una visión general y representativa de la mayoría de las ordenanzas, de forma de

poder comparar los diferentes formatos y ver cuáles las permiten representar de una mejor manera.

Características buscadas

Como primer paso para realizar una preselección de formatos se pueden enumerar algunos requisitos que cada uno debe cumplir:

- Deben ser formatos de texto: ya que de no ser así se perdería el sentido de muchas de las funcionalidades de Git y GitHub. Esto deja afuera formatos como pdf.

- Deben ser formatos abiertos estándar: Un formato abierto estandarizado reduce los costos de reuso o adaptación.

- Deben permitir identificar las diferentes unidades semánticas de cada ordenanza, de forma de poder sacar valor de los datos fácilmente, tanto por una persona como por una máquina (*machine-readable*). Esto deja afuera utilizar archivos de texto plano sin formato.

Una vez pasado este filtro y no habiendo una opción obvia, deben definirse criterios de comparación entre las opciones para poder definir cuál reúne la mayor cantidad de características positivas y la menor cantidad de negativas, y que por lo tanto es la opción preferible. De acuerdo a las características deseables en un formato, teniendo en cuenta los usos particulares de este sistema, se pueden definir los siguientes estándares:

- Visualización clara del contenido
- Expresividad semántica
- Soporte para metadatos
- Simplicidad
- Facilidad de edición
- Reusabilidad
- Sinergia con Git y GitHub
- Extensibilidad

Comparación de los formatos posibles

De acuerdo a los requisitos y parámetros anteriores, y también tomando en cuenta los formatos utilizados en los diferentes casos de estudio enumerados, se identificaron los siguientes formatos posibles: XML, JSON, Markdown y HTML. Para cada uno se va a dar una descripción general, un ejemplo de una ordenanza utilizando ese formato, y un análisis de acuerdo a las características buscadas. En la mayoría de los casos no hay una sola forma de

representar los diferentes componentes de una ordenanza, por lo que se va a tratar de mantener una cierta lógica entre las diferentes representaciones siempre que sea posible. Para poder compararlos se va a utilizar la siguiente ordenanza que posee la mayoría de los elementos generales identificados, sobre la cual se va a aplicar cada formato:

Digesto Municipal de La Plata

Expte. 55154

LA PLATA, 10 de octubre de 2012

El Concejo Deliberante, en su Sesión Ordinaria N° 24 celebrada en el dia de la fecha, ha sancionado la siguiente:

ORDENANZA 10961

ARTICULO 1°. Autorízase al Departamento Ejecutivo a aprobar la ampliación de las instalaciones de la Asociación Cultural y Deportiva "Reconquista"-Biblioteca Joaquín V. González, Personería Jurídica Nº 1497/3, sita en la calle 40 Nº 1027 entre 15 y 16, de acuerdo con el plano que como Anexo I forma parte de la presente.

ARTICULO 2°. La ampliación mencionada en el Artículo precedente se encuadra en el Artículo189º de la Ordenanza 10703.

ARTICULO 3°. De forma.

Inicio Volver Inicio de página

Imagen 9.1 - Ordenanza 10961 en el digesto digital

Esta ordenanza cuenta con un número de expediente, la fecha de sanción, un texto introductorio, el título de la ordenanza, y sus artículos, de los cuales el segundo tiene una referencia a otra ordenanza. Además para cada una se va a necesitar guardar otros datos, específicamente las fechas de publicación y promulgación de la misma, de forma de poder realizar el ordenamiento de la legislación como se comentó previamente y poder contar con estos datos útiles posteriormente. Esto permite también analizar el soporte para metadatos de cada formato.

XML

XML (*Extensible Markup Language*) es un lenguaje de marcado estándar y libre para documentos compuestos de información estructurada. Es un subconjunto de SGML (*Standard Generalized Markup Language)*. Esta información está compuesta por marcas o etiquetas, que definen la estructura del mismo, y por contenido. A diferencia de otros lenguajes de marcado como HTML, XML no define semántica ni un conjunto de elementos predeterminados, por lo que es considerado un metalenguaje para describir lenguajes de marcado. [88] Aunque el

diseño de XML está enfocado para su uso en documentos, es también ampliamente usado para representar estructuras de datos arbitrarias como aquellas usadas en servicios web. [89]

Los documentos están compuestos de marcas y contenido. Hay 6 tipos de marcas: elementos, referencias a entidades, comentarios, instrucciones de procesamiento, secciones marcadas y declaraciones de tipo de documento (*elements*, *entity references*, *comments*, *processing instructions*, *marked sections*, y *document type declarations*). [88] Todos los documentos contienen uno o más elementos que son delimitados por *tags* de comienzo y fin (*start-tags* y *end-tags*) con el mismo nombre, o en el caso de elementos vacíos por un *tag* de elemento vacío (*empty-element-tag*). Estos son de la forma `<element>` y `</element>`. Cada uno de ellos tiene un tipo y puede tener un conjunto de atributos, compuestos por un nombre y un valor.

La especificación de XML define un documento como un texto bien formado cuando cumple con las reglas de sintaxis provistas en la especificación, como por ejemplo que los elementos estén correctamente anidados. Por otro lado, un documento válido es aquel es es bien formado y que a la vez cumple con las reglas extra de un *schema* como un DTD o XML Schema. Estos limitan el conjunto de elementos o atributos que pueden utilizarse, y cómo pueden usarse. [90]

Por su popularidad y versatilidad, varios formatos basados en XML han sido propuestos para la representación de documentos legales, como por ejemplo MetaLex. Este es un estándar abierto para documentos legislativos diseñado para ser independiente del lenguaje y la jurisdicción y con el objetivo de ser el formato de intercambio estándar. Ya sea esto con el propósito de presentación, descripción de las relaciones entre ellos, búsquedas o manejo de versiones. Provee un *framework* genérico y extensible para definir documentos legales de todo tipo, no solamente legislación. La validación de estos diferentes tipos de documentos la logra a través de varios *schemas* particulares para cada uno. De esta forma se tiene una estructura genérica que no aplica a ningún caso específico, permitiendo extender el *schema.* [91]

```
<ordenanza>
  <metadatos>
    <fecha-promulgacion>10/10/2012</fecha-promulgacion>
    <fecha-publicacion>09/11/2012</fecha-publicacion>
  </metadatos>
  <cuerpo>
    <expediente>Expte. 55154</expediente>
    <fecha-sancion>LA PLATA, 10 de octubre de 2012</fecha-sancion>
    <introduccion>El Concejo Deliberante, en su Sesión Ordinaria N° 24
celebrada en el día de la fecha,  ha sancionado la siguiente:</introduccion>
    <titulo>ORDENANZA 10961</titulo>
    <articulos>
      <articulo>
        <titulo>ARTICULO 1°.</titulo>
        <cuerpo>Autorízase al Departamento Ejecutivo a aprobar la ampliación
de las instalaciones de la Asociación Cultural y Deportiva
"Reconquista"-Biblioteca Joaquín V. González, Personería Jurídica Nº 1497/3,
```

```
sita en la calle 40 Nº 1027 entre 15 y 16, de acuerdo con el plano que como
Anexo I  forma parte de la presente.</cuerpo>
      </articulo>
      <articulo>
        <titulo>ARTICULO 2°.</titulo>
        <cuerpo> La ampliación mencionada en el Artículo precedente se
encuadra en el <referencia ord="10703" art="189">Artículo 189º de la Ordenanza
10703<referencia>.</cuerpo>
      </articulo>
      <articulo>
        <titulo>ARTICULO 3°.</titulo>
        <cuerpo>De forma.</cuerpo>
      </articulo>
    </articulos>
  </cuerpo>
</ordenanza>
```

Pros	Contras
• Se puede definir un DTD para poder validar y documentar la estructura • XML es lo suficientemente flexible para permitir identificar las diferentes estructuras semánticas. • El significado de cada elemento está explicitado en su nombre por lo que es fácil entender qué es cada parte del documento • Independiente de la presentación	• Es dificil de leer y escribir para una persona en comparación al texto plano • Su capacidad expresiva puede considerarse como de una complejidad innecesaria, o demasiado *verbose*.

JSON

JSON (*JavaScript Object Notation*) es un formato abierto estándar liviano para el almacenamiento e intercambio de datos en forma de objetos compuestos por pares atributo-valor. Es usado principalmente para transmitir datos entre un servidor y una aplicación web, como alternativa a XML, aunque también para almacenar información. Si bien es originalmente derivado del lenguaje JavaScript, JSON es un formato independiente del lenguaje. La extensión para archivos es ".json". [92]

JSON está compuesto por dos estructuras presentes de alguna u otra forma en virtualmente todos los lenguajes de programación, motivo por el cual tiene sentido utilizarlas en un formato pensado para el intercambio de datos: [93]

- Una colección desordenada de pares nombre-valor: En varios lenguajes esto es representado como un objeto, diccionario o arreglo asociativo. En JSON es

representado por pares nombre-valor (con el nombre de tipo string) separados por dos puntos y a la vez cada uno de esos pares separados por comas. Todos estos contenidos dentro de un corchete izquierdo inicial y un corchete derecho final.

- Una colección ordenada de valores. En la mayoría de los lenguajes esto es representado como un arreglo, vector o lista. En JSON es representado por valores separados por comas, dentro de corchetes.

Un valor puede ser un string contenido entre comillas dobles, un número, *true*, *false*, *null*, un objeto o un arreglo. Estas estructuras pueden estar anidadas.

```
{
  "metadatos":
  {
    "fecha-promulgacion": "10/10/2012",
    "fecha-publicacion": "09/11/2012"
  },
  "expediente": "Expte. 55154",
  "fecha": "LA PLATA, 10 de octubre de 2012",
  "introduccion": "El Concejo Deliberante, en su Sesión Ordinaria N° 24
celebrada en el día de la fecha,  ha sancionado la siguiente:",
  "titulo": "ORDENANZA 10961 ",
  "articulos": [
    {
      "titulo": "ARTICULO 1°.",
      "cuerpo": "Autorízase al Departamento Ejecutivo a aprobar la ampliación
de las instalaciones de la Asociación Cultural y Deportiva
"Reconquista"-Biblioteca Joaquín V. González, Personería Jurídica Nº 1497/3,
sita en la calle 40 Nº 1027 entre 15 y 16, de acuerdo con el plano que como
Anexo I  forma parte de la presente."
    },
    {
      "titulo": "ARTICULO 2°.",
      "cuerpo": " La ampliación mencionada en el Artículo precedente se
encuadra en el Artículo 189º de la Ordenanza 10703."
    },
    {
      "titulo": "ARTICULO 3°.",
      "cuerpo": "De forma."
    }
  ]
}
```

Pros	Contras

• Es más simple y conciso que XML • Permite un uso más directo de los datos por parte de aplicaciones web • Permite un alto grado de flexibilidad a la hora de definir la estructura • Independiente de la presentación	• Es difícil representar *links* dentro del texto de la ordenanza • No contempla la definición de un *schema* para las ordenanzas

HTML

HTML (*hyperText Markup Language*) es el lenguaje de marcado estándar para la creación de páginas web. Originalmente diseñado por Tim Berners-Lee para describir semánticamente documentos científicos, su diseño general permitió que se adapte para llegar a describir otros tipos de documentos y hasta aplicaciones.

Un documento HTML consiste en un árbol de elementos y texto. Como en XML, cada elemento se define con un tag de inicio, como `<body>`, y un tag de fin, como `</body>`. Los elementos pueden tener atributos, que controlan cómo funcionan. Se especifican dentro del tag de inicio con un nombre y un valor entre comillas, separados por un '='. Por ejemplo `<a href="ejemplo.html"></a>`, en el cual se define el atributo *href* con un valor de *ejemplo.html*. El elemento raíz de un documento HTML es el elemento `<html>`, que contiene dos elementos, `<head>` y `<body>`. [94]

```
<!DOCTYPE html>
<html>
<head>
  <meta name=”fecha-promulgacion” content=”10/10/2012”>
  <meta name=”fecha-publicacion” content=”09/11/2012”>
</head>
<body>
  <header>
    <p>Expte. 55154</p>
    <p>LA PLATA, 10 de octubre de 2012</p>
    <p>El Concejo Deliberante, en su Sesión Ordinaria N° 24 celebrada en el
día de la fecha,  ha sancionado la siguiente:</p>
  </header>
  <section>
    <h1>ORDENANZA 10961</h1>
    <ol>
    <li>
      <acticle>
        <h2>ARTICULO 1°.</h2>
```

```
        <p>Autorízase al Departamento Ejecutivo a aprobar la ampliación de las
instalaciones de la Asociación Cultural y Deportiva "Reconquista"-Biblioteca
Joaquín V. González, Personería Jurídica Nº 1497/3, sita en la calle 40 Nº
1027 entre 15 y 16, de acuerdo con el plano que como Anexo I  forma parte de
la presente.</p>
      </acticle>
    </li>
    <li>
      <acticle>
        <h2>ARTICULO 2°.</h2>
        <p>La ampliación mencionada en el Artículo precedente se encuadra en
<a href="#10703?189">el Artículo 189º de la Ordenanza 10703</a>.</p>
      </acticle>
    </li>
    <li>
      <acticle>
        <h2>ARTICULO 3°.</h2>
        <p>De forma.</p>
      </acticle>
    </li>
    </ol>
  </section>
</body>
</html>
```

Pros	Contras
• Cada ordenanza es una página HTML por lo cual se facilitaría la presentación para una aplicación web • Cada elemento ya tiene una semántica definida	• Los elementos son genéricos • Al igual que XML dificulta la lectura y escritura por parte de personas

Markdown

Markdown es una herramienta *open source* de conversión de texto a HTML, escrita por John Gruber. Permite escribir mediante un formato de texto plano fácil de leer y escribir, y luego convertir ese documento a XHTML o HTML válido. Por lo tanto Markdown es al mismo tiempo una sintaxis de formateo de texto plano y una herramienta de software escrita en Perl, que convierte texto plano formateado a HTML.

El objetivo principal de la sintaxis markdown es ser lo más legible posible. La idea es que un documento que utilice este formato debe poder ser publicable en texto plano, sin parecer que haya sido marcado con etiquetas o instrucciones de formateo. Está inspirado

principalmente por el formato de los emails. [95] En la siguiente tabla se ve un poco de la sintaxis básica con la correspondiente representación en HTML:

Markdown	HTML
Los párrafos se separan por una línea en blanco Hay que dejar dos espacios al final de una línea para hacer un salto de línea	Los párrafos se separan por una línea en blanco Hay que dejar dos espacios al final de la línea para hacer un salto de línea
# Título 1 # Título 1 # Título 1 ======	Título 1
## Título 2 ## Título 2 ## Título 2 ----------	Título 2
###### Título 6 ###### Título 6 ######	Título 6
Cursiva o _Cursiva_	Cursiva
Negrita o __Negrita__	Negrita
* Item 1 * Item 2 * Item 3 (puede usarse *,- o +)	 - Item 1 - Item 2 - Item 3
1. Item 1 2. Item 2 3. Item 3	 - Item 1 - Item 2 - Item 3
[link](http://ejemplo.com).	link

No hay actualmente un estándar claramente definido para Markdown, aparte de la implementación original, aunque hay un intento por definir un estándar para Internet media type como `text/markdown`, en la IETF. [96] Varios sitios soportan una sintaxis similar a Markdown, como por ejemplo GitHub. Allí se agrega soporte entre otras cosas para tablas o

bloques de código. Permite usarse en *issues*, comentarios, *pull requests* y obviamente archivos Markdown.

```
fecha-promulgacion: 10/10/2012
fecha-publicacion: 09/11/2012

Expte. 55154

LA PLATA, 10 de octubre de 2012

El Concejo Deliberante, en su Sesión Ordinaria N° 24  celebrada en el día de
la fecha,  ha sancionado la siguiente:

# ORDENANZA 10961

1. ARTICULO 1°. Autorízase al Departamento Ejecutivo a aprobar la ampliación
de las instalaciones de la Asociación Cultural y Deportiva
"Reconquista"-Biblioteca Joaquín V. González, Personería Jurídica Nº 1497/3,
sita en la calle 40 Nº 1027 entre 15 y 16, de acuerdo con el plano que como
Anexo I  forma parte de la presente.

2. ARTICULO 2°. La ampliación mencionada en el Artículo precedente se encuadra
en [el Artículo 189º de la Ordenanza 10703](#10703?189).

3. ARTICULO 3°. De forma.
```

Pros	Contras
• Al no saturar el texto con su sintaxis y dejar el texto muy parecido al original, es fácil de leer y escribir por una persona • Puede traducirse a HTML • Buena integración con GitHub	• Actualmente no hay un standard • Sintaxis limitada con respecto a HTML • No permite definir metadatos • Dificultad para definir listas de una manera estilísticamente correcta

Conclusiones

Los formatos acá enumerados son usados por los diferentes ejemplos de repositorios con legislación que se vieron anteriormente. Si bien esta no es una lista exclusiva, sí son formatos que mejor cumplen con las características buscadas. La elección de cada uno va a definirse a partir del uso principal que se le quiera dar al repositorio. En el caso de la solución planteada en esta tesina, siempre que se espere su uso principal a través de GitHub, al requerir la interacción continua con personas tanto para la lectura o edición, Markdown lleva la

delantera. Es relativamente fácil aprender, leer y editar, y además se integra muy bien con GitHub. Además puede convertirse fácilmente a HTML, lo que en cierta forma descarta su uso, aunque se pierde expresividad en la elaboración del documento cuando se utiliza Markdown.

Por otro lado, lenguajes como XML o JSON están más enfocados en el procesamiento por un máquina o para comunicación. Su uso podría ser posible en caso que no se esperase realizar trabajo directamente sobre GitHub, y en cambio se utilizara alguna interfaz construida sobre el mismo. Esta podría aislar al usuario de la complejidad de los lenguajes y permitirle interactuar con el documento de una manera más simple. También podrían usarse para almacenar la legislación más allá del repositorio principal, de forma de proveer un lenguaje más fácilmente manipulable para ser consumido por otras aplicaciones. Esto podría generarse automáticamente a partir de los contenidos del repositorio.

Lo bueno de cualquier alternativa es que permiten el traspaso relativamente simple de un formato a otro, de forma de que cualquier sea la elegida es factible generar un nuevo repositorio con archivos en otro formato a partir del inicial.

10. Usos de GitHub

En el desarrollo de software, un sistema de versionado como Git permite, entre otras cosas, la colaboración coordinada entre varias personas. Usualmente esta se logra a través de un servidor que hace de punto central hacia y desde donde se envían los cambios en el código. Como se vio en la parte 4, la opción más popular para realizar esto actualmente es GitHub, una plataforma de hosting de código colaborativa que provee diversas funcionalidades y herramientas.

En esta parte se van a analizar las ventajas, posibilidades y problemas que se abren con el uso de GitHub para albergar el repositorio Git con la legislación. El foco va a estar en las herramientas y facilidades que pueden ser de utilidad para nuestro caso, y en los flujos de trabajo que pueden definirse para por un lado lograr una mejor colaboración tanto entre los concejales, como entre ellos y los ciudadanos.

Punto de acceso al repositorio

El uso más simple consiste en tener un repositorio central con toda la legislación el cual es actualizado periódicamente con cualquier nueva ordenanza que sea promulgada. El *hosting* en GitHub facilita el acceso y provee una interfaz más amigable, para cualquier persona con pocos conocimientos, a la hora de inspeccionar el repositorio, que la provista nativamente a través de Git. Este repositorio podría ser accedido a través de la web de GitHub para consultar la legislación actual, pasada o el detalle de las diferencias entre dos puntos de tiempo. Este es el caso más parecido al sistema actual del digesto digital, donde el repositorio sirve como fuente de los datos de la legislación. Ya que en este caso la única función que cumple es la de servir como punto de acceso a los datos y no hay trabajo dependiente de la historia del repositorio, esta puede ser reescrita con el objetivo de corregir errores en una ordenanza o agregar nuevos *commits*. Si en cambio se sigue lo propuesto en esta tesina, deben tomarse recaudos a la hora de realizar esto como se indicó en la parte 7.

Algunas de las posibilidades que permite GitHub específicamente para este caso de uso son:

- Navegar a través del repositorio de la legislación
- Ver el contenido de un archivo con una ordenanza
- Ver el estado de la legislación en un *commit* particular
- Ver el *diff* de un *commit*, esto es, ver qué cambios ese *commit* produjo en el repositorio
- Comparar entre dos *commits* para poder ver las diferencias entre sus contenidos

- Poder ver el *blame* de una ordenanza, esto es, ver qué *commit* modificó qué parte de la misma y cuando.

Uso por parte de los concejales

Este uso de GitHub incluye el descrito anteriormente, pero incorpora además la utilización periódica de la plataforma por parte de los concejales. Si bien obviamente no hay una forma definida en la cual esto puede llevarse a cabo, puede ser útil basarse en los flujos de trabajo usuales de cualquier proyecto dentro de GitHub. Como primer punto, cada concejal debería contar con su usuario propio, a partir del cual van a poder poseer un *fork* del repositorio con la legislación municipal. En este, cada uno va a poder exponer las ideas o proyectos de nueva legislación sobre los cuales están trabajando, en un *branch* propio para cada uno de estos. A medida que se vayan desarrollando se deberían actualizar con las justificaciones necesarias, de forma de contar con la información más actual y una visión general de cómo se fueron tomando las decisiones sobre el texto. De esta forma cualquier ciudadano puede ver en qué y cómo se está trabajando, lo que aumenta la transparencia del proceso legislativo. Para poder identificar más fácilmente aquellos *forks* pertenecientes a miembros activos del concejo podrían agregarse los enlaces en la información del repositorio oficial.

Otra posibilidad que se abre de esta forma, es la colaboración entre los concejales. Actualmente hay una desinformación que impide conocer el trabajo que están realizando los demás concejales hasta el momento en que se presentan los proyectos en el concejo deliberante. [101] Al haber visibilidad de todos los proyectos, se puede posibilitar el *feedback* y facilitar la colaboración antes de la presentación del proyecto.

Feedback de los ciudadanos

No solamente se posibilita el *feedback* entre los concejales, sino que cada ciudadano puede, de diversas formas, colaborar en el desarrollo de proyectos y proponer cambios en la legislación actual. Una de estas formas, que responde a la usada tradicionalmente, es partir del repositorio de un concejal y luego de realizar los cambios, crear un *pull request* hacia el *branch* del proyecto correspondiente del concejal, para el caso de colaborar en el trabajo actual del mismo. Para el caso de proponer cambios que no están siendo considerados por algún concejal, como por ejemplo una ordenanza nueva o alguna mejora no contemplada, se contarían con dos alternativas. Al crear el *pull request* poner como destino el *master branch* o algún *branch* especial creado para recibir sugerencias. Actualmente GitHub no permite cambiar el destino de un *pull request*, [97] lo que facilitaría mucho esto, ya que al recibir uno interesante sólo habría que crear un nuevo *branch* e incorporarlo al mismo. En caso de realizarlo sobre *master*, se tendría que crear un *branch* aparte y recrear el *pull request* para que se integre a ese. Por otro lado, según este esquema no tendría sentido realizar los *pull requests* sobre el repositorio oficial ya que esos cambios no podrían introducirse en el mismo, debido a que representa la legislación vigente y el repositorio sólo contiene el *branch* principal.

Representación del proceso legislativo

De acuerdo a lo analizado sobre el repositorio, hay una cierta separación entre el repositorio oficial con la legislación y el del resto de los concejales. Siempre que se desarrolle un proyecto en GitHub, por ejemplo una ordenanza nueva, y esta se termine promulgando, va a haber un nuevo *commit* en el repositorio oficial independiente de aquél que contiene el proyecto del concejal. Un flujo de trabajo más elaborado involucraría la representación del proceso legislativo dentro del repositorio. Allí, por ejemplo, podrían haber *branches* dedicados para cada concejal o comisión, donde a medida que el proyecto vaya pasando por las diferentes etapas esto se vaya reflejando mediante *merges* entre los *branches* correspondientes. De esta forma se tendría un esquema más completo donde el repositorio pasaría a contener toda la historia del proceso legislativo. Esto no estaría exento de algunos inconvenientes. Involucraría un mayor esfuerzo tanto para armar el repositorio inicial como a la hora de actualizar la información. También claramente la información sería mayor y más compleja, lo que puede aumentar la posibilidad de errores y dificultar su solución. Además cada proyecto debería ser correctamente cargado por su autor, aunque puede llegar a ser paliado con algún mecanismo que actúe por defecto en estos casos. Ya que la aplicación de esta idea aumenta el nivel de complejidad y disminuye la factibilidad de su aplicación, al menos actualmente, este escenario queda fuera del análisis de la tesina.

Consideraciones

Moderación

Hasta ahora sólo se consideró el *feedback* desde la perspectiva más optimista de que todos los aportes van a ser realizados de una manera correcta conforme a lo definido por aquellos encargados del repositorio. En todos los sistemas que permiten a cualquier usuario generar contenido se dan casos de un uso abusivo de esta funcionalidad, que provoca contratiempos a aquellos que la utilizan correctamente y por lo tanto, requiere que se desarrollen mecanismos de moderación. Quizás la opción más útil que provee GitHub actualmente es la de bloquear a un usuario. Esto provoca que en los repositorios propios esta persona no pueda abrir *issues*, crear *forks*, enviar *pull requests*, comentar en ellos, y tampoco agregar o editar las páginas de *wiki.* [98] El problema a resolver es ver cómo esa moderación va a ser llevada a cabo, de forma de no incurrir en lo que puede ser interpretado como un tipo de censura por parte del estado. Esto es algo que no es un problema en sitios privados. [30]

Capacitación

Si se quiere que los concejales utilicen GitHub para trabajar en sus proyectos, y teniendo en cuenta que casi seguramente no cuentan con los conocimientos necesarios sobre versionado de código o de uso de una plataforma de ese tipo, va a ser necesaria algún tipo de capacitación antes de que se cumpla esto. Idealmente no sería necesario un conocimiento completo de Git, ya que gran parte de las acciones pueden ser realizadas dentro de GitHub y el resto de acciones que pueden llegar a necesitar es un conjunto limitado de todo lo que es posible hacer con Git. Igualmente hay una curva de aprendizaje pronunciada para aquellos usuarios no técnicos y, como ya se mencionó, una interfaz simplificada por sobre GitHub puede llegar a facilitar el uso. Esta sacaría la necesidad de conocer muchos detalles que pueden retrasar la utilización de la plataforma hasta conocerlos en profundidad. [75]

11. Aplicaciones de terceros

A partir de la posibilidad de descargar el repositorio completo con toda la historia de la legislación y de interactuar con la API de GitHub, se abre un conjunto de aplicaciones que pueden llegar a ser desarrolladas tanto por el gobierno como por terceros. Esto permitiría a cualquier persona agregar valor a los datos libremente y por lo tanto ir más allá de los objetivos iniciales o mejorando la experiencia con respecto a lo ya implementado.

Si bien el rango completo de aplicaciones posibles no puede enumerarse, sí hay ciertos patrones que han surgido a partir de esquemas similares de datos abiertos en otros países y algunos puntos que probablemente haya que mejorar de forma de facilitar el uso de la plataforma a cualquier persona no familiarizada con GitHub. En esta parte se van a analizar algunas de esas ideas y cómo pueden ser llevadas a cabo.

Interfaz sobre GitHub

Si bien GitHub ha mejorado en las funcionalidades que ofrece para facilitar su uso, y para aquellos en el área del desarrollo de software no ofrece muchos contratiempos para aprender a utilizarla, para alguien no acostumbrado a este tipo de herramientas los conceptos y funcionalidades pueden resultar difíciles de asimilar. [75] Es por eso que puede ser beneficioso utilizar algún tipo de interfaz web que abstraiga lo más posible aquello de GitHub innecesario para nuestro caso de uso y que sea más amigable para las personas que lo necesiten, al menos inicialmente. Esta aplicación incluiría interfaces sobre términos y funcionalidades ya existentes en GitHub, y funcionalidad nueva que sea de utilidad para el manejo de la legislación. Entre todas estas se pueden nombrar:

- Creación de cambios sobre algún documento junto al *pull request* correspondiente: Ya que la API permite el manejo de *pull request*, se puede hacer más intuitivo el flujo mediante un cambio del término, y facilitando la creación de un *branch* especial, la edición de los cambios a sugerir y la posterior creación del *pull request*. Si se facilita la edición de los archivos, como por ejemplo de la forma que lo hace Prose.io, y el *pull request* hacia el repositorio de algún concejal, se puede fomentar la participación y colaboración sobre la legislación.

- Acceso a los proyectos y actualizaciones de los concejales: Un aspecto importante es el de poder ver los proyectos actuales de los concejales. Esta información en GitHub es accesible desde el listado de *forks*. Ya que cualquier persona puede realizarlos, puede ser difícil encontrar aquellos pertenecientes a los concejales. Otra opción es incluir los *links* dentro de la descripción del repositorio, pero de esta manera es difícil ver sobre cuales se realizaron cambios recientemente. Si en cambio en la aplicación se listan los diferentes concejales junto a sus últimas actualizaciones y proyectos, se puede facilitar el acceso y comprensión de esta información.

- Navegación por la historia de la legislación: Puede pensarse en algún método gráfico de presentar los diferentes *commits*, como por ejemplo una línea de tiempo, la cual permitiría seleccionar un punto en particular y explorar la legislación en ese momento. Así se abstrae al usuario de la búsqueda por *commits*, y se permite hacerlo de una manera más natural. También podrían aplicarse filtros para las fechas y proveer la posibilidad de realizar comparaciones entre 2 puntos, que internamente utilizarán los comandos `git diff` y `git blame`.

- Búsqueda avanzada: Si bien GitHub permite la búsqueda dentro del repositorio, una búsqueda sobre la legislación hace necesaria una herramienta que vaya más allá de encontrar las palabras especificadas y que permita búsquedas más complejas como buscar por fechas, por sinónimos, palabras clave, categorías, etc.

Herramientas de visualización

Más allá de poder ver los cambios sobre el texto entre dos versiones de manera tradicional, hay muchas otras formas de visualizar la historia de la legislación o una ordenanza en particular. Estas, al basarse más en imágenes o gráficos facilitan la comprensión y permiten análisis más completos al poder representar más información de una manera más intuitiva.

Un ejemplo de una forma de visualización posible se puede ver en la imagen 11.1 donde se muestra el cambio a lo largo del tiempo de la ley alemana de partidos políticos. Allí se ven sus diferentes párrafos y como fueron agregando, modificando y eliminandose los artículos de la misma. Posicionándose sobre uno de estos párrafos se puede ver una descripcion al estilo `git diff` de los cambios producidos.

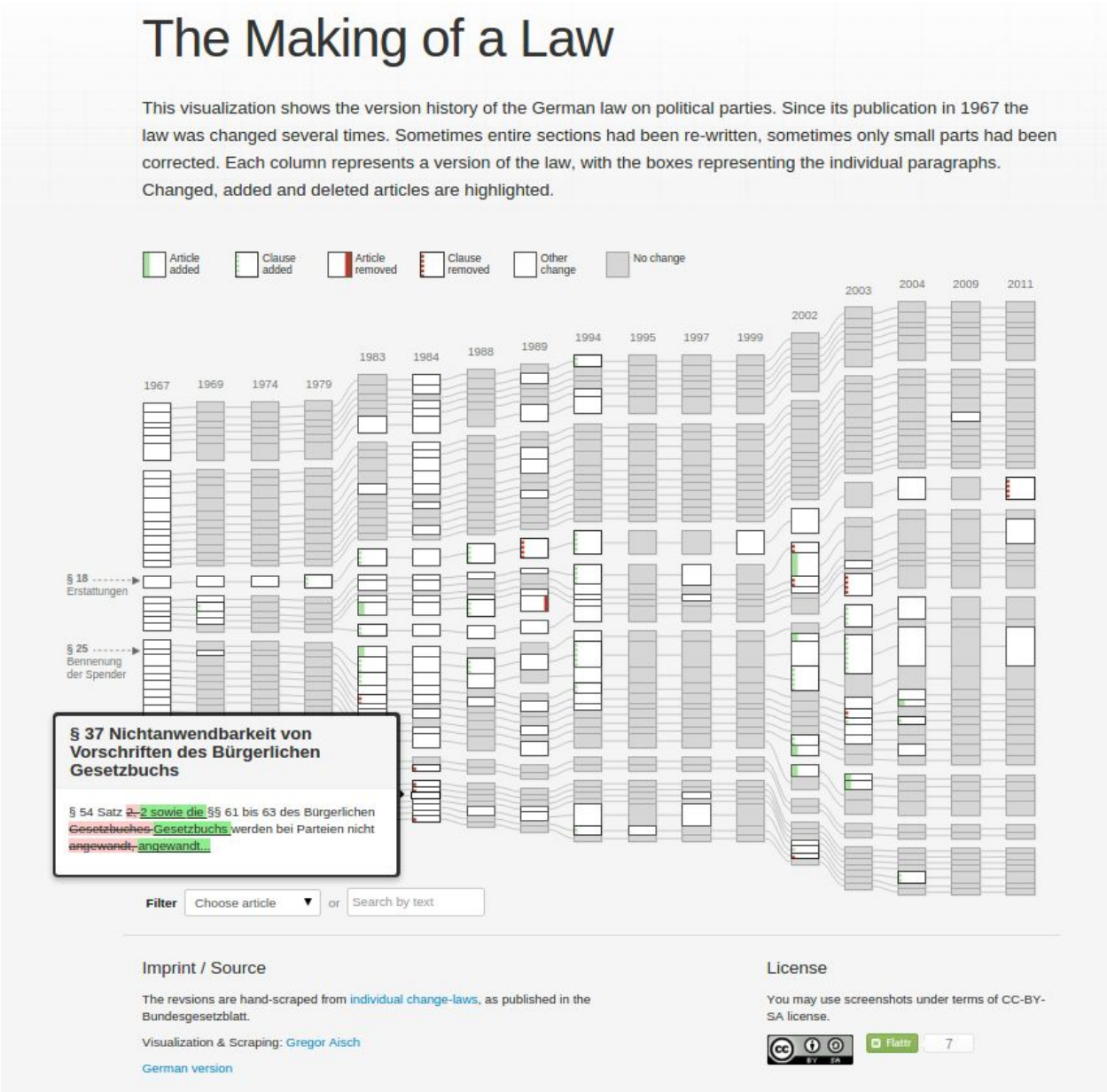

Imagen 11.1 - Visualización de la historia de la ley de partidos políticos de Alemania, realizada por Gregor Aisch. [99]

En la figura 11.2 se puede ver una visualización del código civil francés representado como una red en donde cada nodo es una ley, decreto o artículo y las aristas son las referencias entre ellas. Los datos que hicieron posible la misma se obtuvieron de la página del gobierno francés encargada de la difusión de las leyes y jurisprudencia.

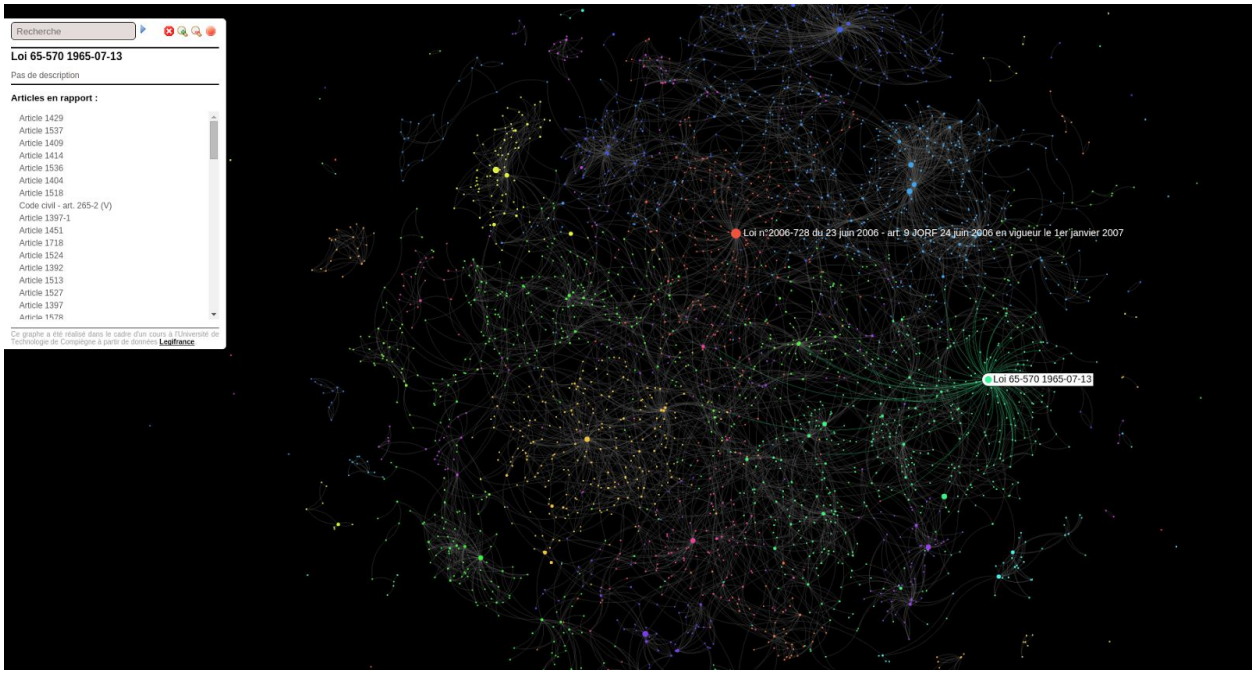

Imagen 11.2 - Visualización en forma de red del código civil francés a partir de los datos de legifrance[34], realizada por Jacques Verrier. [100]

Estos son solo ejemplos de las distintas maneras de visualizar los datos de la legislación. Lo importante es saber que son facilitados a partir del acceso a datos libres y en formatos abiertos. Por eso es de esperar que al permitir acceder a los mismos a través del repositorio se busquen mejores maneras de interactuar con sus datos en cuanto a su presentación por parte de actores privados.

[34] http://www.legifrance.gouv.fr/

12. Conclusión

Esta tesina aporta una solución para el problema del manejo de la legislación desde un enfoque alternativo y simple que hace uso de Git y GitHub, tecnologías desarrolladas para resolver un problema análogo en el desarrollo de software. Si bien se vio que hay detalles a resolver, algunos menores y otros más complejos, que dificultan la implementación o ponen en evidencia cierta disparidad entre el desarrollo de software y la legislación, en general estas tecnologías permiten su uso para este otro fin. Tomando como caso de análisis al municipio de La Plata, se vieron las posibilidades casi inmediatas que se abren con su utilización. Estas van desde la representación de la historia de la legislación municipal, la publicación de la misma y los proyectos de los concejales, la colaboración mediante aportes de cualquier persona y hasta el desarrollo de aplicaciones que hagan uso de los datos del repositorio. No se pretende definir a esta como la solución óptima y a medida para el caso de la legislación, ya que es totalmente factible el desarrollo de otro sistema más enfocado a este caso de uso que posea similares o mayores bondades y ninguno de sus defectos. La idea es poder aprovechar aquellas herramientas usadas para resolver un problema similar, que sirvan lo suficientemente bien para este caso. De esta manera se construye sobre lo ya hecho, ahorrando esfuerzo, riesgos y costos de desarrollo, y por lo tanto facilitando su implementación de forma general. Si bien estas son ventajas destacables, también se vio que en comparación a los otros sistemas, Git es extensible, performante, puede usarse libremente sin costo, y junto con GitHub, son sistemas ya desarrollados, probados, de utilización masiva y en constante mejora.

Tan importante como saber si es factible implementar esta solución, es decidir si vale la pena o es valioso hacerlo. Tomando en cuenta a los movimientos que dan un marco a lo expuesto en esta tesina, gobierno abierto, gobierno electrónico y datos abiertos, entiendo, al igual que el gran número de personas e instituciones anteriormente nombradas, que es positivo empezar a transitar ese camino de gobierno. Hace tiempo que otros gobiernos alrededor del mundo lo han tomado, con mayor o menor éxito, pero continuandolo. Actualmente hay una brecha entre estos y lo realizado en nuestro país.

La principal desventaja de esta idea es que no cuenta con un caso de uso concreto, por eso sería un paso importante poder probarla, sacar conclusiones más fundamentadas y a partir de allí ir corrigiendo aspectos que mejoren su diseño y factibilidad para su uso extendido. Análogamente al alcance elegido para esta tesina, es recomendable empezar desde el nivel municipal, tanto por las características acotadas de su legislación, en cuanto a volumen, complejidad y alcance, como por su área de influencia. Así, cualquier aspecto negativo quedaría restringido a ese municipio, pudiendo experimentar con su uso de una manera más eficiente y segura, hasta de forma replicada entre varios de ellos. Se espera que esta idea aporte, en el ámbito de la legislación, a los pilares del gobierno abierto: transparencia, participación y colaboración.

Referencias

[1] http://www.gob.gba.gov.ar/portal/subsecretarias/relacionescyc/fortalecimiento/descargas/El%20rol%20del%20concejal.pdf

[2] (02/10/2015) Entrevista a personal del digesto

[3] Baragli N, Raigorodsky N. Gómez N, Acceso a la Información en la República Argentina, Revista Probidad, edición número 23, junio de 2003

[4] Germain, C. M. (2007). Legal information management in a global and digital age: revolution and tradition. *Cornell Legal Studies Research Paper*, (07-005).

[5] Lathrop, D., & Ruma, L. (2010). *Open government: Collaboration, transparency, and participation in practice*. " O'Reilly Media, Inc.".

[6] Yu, H., & Robinson, D. G. (2012). The New Ambiguity of'Open Government'.

[7] Rogers, P. J., & Lindsey, T. D. (2012). Principles of Open Government: Transparency, Participation & Collaboration. *California Research Bureau Open Government Short Subject*.

[8] (fecha de acceso: 2015, 8 de Junio) https://opensource.com/resources/open-government

[9] Francoli, M., & Clarke, A. (2014). What's in a name? A comparison of 'open government' definitions across seven Open Government Partnership members.*JeDEM-eJournal of eDemocracy and Open Government*, *6*(3), 248-266.

*[10] House, W. (2009). Open government directive. *Memorandum for the Heads of Executive Departments and Agencies*.

[11] http://www.opengovpartnership.org/

[12] http://www.opengovstandards.org/

[13] Harrison, T. M., Guerrero, S., Burke, G. B., Cook, M., Cresswell, A., Helbig, N., ... & Pardo, T. (2011, June). Open government and e-government: Democratic challenges from a public value perspective. In *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times* (pp. 245-253). ACM.

[14] Brito, J. (2008). Hack, mash, & peer: Crowdsourcing government transparency.*The Columbia Science and Technology Law Review*, *9*, 119-157.

[15] Henry, C. L. (2003). *Freedom of Information Act*. Nova Publishers.

[16] Shkabatur, J. (2013). Transparency with (out) accountability: Open government in the United States. *Yale Law & Policy Review*, *31*(1).

[17] Piotrowski, S. J., & Van Ryzin, G. G. (2007). Citizen attitudes toward transparency in local government. *The American Review of Public Administration*, *37*(3), 306-323.

[18] Ebdon, C. (2002). Beyond the public hearing: Citizen participation in the local government budget process. *Journal of Public Budgeting Accounting and Financial Management*, *14*, 273-294.

[19] Palvia, S. C. J., & Sharma, S. S. (2007). E-government and e-governance: definitions/domain framework and status around the world. *Foundation of e-government*, 1-12.

[20] Saxena, K. B. C. (2005). Towards excellence in e-governance. *International Journal of Public Sector Management*, *18*(6), 498-513.

[21] Schnoll, H. J. (2015). *E-government: Information, technology, and transformation*. Routledge.

[22] Ndou, V. (2004). E-government for developing countries: opportunities and challenges. *The electronic journal of information systems in developing countries*, *18*.

[23] Saparniene, D. (2013). From e-government to e-governance: E-initiatives in Europe. *Siauliai University, Lithuania*

[24] Kolsaker, A., & Lee-Kelley, L. (2008). Citizens' attitudes towards e-government and e-governance: a UK study. *International Journal of Public Sector Management*, *21*(7), 723-738.
[25] Ebrahim, Z., & Irani, Z. (2005). E-government adoption: architecture and barriers. *Business Process Management Journal*, *11*(5), 589-611.
[26] Margetts, H., & Dunleavy, P. (2002). Cultural barriers to e-government. *National Audit Office, UK*.
[27] (fecha de acceso: 2015, 23 de Julio) http://opendatahandbook.org/guide/en/what-is-open-data/
[28] (fecha de acceso: 2015, 24 de Julio) http://opendefinition.org/od/
[29] Ubaldi, B. (2013). Open Government Data.
[30] Robinson, D. G., Yu, H., Zeller, W. P., & Felten, E. W. (2009). Government data and the invisible hand. *Yale Journal of Law & Technology*, *11*, 160.
[31] http://www.censo2010.indec.gov.ar/CuadrosDefinitivos/P2-D_6_441.pdf
[32] http://www.censo2010.indec.gov.ar/CuadrosDefinitivos/P8-D_6_441.pdf
[33] Goldsmith, W. W. (1999, December). Participatory budgeting in Brazil. In *Artigo do "International Meeting on*.
[34] Pussó, D., & Fessia, A. Experiencias de presupuesto participativo del Municipio de La Plata.
[35] Pagani, M. L. (2012). La participación ciudadana para la mejora de la gestión local: realidades, mitos y desafíos sobre los Presupuestos Participativos.*Cuestiones de sociología: Revista de estudios sociales*, (8), 221-244.
[36] http://posdataweb.com.ar/con-modificaciones-el-concejo-aprobo-el-presupuesto-2015/
[37] http://www.gob.gba.gov.ar/portal/subsecretarias/relacionescyc/fortalecimiento/descargas/biblioteca/manual-tecnica-legislativa.pdf
[38] http://www.gob.gba.gov.ar/portal/subsecretarias/relacionescyc/fortalecimiento/descargas/biblioteca/publicacionordenanzas.pdf
[39] http://www.elsindical.com.ar/notas/incumplimientos-de-la-ley-14-491-por-parte-de-varios-municipios/
[40] Koc, A., & Tansel, A. U. (2011). A survey of version control systems. *ICEME 2011*.
[41] Knittl-Frank, D. (2010). Analysis and comparison of distributed version control systems. *Bachelorarbeit, University of Applied Sciences, Upper Austria*.
[42] Otte, S. (2009). Version Control Systems. *Computer Systems and Telematics, Institute of Computer Science, Freie Universität, Berlin, Germany*.
[43] Chacon S. y Straub B. (2014). *Pro Git (Second edition)*. New York: Apress.
[44] The Eclipse Foundation, "Eclipse Open Source Developer Report," The Eclipse Foundation, San Francisco, CA, 2012.
[45] ZeroTurnaround, "Developer Productivity Report 2012: Java Tools, Tech, Devs & Data," ZeroTurnaround, Boston, MA, 2012.
[46] Microsoft Corporation, "Survey Results: Open Source Developer Preferences (June 2011)," Microsoft Corporation, 11 July 2011. [Online]. Available: http://blogs.msdn.com/b/codeplex/archive/2011/07/11/survey-results-open-sourcedeveloper-preferences-june-2011.aspx. [Accessed 4 February 2014].
[47] https://ianskerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/
[48] Linus Torvalds (2007-05-03). *Google tech talk: Linus Torvalds on git*. Event occurs at 02:30. Retrieved2007-05-16.
[49] Torvalds, Linus (2007-06-10). "Re: fatal: serious inflate inconsistency". *git* (Mailing list). A brief description of Git's data integrity design goals.
[50] Torvalds, Linus (2005-04-07). "Re: Kernel SCM saga..". *linux-kernel* (Mailing list). "So I'm writing some scripts to try to track things a whole lot faster."
[51] Torvalds, Linus (2005-04-08). "Re: Kernel SCM saga".*linux-kernel* (Mailing list). Retrieved 2008-02-20.

[52] Loeliger y McCullough M. (2012). *Version Control with Git (Second edition)*. Sebastopol: O'Reilly Media.
[53] Eastlake 3rd, D., & Jones, P. (2001). *US secure hash algorithm 1 (SHA1)* (No. RFC 3174).
[54] Wang, X., Yin, Y. L., & Yu, H. (2005). Collision search attacks on SHA1.
[55] http://git-scm.com/about
[56] Ylonen, T., & Lonvick, C. (2006). The secure shell (SSH) authentication protocol.
[57] Perez De Rosso S. y Jackson D. (2013). *What's Wrong with Git? A Conceptual Design Analysis*. Proceedings of the 2013 ACM international symposium on new ideas, new paradigms, and reflections on programming & software, Cambridge, MA.
[58] http://oss-watch.ac.uk/resources/gpl
[59] https://github.com/about/press 9 de julio 2015
[60] Williams, Alex (9 July 2012). "GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz". Tech Crunch. Andreessen Horowitz is investing an eye-popping $100 million into GitHub
[61] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., & Damian, D. (2014, May). The promises and perils of mining GitHub. In*Proceedings of the 11th Working Conference on Mining Software Repositories*(pp. 92-101). ACM.
[62] Zagalsky, A., Feliciano, J., Storey, M. A., Zhao, Y., & Wang, W. (2015, February). The Emergence of GitHub as a Collaborative Platform for Education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (pp. 1906-1917). ACM.
[63] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012, February). Social coding in GitHub: transparency and collaboration in an open software repository. In*Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (pp. 1277-1286). ACM.
[64] https://github.com/features
[65] Hethey, J. M. (2013). *GitLab Repository Management*. Packt Publishing Ltd.
[66] https://about.gitlab.com
[67] Open Source Initiative. (2006). The MIT License.
[68] Germain, C. M. (2010). Digitizing the World's Laws.
[69] Arnold-Moore, T. (2004). Point in time publication for legislation (xml and legislation). In *Proceedings ot the 6th Conference on Computerisation of Law via the Internet, Paris, France (December 2004)*.
[70] New Zealand Parliamentary Counsel Office (1998). Public access to legislation: A discussion paper for public comment. Wellington: NZ PCO, ¶3.1.5.
[71] Arnold-Moore, T. (2001). About time: legislation's forgotten dimension. In *Proceedings of the 3rd AustLII Law via the Internet Conference*.
[72] Grandi, F., Mandreoli, F., & Tiberio, P. (2005). Temporal modelling and management of normative documents in XML format. *Data & Knowledge Engineering*, *54*(3), 327-354.
[73] Ghosh, S. S., Klein, A., Avants, B., & Millman, K. J. (2012). Learning from open source software projects to improve scientific review. *Frontiers in computational neuroscience*, *6*.
[74] Ram, K. (2013). Git can facilitate greater reproducibility and increased transparency in science. *Source code for biology and medicine*, *8*(1), 7.
[75] Longo, J., & Kelley, T. M. Use of GitHub as a Platform for Open Collaboration on Text Documents.
[76] http://www.wired.com/2013/09/github-for-anything/
[77] http://datasf.org/about/
[78] http://okfnlabs.org/blog/2012/12/13/bundesgit-german-laws-on-github.html
[79]http://www.wired.com/wiredenterprise/2012/08/bundestag/
[80]http://www.heise.de/open/meldung/Entwicklungshistorie-von-Gesetzen-mit-Git-verfolgen-1662758.html
[81]http://www.golem.de/news/bundesgit-ein-git-repository-fuer-deutsche-gesetze-1208-93709.html

[82] http://innovation.globalintegrity.org/idea-submissions/2012/12/10/applying-version-control-to-the-legislative-process
[83] http://events.ccc.de/congress/2012/Fahrplan/events/5263.en.html
[84] https://www.data.gouv.fr/fr/datasets/legi-codes-lois-et-reglements-consolides/
[85] http://www.itworld.com/article/2904079/france-s-laws-are-now-on-github.html
[86] https://news.ycombinator.com/item?id=9295484
[87] http://marc.info/?l=git&m=119638337122721&w=2
[88] Walsh, N. (1998). A technical introduction to XML. *Available on the World Wide Web (accessed Oct 18, 2000): www. isgmlug. org*, (4), 2-49.
[89] Fennell, Philip (June 2013). "Extremes of XML". *XML London 2013*: 80–86.
[90] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1998). Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, *16*.
[91] Boer, A., Hoekstra, R., Winkels, R., Van Engers, T., & Willaert, F. (2002). Metalex: Legislation in xml. *Legal Knowledge and Information Systems. Jurix*, 1-10.
[92] Bray, T. (2014). The JavaScript Object Notation (JSON) Data Interchange Format.
[93] http://json.org/
[94] Raggett, D., Le Hors, A., & Jacobs, I. (1999). HTML 4.01 Specification. *W3C recommendation*, *24*.
[95] http://daringfireball.net/projects/markdown/
[96] http://blog.codinghorror.com/the-future-of-markdown/
[97] https://github.com/isaacs/github/issues/18
[98] https://help.github.com/articles/blocking-a-user/
[99] http://visualisiert.net/parteiengesetz/index.en.html
[100] http://www.lexmex.fr/
[101] (Octubre 2015) Entrevista a concejal
[102] (Noviembre 2015) Entrevista a encargado del digesto digital
[103] http://unpan3.un.org/egovkb/Portals/egovkb/Documents/un/2014-Survey/E-Gov_Complete_Survey-2014.pdf
[104] http://sedici.unlp.edu.ar/bitstream/handle/10915/4213/Documento_completo.pdf?sequence=2