ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO:

Grado en Ingeniería Informática (Mención Tecnologías de la Información)

Extensión de FIWARE para soporte de privacidad acorde a las nuevas reglas del RGPD relativas a la gestión del nivel de privacidad de los datos personales

Autor:
David García Matía

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO:

Grado en Ingeniería Informática (Mención Tecnologías de la Información)

Extensión de FIWARE para soporte de privacidad acorde a las nuevas reglas del RGPD relativas a la gestión del nivel de privacidad de los datos personales

Autor:
David García Matía

Tutores:
Pablo de la Fuente Redondo
Mª Mercedes Martínez González

Agradecimientos

A mis tutores por su inestimable ayuda e implicación a la hora de resolver los problemas que se han ido presentando durante la realización de este TFG.

A todos mis profesores presentes y pasados por todos los conocimientos que me han transmitido a lo largo de los años a veces con más esfuerzo por su parte del requerido.

A mi familia y amigos por todo su apoyo y comprensión en los momentos más difíciles. Sin ellos no podría haber llegado hasta aquí.

A Roberto Bahillo Ortego por su apoyo y ayuda

Resumen

El Reglamento General de Protección de Datos es una norma proveniente de Europa que redefine como deben ser tratados los datos personales de los ciudadanos europeos. Esta norma sustituye a la Ley Orgánica de Protección de Datos y por tanto las empresas deben adecuar sus sistemas a esta nueva norma. Dentro de estos sistemas a adecuar se encuentran los desarrollados en el entorno de las smart-cities, sistemas que manejan multitud de datos muchos de ellos de usuarios amparados por el RGPD. FIWARE es una plataforma abierta para la creación y desarrollo de aplicaciones para smart-cities y para el Internet of Things.

Este Trabajo Fin de Grado se ha desarrollado para discernir los cambios que deberán llevar a cabo los sistemas en producción basados en FIWARE y como se podrían implementar algunos de ellos.

Índice

Índice de figuras

Índice de cuadros

1. Introducción

1.1. Objetivos

El objetivo de este proyecto es realizar un estudio sobre el Reglamento General de Protección de Datos (en adelante RGPD), analizar las diferencias que presenta con respecto a la actual Ley Orgánica de Protección de Datos (en adelante LOPD), tanto para el usuario como para la empresa. Estudiar las posibilidades de FIWARE a la hora de construir una aplicación para smart cities acorde con el RGPD. Esto se realizará mediante la creación de un prototipo donde se atacarán ciertos aspectos de la privacidad que aparecen en el RGPD.

Como subobjetivos específicos se pueden destacar:

- Utilizar tecnologías que agilicen el desarrollo de un prototipo y den como resultado una aplicación escalable y mantenible fácilmente.
- Aplicar los conocimientos adquiridos a lo largo de la carrera para diseñar una aplicación competente.
- Realizar un desarrollo correcto que ofrezca unas garantías mínimas de seguridad, protección de datos y estabilidad de la aplicación.
- Desarrollar una aplicación que parta de un sistema legado y adecuarla al RGPD.

1.2. Problemática y necesidades a afrontar

Con la llegada del RGPD, muchas empresas han tenido que cambiar la forma en que tratan los datos, con respecto a como lo hacían con la LOPD.

Una de las cuestiones más importantes y, aunque la LOPD haya incidido en ello, el RGPD le da también mucha importancia, es la responsabilidad proactiva[10].

1.2.1. ¿Qué implica la responsabilidad proactiva?

Con el RGPD las medidas de seguridad se alejan de la Agencia Española de Protección de Datos (en adelante AEPD), para acercarse a las propias empresas. Son estas las que, siguiendo las directrices del RGPD tienen que encargarse de la protección de los datos sensibles que poseen. Este, el RGPD, provee una lista de medidas a seguir para su cumplimiento sobre:

> "protección de datos desde el diseño, protección de datos por defecto, medidas de seguridad, mantenimiento de un registro de tratamientos, realización de evaluaciones de impacto sobre la protección de datos, nombramiento de un delegado de protección de datos, notificación de violaciones de la seguridad de los datos, y promoción de códigos de conducta y esquemas de certificación."
>
> (Expansión , 9 de jun. de 2018)

Cada empresa deberá cumplir todas estas medidas para poder actuar en territorio europeo o tratar datos sobre ciudadanos europeos. En nuestro caso no vamos a tener que tratar con todas para el desarrollo del prototipo.

Se tiene muy en cuenta la protección de los datos desde el diseño y por defecto. Para el proyecto a desarrollar, las medidas de seguridad se fundamentarán en las opciones que ofrezca el *framework* utilizado, ampliándolas cuando sea necesario.

1.3. Estructura del documento

Esta memoria se estructura en diferentes capítulos.

El capítulo dos se habla sobre la planificación del proyecto y los riesgos y que pueden surgir y como serán tratados.

En el capítulo tres se explican las principales diferencias entre el RGPD y la LOPD y se muestra una tabla con dichas diferencias resumidas.

El capítulo cuatro explica que es FIWARE, su arquitectura, modelos de datos y elementos que pone a disposición de los usuarios.

El Capitulo cinco abarca todo lo relacionado con el desarrollo del prototipo análisis, diseño e implementación, para saber lo preparado que esta actualmente FIWARE para el nuevo reglamento.

El capítulo seis detalla las pruebas de funcionalidad realizadas al prototipo.

El capítulo ocho presenta las conclusiones sacadas tras la realización del proyecto y el trabajo futuro.

En los anexos se encuentran los manuales de usuario e instalación del prototipo.

2. Plan de Control

2.1. Planificación del trabajo

En la planificación inicial de este trabajo, iniciado en Octubre de 2017, se estimó su duración en unos cuatro meses, estando prevista su finalización para el mes de Febrero de 2018. Pero debido a imprevistos a la hora de encontrar la información necesaria para realizar un análisis válido de el RGPD y sobre todo de FIWARE, el desarrollo de se tuvo que posponer hasta los meses de Abril y Mayo.

Para la planificación del trabajo hemos seguido un método iterativo e incremental en el que cada una de las partes del proyecto se ha realizado en varias iteraciones tras las cuales se ha completado el análisis de nuevas características de FIWARE y el RGPD en el caso de las tareas de análisis, o se han añadido nuevas funcionalidades o mejoras a las ya existentes en el caso del desarrollo de la aplicación. No hemos optado por utilizar SCRUM debido a que es complicado plantearlo en una prueba de concepto y por lo cual nos entorpecería más que ayudarnos en la gestión del proyecto.

Nuestro proyecto consta de dos fases bien diferenciadas, la fase de análisis del RGPD y FIWARE y la fase de elaboración del sistema. Para cada una de estas fases hemos establecido las siguientes iteraciones:

- Fase de análisis del RGPD y FIWARE
  - **Estudio inicial**: En esta iteración hemos estudiado los conceptos básicos del RGPD y FIWARE.
  - **Características únicas**: En esta iteración nos hemos centrado en el análisis de los derechos recogidos en el RGPD y en el análisis de la arquitectura de FIWARE y sus *Generic Enablers*.
  - **Análisis final**: En esta última iteración hemos desarrollado la comparativa entre la LOPD y el RGPD y hemos analizado los modelos de datos utilizados por FIWARE y la implementación de los *Generic Enablers* de *Security*.
- Fase de desarrollo del sistema
  - **Inicio**: En esta iteración se ha especificado el escenario para el desarrollo del sistema así como los derechos del RGPD que queremos que asegure.
  - **Análisis**: En esta iteración hemos concretado los requisitos, casos de uso, datos y modelo de dominio del sistema.
  - **Diseño**: En esta iteración hemos definido las cuestiones referentes al diseño del sistema.
  - **Implementación de la gestión de usuarios**: En esta iteración hemos desarrollado los modelos, serializers y vistas necesarias para gestión de los diferentes tipos de usuarios.
  - **Implementación de llamadas a Orion**: En esta iteración hemos desarrollado los modelos, serializers y vistas para las llamadas a Orion.
  - **Implementación de los permisos de usuarios**: En esta iteración hemos desarrollado los políticas de acceso a las diferentes vistas para cada tipo de usuario y de filtrado para los datos provenientes de Orion.

- **Pruebas**: En esta interacción hemos realizado las pruebas de las funcionalidades de la aplicación y las modificaciones a los errores surgidos.
- **Documentación**: En esta iteración hemos desarrollado los manuales de usuario, administrador e instalación y hemos llevado a cabo la revisión de la memoria.

Los recursos disponibles para este proyecto son los siguientes:

- **Recursos humanos**: Una persona realizará el proyecto, por lo que tendrá los roles de analista, programador y *tester*.
- **Otros recursos**: un ordenador portátil para el desarrollo del documento y del prototipo.

La realización de las tareas de desarrollo se va compaginando con la documentación de la memoria, realizando las tareas de desarrollo por las mañanas y las de documentación por las tardes para así ir añadiendo las novedades regularmente.

En la Figura 1 se puede observar una estimación del tiempo total empleado en el desarrollo del proyecto.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nom recur
1		Buscar información RGPD	116 días	mié 18/10/17	mié 28/03/18		
2		Buscar información FIWARE	116 días	mié 18/10/17	mié 28/03/18		
3		Revisión del documento	73 días	jue 29/03/18	lun 09/07/18	1;2	
4		Anális del sistema	10 días	mar 24/04/18	lun 07/05/18	1FC+18 días;2FC	
5		Diseño del sistema	18 días	mar 08/05/18	jue 31/05/18	4	
6		Aprendizaje de Orion	2 días	vie 01/06/18	lun 04/06/18	4;5	
7		Análisis de los Ges de security y su posible aplicación al sistema	3 días	mar 05/06/18	jue 07/06/18	4;5;6	
8		Aprendizaje de DRF	3 días	vie 08/06/18	mar 12/06/18	7	
9		Implementación de los modelos para accounts	2 días	mié 13/06/18	jue 14/06/18	8	
10		Implementación de los serializers para accounts	3 días	vie 15/06/18	mar 19/06/18	9	
11		Implementación los viewsets para accounts	3 días	mié 20/06/18	vie 22/06/18	10	
12		Implementación de los modelos y serializers para request	2 días	lun 25/06/18	mar 26/06/18	11	
13		Implementación de las viewsets para request	7 días	mié 27/06/18	jue 05/07/18	12	
14		Testeo	2 días	vie 06/07/18	lun 09/07/18	13	

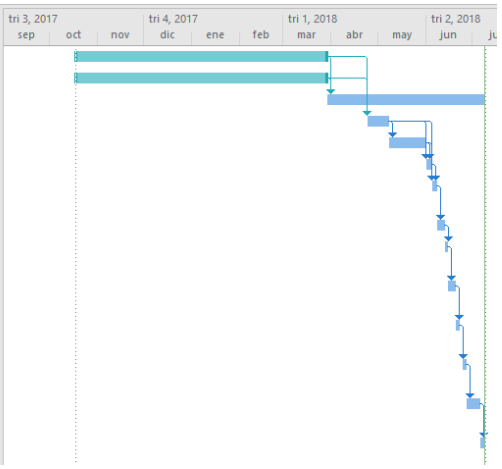

Figura 1: Planificación real del proyecto

Como se puede observar en la Figura 1, la búsqueda de información sobre el RGPD y FIWARE se realizaron de manera simultánea, tras ellas se empezó el desarrollo de la aplicación mientras que se iba completando la memoria con las partes del desarrollo realizadas.

Para la fase de inicial del proyecto correspondiente al análisis del RGPD y FIWARE y su documentación se han dedicado aproximadamente unas 400 horas, mientras que para la parte de desarrollo de la aplicación y su documentación se han dedicado aproximadamente unas 350. En total se han dedicado al proyecto 750 horas.

2.2. Plan de Riesgos

En el desarrollo de cualquier proyecto pueden producirse una serie de imprevistos que implican retrasos o incluso la cancelación del proyecto. Por eso es necesario identificar los riesgos que pueden afectar al proyecto y preparar planes de contingencia para cada uno de ellos. Para categorizar los riesgos seguiremos el ejemplo propuesto por Pressman[11], de esta forma podremos saber su repercusión en el proyecto.

Impacto / Probabilidad	Muy Bajo	Bajo	Medio	Alto	Muy Alto
Despreciable	Bajo	Bajo	Bajo	Moderado	Moderado
Marginal	Bajo	Bajo	Moderado	Moderado	Alto
Moderado	Bajo	Moderado	Moderado	Moderado	Alto
Serio	Moderado	Moderado	Moderado	Moderado	Alto
Crítico	Moderado	Alto	Alto	Alto	Alto

Cuadro 1: Repercusión de los riesgo según Probabilidad/Impacto. Fuente: Torres de Fenicia

A continuación vamos a evaluar los diferentes riesgos que pueden suceder en el proyecto y establecer planes de contingencia para ellos.

Riesgo 01	Máquina de trabajo estropeada
Impacto	Crítico
Probabilidad	Baja
Plan de Protección	Hacer copias de seguridad regularmente y tener una segunda máquina con la que trabajar.
Plan de Contingencia	Recuperar la última copia de seguridad en el otro ordenador.

Cuadro 2: Riesgo 01: Máquina de trabajo estropeada

Riesgo 02	Mala documentación de los recursos software
Impacto	Marginal
Probabilidad	Alta
Plan de Protección	Ninguno
Plan de Contingencia	Buscar alternativas a ese software o preguntar si es posible al desarrollador

Cuadro 3: Riesgo 02: Mala documentación de los recursos software

Riesgo 03	Cambios en la estructura y estilo de la página principal de FIWARE
Impacto	Marginal
Probabilidad	Alta
Plan de Protección	guardar en los marcadores del navegador las páginas principales de las secciones más visitadas.
Plan de Contingencia	Usar los marcadores a esas páginas y seguir sus enlaces.

Cuadro 4: Riesgo 03: Cambios en la estructura y estilo de la página principal de FIWARE

Riesgo 04	Enfermedad del trabajador
Impacto	Crítico
Probabilidad	Baja
Plan de Protección	Al consistir el personal de una única persona, está deberá evitar situaciones de riesgo.
Plan de Contingencia	Compensar las horas perdidas una vez recuperado.

Cuadro 5: Riesgo 04: Enfermedad del trabajador

Riesgo 05	Fallos ortográficos en la memoria
Impacto	Medio
Probabilidad	Medio
Plan de Protección	Revisiones del documento con un corrector ortográfico.
Plan de Contingencia	Revisión de la memoria por una tercera persona.

Cuadro 6: Riesgo 05: Fallos ortográficos en la memoria

3. RGPD y LOPD

3.1. Comparativa RGPD y LOPD

Los siguientes puntos que a tratar muestran los cambios que más caben destacar entre el Reglamento General de Protección de Datos y la Ley Orgánica de Protección de Datos[15].

3.1.1. Datos sensibles o especialmente protegidos

En primer lugar lo más importante es definir que se considera como **dato sensible** o **especialmente protegidos** en el RGPD, puesto que es algo que se va a tratar a lo largo de todo el documento.

Un dato se considera sensible si[7]:

- Contiene información de identidad directa (nombre, apellidos, DNI...).
- Datos anonimizados o seudonimizados, que aunque no permitan la identificación directa, si permiten individualizar comportamientos.

Estos datos, de manera más especifica, también son tratados en la LOPD. Según la Ley Orgánica de Protección de datos, los datos sensibles son[18]:

- ideología,
- religión,
- afiliación sindical,
- creencias,
- origen racial o étnico,
- salud,
- vida sexual y
- comisión o infracciones penales o administrativas

Nadie podía obligar a un usuario a revelar estos datos, salvo en las excepciones. Si los datos de ideología, religión, afiliación sindical o creencias son tratados por:

- partidos políticos,
- sindicatos,
- iglesias,
- confesiones o comunidades religiosas o
- asociaciones, fundaciones y otras entidades sin ánimo de lucro.

Si los datos de salud, vida sexual u origen racial son tratados por razones de interés general y vienen dispuestos por una ley y el consentimiento explícito del afectado.

Los datos relativos a la comisión o infracciones penales o administrativas solo podrán ser tratados por las Administraciones públicas competentes.

El Reglamento General de Protección de Datos, por su parte contempla todos estos datos como sensibles y a mayores añade otros tres más, siendo estos:

- dato genético,
- dato biométrico y
- orientación sexual.

También cambia la forma de referirse a estos datos como es el caso de las **creencias** en la LOPD que ahora pasa a llamarse **convicciones filosóficas**.

3.1.2. La extensión del ámbito territorial de aplicación

El RGPD supone una ampliación de aplicación de la normativa con respecto a la LOPD. Mientras que la LOPD solo afecta a España, el RGPD se tiene que cumplir siempre que haya datos personales de ciudadanos europeos, tanto dentro como fuera de Europa.

3.1.3. El refuerzo del consentimiento del interesado

En la LOPD se indica que es obligatorio pedir permiso a los usuarios cuando la aplicación va a recoger datos personales, esta petición puede ser efectuada de manera implícita como pasaba con las *cookies* (si continua navegando por esta página se considerara que acepta nuestro uso de las *cookies*). Esto ha cambiado con la puesta en marcha del RGPD el usuario es quien tiene que expresar de forma explicita su consentimiento de permitir recopilar los datos. En el caso de las *cookies* tiene una entrada especifica en el RGPD puesto que si no se van a recoger datos privados con ellas no es necesario pedir el consentimiento del usuario.

El LOPD entre otras cosas pide un consentimiento implícito y por escrito si se pretende recopilar datos como origen racial, salud, vida sexual... Esto se mantiene con el RGPD y a mayores establece condiciones especiales para los menores de edad (menores de 16 años reducible hasta 13) obligando que el consentimiento venga dado por su tutor legal. Con la LOPD esa edad es de 14 años, y es previsible que se mantenga.

3.1.4. Los derechos ARCO y la introducción del derecho al olvido y el derecho a la portabilidad

El RGPD reconoce los derechos de Acceso, Rectificación, Cancelación y Oposición, estos se conocen como los derechos ARCO. [20]

- derecho de acceso: El usuario tiene derecho a saber si una empresa u organización esta tratando sus datos. En caso afirmativo, este derecho también recoge que la empresa deberá determinar la finalidad, el origen de esos datos y las comunicaciones realizadas o previstas de los mismos. El plazo máximo para presentar la resolución es de 30 días, teniendo el usuario 10 días hábiles tras la comunicación de la resolución para realizar el acceso.

- derecho de rectificación: como bien indica su nombre este derecho permite al usuario modificar datos incorrectos o incompletos que una empresa u organización pueda tener. En este caso el cambio debe ser justificado con los datos referidos y su corrección pudiendo realizarse su modificación en un plazo máximo de 10 días hábiles tras la realización de la petición.

- derecho de cancelación: El usuario tiene el derecho para poder solicitar la supresión de los datos que este considere excesivos o inadecuados para el ámbito sobre el que se recogen (sea una aplicación, un estudio de población...). Esta petición deberá venir acompañada de un justificante donde se indican los campos y el motivo por el que deben suprimirse. La empresa u organización tiene un máximo de 10 días hábiles para llevarlo a cavo. La aplicación de este derecho evita el deber de bloqueo por el cual la empresa u organización evita que uses su aplicación por no ceder los datos que pide.

- derecho de oposición: La aplicación de este derecho permite al usuario oponerse al tratamiento de sus datos cuando, no sea necesario su consentimiento, se vayan a usar con finalidades publicitarias o cuyo tratamiento tenga por finalidad la adopción de una decisión que afecte al usuario.

Esto se ha visto con la llegada del RGPD y la inmensa cantidad de cambio en las políticas de privacidad de las diferentes empresas y organizaciones, como es en el caso de bancos donde estos piden datos para la personalización de publicidad y la compartición de datos del usuario con otras entidades bancarias y compañías aseguradoras.

También se recoge en el RGPD el conocido como derecho al olvido y el derecho a la portabilidad de los datos.

- Derecho al olvido: Este derecho permite al usuario obtener, sin dilación indebida, la supresión de los datos personales que le conciernan. Este derecho es una manifestación de los derechos de oposición y cancelación previamente explicados garantizados también para el entorno de internet. El principal límite que tiene este derecho es que estos datos sigan siendo importantes de acuerdo con el derecho a la libertada de expresión e información. Con respecto a la LOPD, el RGPD extiende la aplicación de este derecho de solo los buscadores a cualquier responsable de tratamiento de datos. Este derecho también viene recogido en la LOPD gracias a una sentencia del tribunal.

- Derecho a la portabilidad: Este derecho permite al interesado recibir los datos personales, que haya facilitado a la empresa u organización, en un formato estructurado, de uso común y de lectura mecánica.

En el RGPD también se incluye el derecho a que los datos del usuario no sean usados para la elaboración de perfiles, salvo en excepciones.

3.1.5. El refuerzo del deber de información

La LOPD establece que información ha de facilitarse al usuario en el momento de recoger sus datos. El RGPD refuerza esta obligación mediante la adición de nuevos datos que deben también ser comunicados. Ahora debe informase al usuario, entre otras cosas, de los datos de contacto del delegado de la empresa u organización encargado de la protección de los datos si existiera, la base jurídica del tratamiento de estos datos, los destinatarios o categorías de destinatarios, la

intención de transferir los datos personales a una tercera parte, sea esta un país u otra empresa u organización internacional...

El RGPD también detalla que esta información ha de detallarse antes de la recopilación de los datos del usuario y debe ser presentada, como máximo, en un plazo de un mes desde que se han obtenido los datos personales, o cuando se realice la primera comunicación con este (el usuario) y antes de que estos datos se envíen a otros destinatarios. Esto reduce el tiempo que daba la LOPD de tres meses.

3.1.6. La obligación de notificar los fallos de seguridad

El RGPD incluye que el responsable del tratamiento de datos debe notificar cualquier violación de la seguridad de los datos personales, tanto a la autoridad de control, en nuestro caso la Agencia Española de Protección de Datos (en adelante AEPD), como al usuario de la brecha en la seguridad. Este tiene un plazo máximo de 72 hora para notificarlo, si se superan deberá ir acompañada del los motivos de la dilación. Esta comunicación deberá describir:

> " la naturaleza de la violación de la seguridad de los datos personales, con las categorías y el número aproximado de interesados afectados, y las categorías y el número aproximado de registros de datos personales afectados; comunicar el nombre y los datos de contacto del delegado de protección de datos o de otro punto de contacto en el que pueda obtenerse más información; describir las posibles consecuencias de la violación de la seguridad de los datos personales; y describir las medidas adoptadas o propuestas por el responsable del tratamiento para poner remedio a dicha violación."
>
> (Lalanda , 6 de nov. de 2017)

Si la violación de seguridad se produce en la desde del encargado de tratamiento de los datos, esta notificación deberá efectuarse sin dilación indebida al responsable de la misma.

También se establece que el responsable deberá notificar a los usuarios, sin dilación indebida, las violaciones de seguridad cuando estas impliquen un riesgo alto para los derechos y libertades de los mismo. Esto se debe realizar en un lenguaje claro y sencillo, al igual que antes, comunicando los datos del delegado de protección de dato, describiendo posibles consecuencias, etc.

Cabe destacar que:

> "Esta comunicación no será necesaria si ha adoptado medidas de protección apropiadas y se han aplicado a los datos personales afectados, en particular las que supongan la encriptación de los datos; si ha tomado medidas ulteriores que garanticen que ya no exista la probabilidad de que se concretice el alto riesgo para los derechos y libertades del interesado; o si supone un esfuerzo desproporcionado. En este caso, se optará en su lugar por una comunicación pública o semejante por la que se informe de manera igualmente efectiva a los interesados."
>
> (Lalanda , 6 de nov. de 2017)

3.1.7. Obligaciones que se tienen que llevar a cabo en el seno de la empresa y fin de obligación de notificar ficheros

"El RGPD traslada la responsabilidad de la adecuada protección de los datos personales a las empresas que lleven a cabo los correspondientes tratamientos." (Lalanda , 6 de nov. de 2017)

Esto significa que, al contrario que con la LOPD, la AEPD ya no tiene nada que ver a la hora de como las empresas y organizaciones protegen los datos personales de los usuarios. Esto junto a que las empresas que constituidas por más de 250 personas y que traten datos de manera frecuente, o que estos se consideren de riesgo para los derechos y libertades o relativos a datos especialmente protegidos de los usuarios, deberán tener un registro de las actividades de tratamiento efectuadas bajo su responsabilidad.

En los casos en los que dicho tratamiento entrañe un alto riesgo para los derechos y libertades de los usuarios, el responsable estará obligado a realizar, previamente al tratamiento, un evaluación del impacto de las operaciones de tratamiento en la protección de los datos personales. Entre otras cosas, esta evaluación debe contener la descripción de las operaciones de tratamiento prevista, los fines, etc. con el fin de saber que riesgo entrañan estas operaciones. Si esto entraña un riesgo alto, el responsable deberá consultar a la autoridad de control antes de realizar el tratamiento.

Las empresas u organizaciones que estén a cargo del tratamiento y cuyas actividades principales consistan en operaciones que por su razón de naturaleza, alcance o fin, sean necesarias observaciones habituales de los usuarios a gran escala o consistan en tratar a gran escala datos personales o relativos a condenas e infracciones penales, deberán nombrar un Delegado de Protección de Datos. Las empresas que no cumplan estos requisitos también podrán nombrar un Delegado si así lo desean, y un grupo de empresas podrán nombrar un único Delegado para todo el grupo. Esta figura puede ser un empleado en plantilla o contratado de manera externa, siendo necesario que participe en todas las cuestiones relativas a la protección de datos y ser la figura con la que los usuarios contacten para la gestión de sus derechos. Hay que tener en cuenta que "si se produce alguna infracción, la responsabilidad seguirá recayendo en el responsable del tratamiento, no en el delegado de protección de datos." como indica Lalanda

3.1.8. Cambios sobre la legitimación de las autoridades de control

Una de las mayores diferencias con respecto a la LOPD son las autoridades de control. Mientras la LOPD estaba vigente, la encargada del control era la AEPD, sin embargo, ahora con el RGPD todos los estados miembro poseen una legislación común al respecto, y pretende asegurar una aplicación uniforme para las empresas. Por esto se establece como autoridad de control la del territorio del establecimiento principal, o del único establecimiento del responsable del tratamiento. De esta forma, cada empresa solo se somete a una única autoridad de protección de datos independientemente de en cuantos estados opere.

3.1.9. Cambios sobre la legitimación de las medidas de seguridad, y sanciones

El RGPD, al contrario que la LOPD, no ofrece una lista de medidas de seguridad, si no que solo indica que las medidas que se tomen sean apropiadas en función del riesgo. Esto no invalida el listado de la LOPD, pero si lo transforma en algo meramente orientativo para las empresas.

También se modifican las cuantías que las empresas deberán pagar por el incumplimiento de la nueva normativa. Estas dependerán del ejercicio financiero del año anterior de esta y podrá ser de entre un 2 y un 4 % de este.

3.1.10. Inscripción de ficheros en la AEPD

El RGPD ya no obliga a la inscripción de ficheros por parte de los responsables, pero si será necesario llevar un registro d actividades de tratamiento internamente.

3.1.11. Limite en el tiempo de conservación de datos

El RGPD obliga a las empresas a notificar cuando los datos recopilados van a dejar de usarse[16]. Esto viene principalmente dado por un tiempo o hasta que el usuario se de de baja del servicio o la aplicación.

Todos los datos tienen un límite máximo de tiempo, diferente para cada finalidad, salvo en ciertos casos[19]:

- si los datos tienen fines de archivo en interés público,
- finalidad de investigación científica e histórica o
- fines estadísticos

Si los datos caen en alguna de estas categorías deberán ser tratados para evitar que puedan ser identificados, como por ejemplo la anonimización.

3.2. Comparativa entre el Reglamento General de Protección de Datos (RGPD) y la Ley Orgánica de Protección de Datos (LOPD)

A continuación, se presenta una tabla donde se muestran las diferencias más importantes entre el RGPD y la LOPD.

LOPD	RGPD
Ámbito de aplicación español.	Ámbito de aplicación europeo siempre y cuando se traten datos de usuarios europeos.
Petición de consentimiento del interesado. Esta puede venir dada de manera implícita como por ejemplo en el caso de las *cookies*, donde dice que si sigues navegando en la página muestras tu consentimiento.	Refuerzo del consentimiento del interesado. Ahora el usuario tiene que expresar su consentimiento explicito para que se tomen sus datos.
Los principales derechos que recoge son los derechos de oposición y cancelación.	Se recogen los derechos ARCO (Acceso, Rectificación, Cancelación y Oposición), derecho al olvido y derecho a la portabilidad.
Las empresas no tienen la obligación de enviar pasar los datos.	Los usuarios tienen derecho a la portabilidad de los datos y deben recibirlos en un formato de fácil uso y lectura.
Cuando la empresa planea usar datos del usuario, debe pasar al interesado la información de esto en un plazo de hasta 3 meses.	Se refuerzo del deber de paso de información al interesado dando un plazo de hasta un mes tras obtenerse los datos y siempre antes de enviarse a otros destinatarios.
Las empresas no tienen la obligación de notificar los fallos de seguridad.	Las empresas deben notificar los fallos de seguridad en un plazo de hasta 72 horas o inferior si los datos son considerados de alto riesgo. A mayores las empresas deberán poner a disposición de los usuarios puntos de contacto donde se pueda obtener más información.
Datos sensibles como ideología, religión, creencias...	Se consideran los datos sensibles pertenecientes a la LOPD cambiando su nombre en algunos casos y se añaden los datos genéticos, biométricos y la orientación sexual a la lista.

Cuadro 7: Tabla comparativa entre RGPD y LOPD

4. FIWARE

4.1. Introducción

Según sus propias palabras, FIWARE es una comunidad abierta e independiente cuyos miembros están comprometidos a llevar a cabo la misión FIWARE, definida en la siguiente frase:

> "construir un ecosistema abierto y sostenible en torno a estándares para plataformas software públicas, sin derechos y orientadas a la implementación que facilitarán el desarrollo de nuevas aplicaciones inteligentes en múltiples sectores. "
>
> (FIWARE , 27 de dic. de 2017)

FIWARE fue financiada por la Unión Europea dentro del proyecto de colaboración público privada para el internet del futuro "FI-PPP" y en 2012 un consorcio europeo formado por Telefónica y Orange entre otras anunció un proyecto para desarrollar estándares de FIWARE para smart cities.

4.2. Objetivos

El principal objetivo de FIWARE es la creación de una plataforma de código libre que permita el desarrollo de aplicaciones pensadas para *smart cities*, *big data* y el Internet de las cosas de una manera fácil y estandarizada.

FIWARE se basa en los llamados *Generic Enablers* (GE) que ofrecen funciones reutilizables de propósito general en diversas categorías. Por tanto, uno de los objetivos de FIWARE, es el desarrollo de implementaciones de dichos GE para permitir a los desarrolladores su uso, modificación y extensión según las necesidades de sus aplicaciones.

4.3. Categorías de los *Generic Enablers*

Los diferentes GEs se enmarcan en una de las siguientes categorías según su funcionalidad.

- ***Cloud Hosting***: En esta categoría se enmarcan los GEs que comprenden las funcionalidades y fundamentos para el diseño de infraestructuras en la nube como por ejemplo capacidades de *hosting* a diversos niveles de abstracción de recursos.
- ***Data/Context Management***: En esta categoría se recogen los GEs que proporcionan una manera sencilla de recoger, tratar, publicar y explotar gran cantidad de información en tiempo real. Entre estos GEs destaca Orion, pilar fundamental de cualquier aplicación basada en FIWARE, que permite gestionar información de contexto, habilitando la actualización de la vista de los datos por parte del usuario en tiempo real.
- ***Internet of Things* (IoT) *Services Enablement***: En esta categoría se encuentran los GEs necesarios para el desarrollo de servicios del IoT. Típicamente una aplicación sobre el IoT se componen de una serie de dispositivos que realizan mediciones, varias *gateways* y un *backend*. Los GEs de esta categoría se dividen entre dos tipos: Los *gateway* que se encargan de realizar las conversiones entre los protocolos del los sensores y el *backend* y los GEs de *backend* que proporcionan funcionalidades de gestión de los dispositivos a las aplicaciones.

- ***Applications, Services and Data Delivery***: Los GEs de esta categoría fundamentan el mantenimiento de aplicaciones, servicios y datos en un marco empresarial a lo largo de todo su ciclo de vida.
- **Security**: En esta categoría se encuentran los GEs relacionados con la seguridad. Una de las metas de FIWARE en cuestión de seguridad es ofrecer los medios adecuados para desarrollar aplicaciones seguras desde el diseño, para lo cual ofrecen cuatro bloques de GEs: ciberseguridad, Administración de identidades y acceso, privacidad y confianza y confiabilidad.
- ***Interface to Networks and Devices* (I2ND) *Architecture***: Los GEs ubicados en esta categoría proporcionan tres conjuntos de funcionalidades, *middleware* de integración avanzado para los GEs con necesidades de comunicaciones de alto rendimiento, elementos de interconexión abierta para aplicaciones en la nube y elementos de gestión de robots por medio de componentes e interfaces a otros GEs de FIWARE.
- ***Advanced Web-based User Interface***: Los GEs de esta categoría proporcionan experiencias de usuario avanzadas usando HTML-5 y un acercamiento a interfaces de usuario basadas en web. Para esto estos GEs añaden nuevas capacidades de interacción como por ejemplo gráficos 3D e interacción inmersiva por medio de la realidad aumentada.

4.4. Arquitectura

FIWARE se basa en la utilización de módulos independientes o semi independientes entre sí llamados *Generic Enablers* (GE) para el desarrollo de aplicaciones para *smart cities* de última generación.

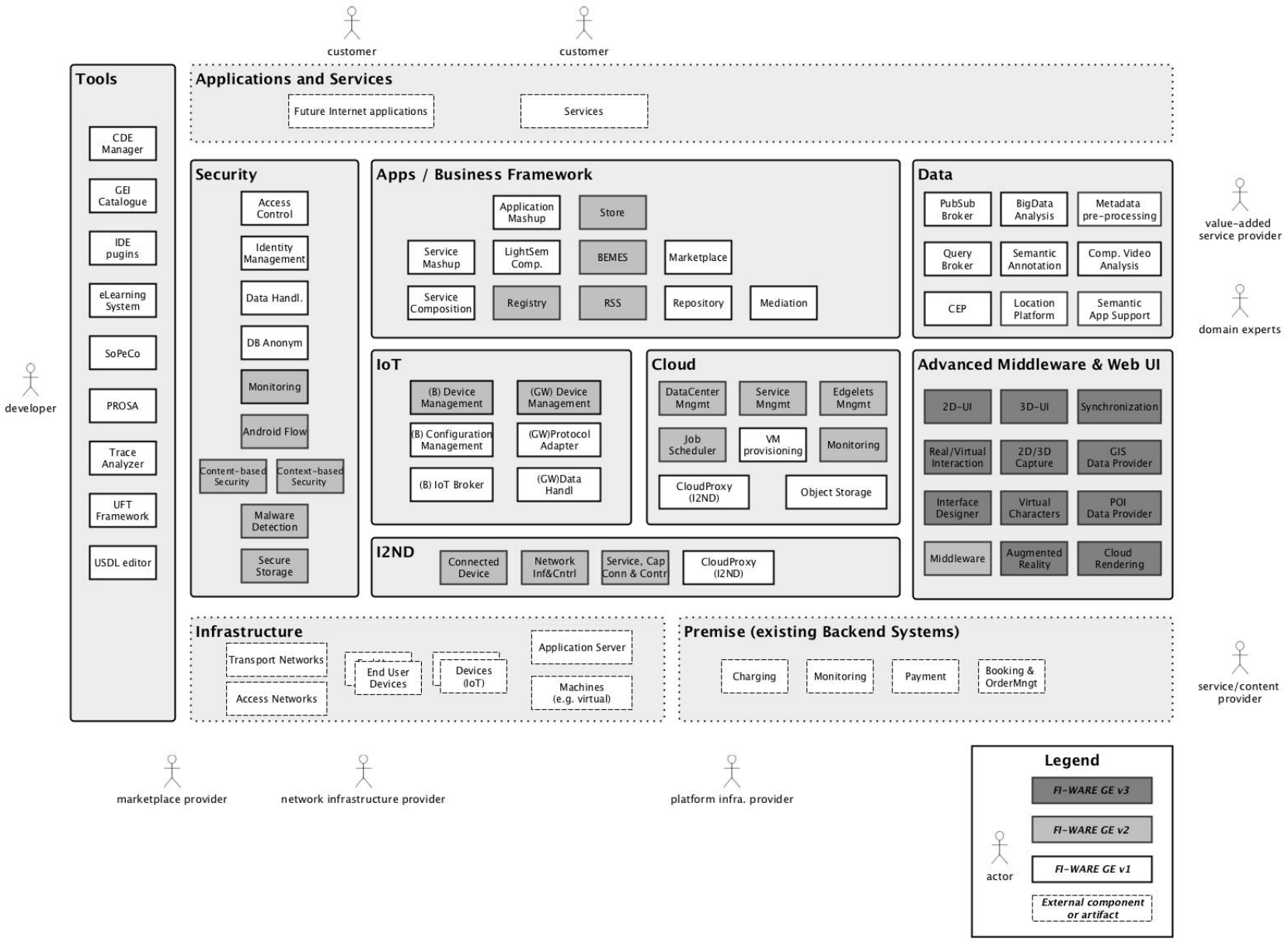

Figura 2: Arquitectura general de FIWARE. Fuente: FIWARE, *Architecture* R4

Como podemos ver en la Figura 2 los diferentes GEs están enmarcados en distintas categorías por áreas de funcionalidad. Estos GEs pueden relacionarse tanto con GEs de su propia categoría como con los de las otras, permitiendo de esta manera a los desarrolladores utilizar únicamente los GEs necesarios para su aplicación.

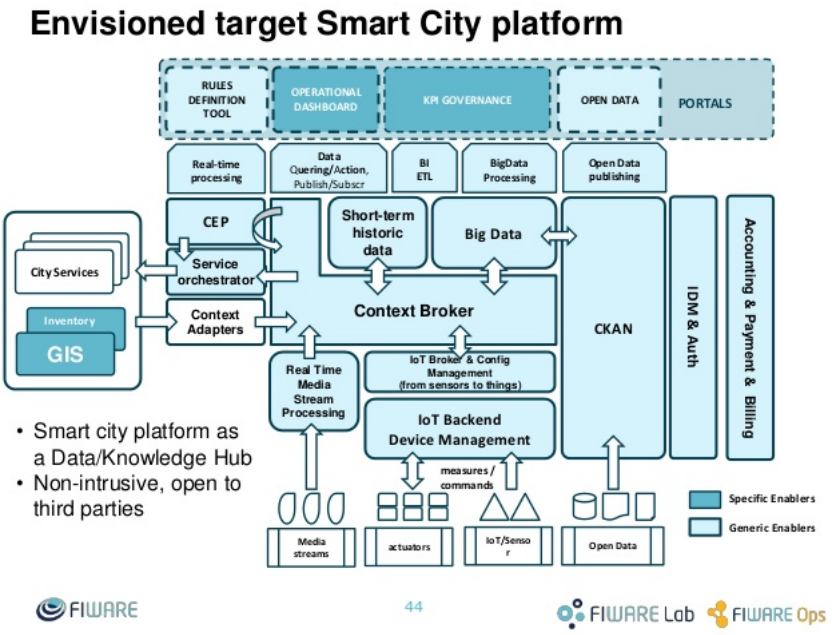

Figura 3: Objetivo previsto de una plataforma de *smart city*. Fuente: FIWARE: *an open standard platform for smart cities* [14]

La Figura 3 nos muestra la arquitectura objetivo para una aplicación basada en FIWARE. En ella se aprecia como el *Context Broker* es el eje central sobre el que se van conectando los diferentes GEs, que proporcionan funcionalidades específicas a la aplicación tales como tratamiento de *Big Data* o interconexión con elementos externos (usuario o sensores). Por otra parte vemos como los GEs sobre seguridad (IDM & Auth en la imágen), tienen una aplicación transversal, incidiendo sobre diversos módulos que forman el *core* de la aplicación. Debido a la importancia de la gestión de los datos y el papel clave de Orion en esto, para que una aplicación se la considere basada en FIWARE, debe incorporar obligatoriamente la implementación de GE *Context Broker* Orion. [14]

Para nuestro estudio es necesario saber más en profundidad la arquitectura de las categorías *Data/Context Management* y *Security*.

4.4.1. Arquitectura *Data/Context Management*

La recogida y tratamiento de los datos son dos de los elementos más importantes de cualquier aplicación y en el caso de las aplicaciones inteligentes, su importancia es capital. Es por esto que uno de los capítulos más desarrollados en FIWARE es el *Data/Context Management*. En este capítulo se recogen los GEs encargados de tratar con los datos.

En la Figura 4 se puede observar una vista general de la arquitectura de este capítulo, en ella podemos apreciar como el GE *Context Broker* es una pieza central de la arquitectura y uno de los elementos más importantes de cualquier aplicación basada en FIWARE. A este GE se van

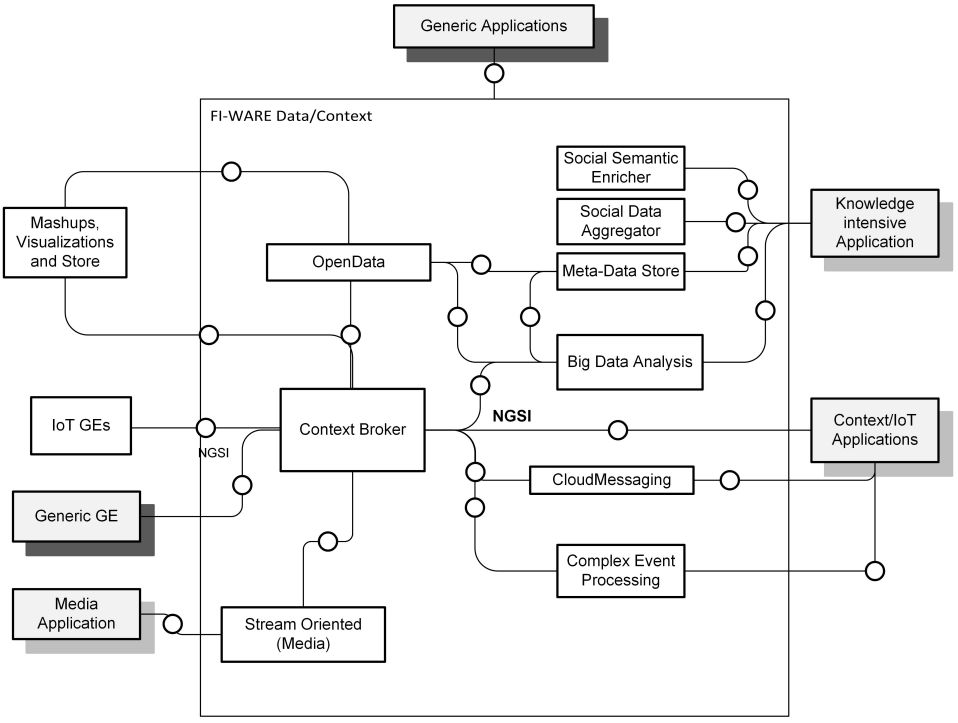

Figura 4: Fuente: FIWARE, *Architecture Context Management*

conectando distintos GEs tanto del mismo grupo como es el GE **OpenData** como de otros grupos como los GEs del grupo *Internet of Things*. De esta forma por ejemplo, los datos recibidos del GE encargado de la interfaz de sensores perteneciente al grupo *Internet of Things* pasaría los datos de múltiples sensores a *Context Broker*, el cual a su vez se los dejaría al GE *Big Data Analysis* el análisis de los mismos. De esta forma se pueden construir aplicaciones completas, complejas y especializadas usando únicamente GEs ofrecidos por FIWARE o integrar los GEs necesarios para una determinada tarea a una aplicación propia.

4.4.2. Arquitectura *Security*

La seguridad en FIWARE se fundamenta en la gestión del acceso y la identidad de los usuarios y en la ejecución del control de acceso según las políticas y permisos establecidos.

La arquitectura de seguridad de FIWARE consiste en tres GEs que trabajando en conjunto logran dotar a las aplicaciones de mecanismos para la gestión de identidad y el control del acceso de forma sencilla.

En la Figura 5 se describe las interacciones de los GEs de este capítulo tanto entre ellos como con elementos ajenos que procederemos a describir a continuación.

Un usuario que puede ser tanto una persona física como otro sistema que quiera acceder a alguno de los recursos protegidos por los GEs de seguridad deberá estar registrado en el GE *Identity Management*, encargado de la gestión de la identidad, tras lo cual se le asignarán unos permisos que se guardarán en el GE *Authorization* PDP. Cuando quiera acceder a los recursos protegidos deberá identificarse contra el GE *Identity Management* el cual devolverá un token de acceso, tras esto deberá hacer la petición de acceso al GE PEP *Proxy* encargado del control de acceso que comprobará los permisos para dicho usuario con el GE *Authorization* PDP. Si cumple

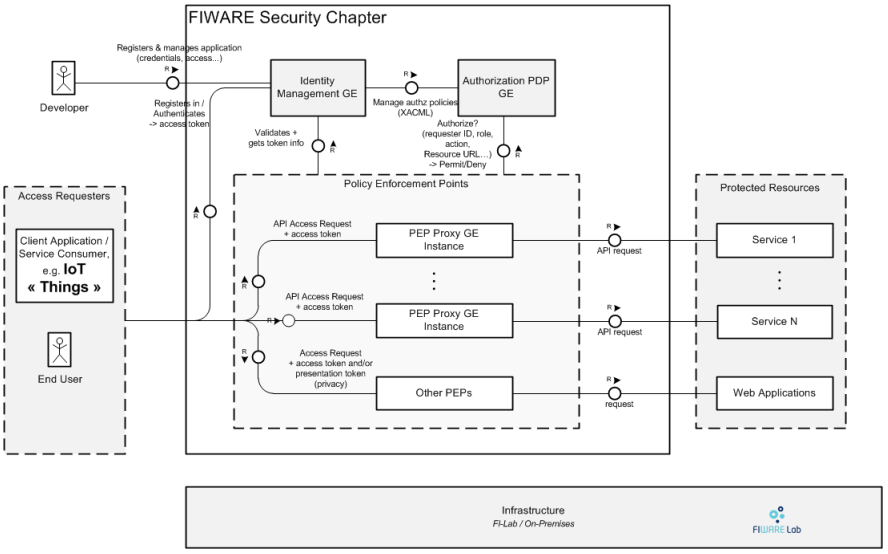

Figura 5: Fuente: FIWARE, *Architecture Security*

todos los requisitos citados, el GE PEP *Proxy* pasará la petición al recurso protegido en cuestión y devolverá la respuesta al usuario.[21]

4.5. Orion

Orion es la implementación del GE *Context Manager* enmarcado en el capítulo *Data/Context Management* y su pilar central. Permite gestionar información de contexto de una forma altamente descentralizada y a gran escala. Proporciona la API NGSIv2 de FIWARE, una API Restful cuya mayor virtud es permitir suscribirse a cambios en la información de contexto.

Para Orion y FIWARE esta información de contexto es cualquier tipo de dato que le llegue al sistema, desde información recogida por sensores hasta la introducida por usuarios es tratada de la misma manera. En la Figura 6 podemos ver como se estructura un elemento del contexto.

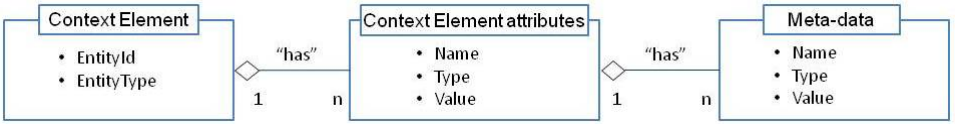

Figura 6: Modelo de la estructura de un elemento de contexto

Los elementos del contexto se componen de un id y un tipo de entidad y contienen uno o varios atributos, los cuales a su vez tienen asociados meta-datos que definen la semántica de dichos atributos.

Esta información de contexto es tratada y tras eso sirve para informar a los actores externos, permitiéndoles actuar en consecuencia. Este ciclo de acciones, capturar → tratar → actuar → capturar, es una de las necesidades básicas de las aplicaciones inteligentes y el propósito principal de Orion.

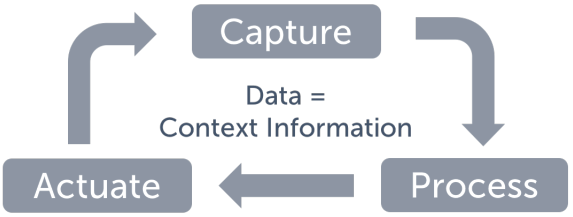

Figura 7: Ciclo de acción de los datos en FIWARE

4.6. Implementación de los *Generic Enablers* de *Security*

En la categoría *Security* de FIWARE nos encontramos únicamente con tres GE funcionales **KeyRock**, **AuthzForce** y **Wilma** los cuales trabajan en conjunto para añadir autenticación y control de acceso a las aplicaciones. Además de estos tres hay un GE llamado **Privacy** encargado de incluir privacidad en la autenticación.

- **KeyRock**[2]: Implementación del GE **Identity Management**[3] para el control de acceso. Este *enabler* lleva en desarrollo tres años y su última actualización en GitHub es del 27/06/2018 lo que nos indica que el proyecto sigue activo.
- **AuthZForce**[9]: Implementación del GE **AuthorizationPDP**[8] que permite obtener decisiones de autorización basadas en políticas de autorización, y peticiones de autorización de PEPs (*Policy Enforcement Point* - Punto de Aplicación de la Política). Lleva un año de desarrollo y su última actualización en GitHub fue en Abril de 2018
- **Wilma**[1] Implementación de la especificación del GE **PEPProxy**[4]. Gracias a este componente conjuntamente a **IdentityManagement** y **AuthorizationPDP** permite añadir seguridad en autenticación y autorización a aplicaciones *backend*. Última actualización el 27/06/2018.
- **Privacy**[17] Implementación de la especificación del GE **Privacy**[5]. Permite autenticación mientras preserva mucha privacidad. Última actualización en 2015.

Nuestro objetivo principal es aplicar el RGPD en una aplicación FIWARE por lo que debemos asegurar la privacidad de los datos. Por ello nos vamos a centrar en la especificación de FIWARE sobre **Privacy** y si podemos utilizar el GE del mismo nombre.

La especificación **Privacy** tiene como objetivo permitir el uso de P2ABC[1] a las aplicaciones que incorporen la implementación de este *enabler*.

Aunque el GE **Privacy** lleva sin actualizarse desde 2015 y según la página del catálogo de FIWARE no está terminado, según su documentación se puede instalar y usar, por lo que vamos a probar hasta que punto es utilizable y nos puede servir para adecuarlo al RGPD.[2]

[1]Privacy-Preserving Attribute-Based Credentials

[2]Tras intentar instalar **Privacy** siguiendo diferentes manuales no hemos sido capaces de ponerlo en funcionamiento, además en Mayo dejó de estar visible en el catálogo de *enablers* de FIWARE

4.7. Modelos de datos de FIWARE

Para desarrollar un módulo en FIWARE que siga el RGPD debemos analizar qué tipos de datos usan los GE y entre ellos en cuales debemos centrarnos para mantener la privacidad.

En la página de modelos de datos de FIWARE estos se encuentran organizados por categorías que procederemos a analizar a continuación, intentando destacar los elementos que en un primer vistazo nos parezcan más relevantes en cuestión de privacidad.

1. ***Alert***: Esta entidad modela una alerta y puede ser usada para enviar alertas relacionadas con retenciones de tráfico, accidentes, condiciones climatológicas, etc.

   Como su propio nombre indica esta entidad es generada a raíz de un evento por lo que no es predecible y no es un dato recurrente.

   A continuación veremos que atributos podrían necesitar de un grado de privacidad especial:

   - ***category***: Define la categoría de la alerta. Por si solo no necesita privacidad extra pero junto con otros atributos si podría necesitarla, especialmente cuando la categoría fuese "*health*" o "*security*".
   - ***subCategory***: Describe la subcategoría de la alerta. Como en el caso anterior junto con otros atributos este podría necesitar de un nivel de privacidad extra, o ser indicativo de que otro de los atributos necesita esa privacidad extra.
   - ***location***: Localización del lugar donde se ha producido la alerta. Representada por geometría **GeoSON** Obligatoria si no esta presente "*address*".
   - ***address***: Dirección física de la alerta. Obligatoria si no esta presente "*location*".

     Estos dos atributos pueden necesitar de un nivel de privacidad especial dependiendo del tipo de alerta.

2. ***Civic Issue Tracking***: En esta categoría se encuentran dos entidades para el seguimiento de problemas civiles. Estas dos entidades por temas de interoperabilidad entre FIWARE y Open311[3] (un estándar abierto y modelo cooperativo para el seguimiento de problemas civiles desarrollado en Estados Unidos) sigue una nomenclatura especial.

   La primera entidad, Open311:*ServiceType*, se encarga de definir el tipo de solicitud de servicio. Luego, esta entidad será usada por Open311:*ServiceRequest* para enviar la petición del servicio. Al ser dos entidades con muchos de sus parámetros predefinidos (por ejemplo los tipos de solicitudes, limpieza, arreglo de desperfectos, etc) solo habría que tener en cuenta los datos a rellenar por el usuario. El único atributo a tener en cuenta sería "media_url" de Open331:*ServiceRequest* ya que es un campo para urls con imágenes en las cuales pueden salir personas sin su consentimiento.

3. ***Device***: En esta categoría se encuentran dos entidades que permiten representar diferentes dispositivos (*wereables*, móviles, etc).

   La primera entidad *Device*, representa a un dispositivo en concreto. Un dispositivo está compuesto de hardware + software + *firmware* y se ha creado para cumplir una determinada tarea. El dispositivo debe ser capaz de conectarse a la red. De esta entidad debemos prestar atención a los siguientes atributos:

[3] http://www.open311.org/

- ***provider***: El proveedor del dispositivo. Según **schema.org**[4] este puede ser tanto una organización como una persona por lo que es necesario prestar atención a la privacidad de este atributo para evitar que a través de el se llegue a información sensible.
- ***owner***: el dueño o dueños del dispositivo. Este campo es una lista con referencias a personas y/u organizaciones por lo que necesitamos tener el mismo cuidado que con el apartado anterior.

La segunda entidad *DeviceModel* contiene las propiedades estáticas y comunes a múltiples instancias de *Device* como por ejemplo la marca y el modelo del dispositivo. Por lo tanto ya que esta información es común y únicamente relacionada con características de los dispositivos no es necesario aumentar el nivel de privacidad de ninguno de sus atributos.

4. ***Environment***: En esta categoría se encuentran las entidades principales para tratar con problemas medioambientales. Estas entidades son las siguientes:

   - ***AirQualityObserved***: Esta entidad se encarga de representar una observación de la calidad del aire en un lugar y momento determinados.
   - ***WaterQualityObserved***: Esta entidad se encarga de representar la calidad de una determinada masa de agua (río, lago, etc).
   - ***NoiseLevelObserved***: Esta entidad representa los niveles de ruido en un lugar y momento determinados.

   Como cada una de estas entidades trabaja con datos medioambientales de lugares públicos, ninguno de sus atributos necesita de una privacidad especial.

5. ***Indicators***: En esta categoría se encuentra la entidad *KeyPerformanceIndicator* que captura el valor y detalles asociados a un "*key performance indicator*" KPI (un tipo de medición de rendimiento para evaluar el éxito de una organización o de una actividad en particular). Entre sus atributos debemos prestar atención a :

   - ***organization***: Organización objetivo del KPI.
   - ***provider***: El proveedor del producto o servicio a evaluar. Este puede ser tanto una organización como una persona por lo que es necesario prestar atención a la privacidad de este atributo para evitar que a través de el se llegue a información sensible.
   - **calculatedBy**: Organización a cargo de calcular el KPI.

6. **Parking**: En esta categoría se encuentran varias entidades relacionadas con la gestión de aparcamientos en *smart cities*. Cada una de las entidades se utilizan para distintos tipos de aparcamientos, (parkings subterráneos, agrupaciones de aparcamientos por zonas, etc.

   - ***OffStreetParking***: Para aparcamientos fuera de calle, independientes y con en entradas y salidas definidas.
     - ***provider***: Proveedor del servicio de parking. Puede ser tanto una organización como una persona por lo que es necesario prestar atención a la privacidad de este atributo para evitar que a través de él se llegue a información sensible.

[4] https://schema.org/

Los campos restantes contienen información sobre el parking en sí (número de plazas, dimensiones máximas permitidas, precio, etc) por lo que no son datos a tener en cuenta en relación a la privacidad.

- ***OnStreetParking***: Para aparcamientos en zonas de calle al aire libre con acceso directo desde una carretera.

  En esta entidad no hay ningún campo a tener en cuenta en cuestión de privacidad.

- ***ParkingGroup***: Un grupo de estacionamientos. Pueden ser tanto un grupo de estacionamientos dentro de un parking mayor como aparcamientos a pie de calle. El agrupamiento se realiza por características comunes entre los estacionamientos.

  Como en el anterior no hay ningún campo a tener en cuenta en cuestión de privacidad.

- ***ParkingAccess***: Punto de acceso a un parking, normalmente fuera de calle.

  Al igual que los anteriores no necesita de medidas especiales de privacidad en ninguno de sus campos.

- ***ParkingSpot***: Estacionamiento bien delimitado para un único vehículo. El objetivo de esta entidad es tener control individual de cada uno de los estacionamientos y por tanto ninguno de sus campos requiere de medidas especiales de privacidad.

7. ***Points of Interest***: en esta categoría se encuentran varias entidades para modelar puntos de interés y tipos de entidades relacionados.

   - ***PointOfInterest***: Esta entidad contiene la descripción geográfica de un punto de interés. Como tal ninguno de sus campos necesita de una privacidad especial.
   - ***Beach***: Esta entidad contiene la descripción geográfica de una playa. Esta entidad es una ampliación del esquema descrito en "https://schema.org/Beach". Como punto de interés público ninguno de sus campos necesita de una privacidad especial.
   - ***Museum***: Esta entidad contiene una descripción geográfica armonizada de un museo. Los atributos a tener en cuenta son:
     - ***owner***: Persona u organización dueña del museo.
     - ***featuredArtist***: Una lista de los artistas expuestos.
   - ***TouristInformationCenter***: Como lugar que sirve como centro de información para turistas, este puede ser representado tanto como un *PointOfInterest* con categoría 439 como por el esquema descrito en "https://schema.org/TouristInformationCenter"

8. ***Point of Interaction***: En esta categoría se encuentran las entidades relacionadas con puntos de interacción inteligentes.

   - ***SmartPointOfInteraction***: Esta entidad define un lugar con tecnología para interactuar con usuarios con cualquier tipo de interfaz basada en proximidad.Como el área de interactividad puede estar compuesta por más de un dispositivo, esta entidad engloba un grupo de *SmartSpots*. Como tal ninguno de sus campos necesita de una privacidad especial.

- ***SmartSpot***: Los *Smart Spots* son los dispositivos que proporcionan la tecnología que permite a los usuarios el acceso a puntos inteligentes de interacción. Esta entidad contiene recursos para configurar el servicio de interacción como por ejemplo la URL de *broadcast*, el periodo entre *broadcast*, etc. Dada su naturaleza ninguno de sus campos necesita de una privacidad especial.

9. ***Street Lighting***: En esta categoría se encuentran las entidades relacionadas con la iluminación pública para hacer más seguras las calles tanto a conductores como viandantes.

   - ***Streetlight***: Esta entidad representa una instancia particular de un farola.
   - ***StreetlightGroup***: Esta entidad representa a un grupo de farolas parte del mismo circuito y controladas conjuntamente por un sistema automatizado.
   - ***StreetlightModel***: Esta entidad representa un determinado modelo de farola.
   - ***StreetlightControlCabinet***: Esta entidad representa un equipo que controla un grupo o varios de farolas.

     Como todas estas entidades están relacionadas con la iluminación pública, tanto para definir farolas, grupos de estas, modelos, etc ninguno de sus campos necesitan de un privacidad especial.

10. ***Transportation***: En esta categoría se recogen las entidades involucradas en aplicaciones para lidiar con problemas de tráfico.

    - ***TrafficFlowObserved***: Esta entidad representa la observación de las condiciones del tráfico en un momento y lugar determinados. Al solamente indicar las condiciones del tráfico sin incluir datos de los vehículos o sus conductores ninguno de sus campos requiere de una privacidad especial.
    - ***Road***: Esta entidad contiene la descripción contextual de una carretera. Esta entidad está formada por una o varias entidades de tipo *RoadSegments*. Al tratarse de la definición de una carretera ninguno de sus campos requiere de una privacidad especial.
    - ***RoadSegment***: Esta entidad contiene la descripción contextual de un segmento de carretera. Un conjunto de segmentos de carretera se usan para describir una carretera. Como en la entidad anterior ninguno de sus campos requiere de una privacidad especial.
    - ***Vehicle***: Entidad que describe un vehículo.
      - ***owner***: Dueño del vehículo tanto persona física como organización.
    - ***VehicleModel***: Esta entidad modela un modelo de vehículo en particular, incluyendo todas las características comunes a múltiples instancias de vehículos pertenecientes a dicho modelo. Como tal ninguno de sus campos requieren de una privacidad especial.

11. ***Weather***: En esta categoría se recogen entidades útiles para manejar datos climatológicos.

    - ***WeatherForecast***: Esta entidad contiene un a descripción armonizada del pronostico del tiempo para un lugar y momento determinado.
    - ***WeatherObserved***: Esta entidad representa la observación de las condiciones climatológicas para un lugar y momento determinados.

- ***WeatherAlarm***: Esta entidad modela una alerta por riesgo debido a factores climáticos peligrosos (nieve, hielo, niebla, etc).

  Ninguna de estas entidades contiene campos que necesitan de una privacidad especial.

12. ***Waste Management***: En esta categoría se encuentran las entidades involucradas en escenarios de tratamiento de residuos.

    - ***WasteContainerIsle***: Entidad que define un área en la que se encuentran uno o varios contenedores de residuos.
    - ***WasteContainerModel***: Entidad que modela las características estáticas de un determinado modelo de contenedor de residuos.
    - ***WasteContainer***: Entidad que representa a un único contenedor de residuos.

    Ya que estas entidades modelan aspectos de la gestión de residuos sin llevar a cabo un control de cuotas o de uso, ninguno de sus campos necesita de medidas especiales de privacidad.

A lo largo de estas entidades nos hemos encontrado con atributos de tipo "*Person*", "*Organization*" y/o "*Provider*". Estos tipos de datos definidos en schema.org/Person, schema.org/Organization y schema.org/Provider respectivamente, contienen campos sensibles para la privacidad que debemos tener en cuenta:

- ***Person***
  - ***additionalName***: Nombre adicional para una persona, por ejemplo segundo nombre. Sub-propiedad de *memberOf*.
  - ***address***: Dirección postal.
  - ***affiliation***: Organización o grupo al que esta afiliada una persona.
  - ***birthDate***: Fecha de nacimiento.
  - ***birthPlace***: Lugar de nacimiento.
  - ***children***: Hijo.
  - ***familyName***: Apellido. Junto con *givenName* puede sustituir al atributo *name*.
  - ***gender***: Genero.
  - ***givenName***: Nombre. Junto con *familyName* puede sustituir al atributo *name*.
  - ***height***: Altura.
  - ***memberOf***: Organización a la que pertenece la persona.
  - ***nationality***: Nacionalidad.
  - ***netWorth***: Valor financiero de la persona.
  - ***sibling***: Hermano.
  - ***taxID***: Identificador fiscal, por ejemplo el NIF en España o el TIN en EEUU.
  - ***telephone***: Número de teléfono.
  - ***weight***: Peso.

- ***image***: Imagen o fotografía de la persona.
- ***name***: Nombre completo.

- ***Organization*** En el caso de *Organization* los campos con los que debemos tener cuidado son los referentes a sus miembros, trabajadores y/o alumnos (definidos como tipo *Person*) ya que en ellos se pueden incluir los campos sensibles de *Person* mencionados anteriormente. Los campos de tipo *Person* son:
  - ***alumni***: exalumno de la organización.
  - ***employee***: Empleado.
  - ***founder***: Fundador.
  - ***funder***: Inversor.
  - ***member***: Miembro de la organización.
  - ***sponsor***: Patrocinador.
- ***Provider***: Según la página schema.org por la cual se rigen los modelos de datos de FIWARE, un campo de tipo *provider* espera datos de tipo "*Person*" u "*Organization*" por lo que debemos tratarlo de la misma manera que estos.

5. Sistema a desarrollar

5.1. Análisis

5.1.1. Introducción

A partir del 25 de Mayo de 2018 toda empresa que maneje datos sensibles de usuarios deberá tener adecuadas sus infraestructuras y políticas de protección de datos a lo dispuesto en el RGPD. Uno de los problemas principales que se encuentran las empresas a la hora de cumplir el RGPD reside en la adecuación de sus aplicaciones ya existentes a la nueva normativa. Esto es especialmente relevante en el ámbito de las smart cities ya que manejan gran cantidad de datos, siendo muchos de ellos sobre población y usuarios.

En el presente capítulo procederemos a desarrollar un módulo para una aplicación basada en FIWARE ya existente, que permita a dicha aplicación cumplir con los derechos recogidos en el RGPD.

5.1.2. Escenario

El ayuntamiento de la ciudad nos ha contratado para adecuar a el RGPD la aplicación que permite acceder a los datos sobre la población empadronada en el municipio y datos de vehículos a aplicaciones de las empresas concesionarias del ayuntamiento y a las propias del ayuntamiento que se desarrollen en el futuro. Deberemos para ello desarrollar un prototipo que implemente las funcionalidades necesarias para cumplir con los derechos del RGPD indicados en la sección siguiente.

5.1.3. Derechos del RGPD a tratar

Hasta ahora la implementación de Orion del ayuntamiento solo ha sido utilizada por aplicaciones propias en un ambiente controlado por lo que ciertas cuestiones en materia de seguridad no han sido implementadas. Nuestro módulo se encargará de implementar dichas medidas de seguridad atendiendo a los derechos recogidos en el RGPD ubicados en el cuadro 8.

Derecho del RGPD	Significado
Derecho de Acceso.	Los usuarios deberán saber para que se están usando sus datos.
Derecho de Rectificación.	Los usuarios tienen derecho a poder modificar sus datos si estos son incorrectos.
Derecho de Cancelación	El usuario puede solicitar la supresión de sus datos que resulten ser excesivos.
Derecho de Oposición	El usuario puede oponerse al tratamiento de sus datos cuando, por ejemplo, estos se usen de manera publicitaria.
Derecho al Olvido	El usuario tiene derecho a obtener, sin dilación indebida, la supresión de sus datos personales.
Derecho a la Portabilidad	El usuario podrá pedir que la empresa, en este caso nosotros, le pasemos un archivo con todos los datos sobre él que tenemos.
Limite en el tiempo de conservación de datos	Esta derecho nos obliga como empresa a notificar cuando los datos recopilados van a dejar de usarse.
Protección de datos desde el diseño	El RGPD insiste en el desarrollo de aplicaciones teniendo en cuenta desde el diseño la protección de los datos de los usuarios.

Cuadro 8: Definición de los puntos del RGPD que tratamos en nuestra aplicación

Nuestra aplicación se basa en limitar el acceso a los datos por parte de terceros. Para ello además de filtrar los campos accesibles, para cierto tipo de datos como la fecha de nacimiento aplicaremos el protocolo *"zero knowledge proof"*[5] evitando así la transmisión del dato al solo contestar si el dato cumple la condición dada. Aunque los datos recogidos en Orion incluyen datos no protegidos por el RGPD como todos los referentes a vehículos, en nuestra aplicación trataremos todos los datos como si sí estuvieran protegidos por el RGPD.

Nuestra aplicación además guardará datos sobre los usuarios, en nuestro caso las aplicaciones de las empresas concesionarias que utilicen nuestros servicios, así como las propias empresas. Aunque estos datos no están protegidos por el RGPD y no estamos obligados a cumplir lo dispuesto en él, creemos que debemos tratar estos datos como si estuvieran amparados por el RGPD. El motivo de esta decisión radica en que no sabemos como puede evolucionar el uso de la sistema con el tiempo y es posible que en el futuro se quiera permitir que personas físicas se registren en la aplicación, siendo esos datos protegidos por el RGPD. Por lo tanto y de acuerdo a lo dispuesto en el RGPD permitiremos a los usuarios la consulta, modificación y borrado de sus datos cumpliendo así con los derechos de acceso, rectificación y cancelación. Al tratar los datos de usuarios como si estuvieran amparados por el RGPD aunque de inicio no lo estén, cumple además con la recomendación de abordar la protección de los datos desde el diseño. Además y tal y como está recogido en el RGPD los datos sobre los usuarios se guardarán el mínimo tiempo necesario.

En lo referente a los datos guardados en Orion propios del ayuntamiento u otros organismos gubernamentales como la Dirección General de Tráfico, está recogido su uso y mantenimiento mientras las entidades a las que hacen referencia dichos datos cumplan los requisitos para su inclusión en la base de datos de Orion, por ejemplo ciudadanos empadronados en el municipio. El cumplimiento de los derechos de transparencia y cancelación de dichos datos, corresponde a la implementación de Orion y sus administradores y queda fuera del alcance del modulo a desarrollar. Como nuestro módulo actúa como intermediario entre Orion y las aplicaciones que quieran acceder a él, implementaremos las operaciones necesarias para que estas aplicaciones puedan cumplir con los derechos del RGPD.

Además, dado que la motivación para el desarrollo de este prototipo radica en la adecuación de una aplicación basada FIWARE al RGPD, aseguraremos la protección de datos desde el diseño.

[5] https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/

Derecho del RGPD	Cómo resolverlo
Derecho de Acceso.	En el prototipo, el tratamiento de los datos que se realizará será para la autenticación de las peticiones de los usuario.
Derecho de Rectificación.	Todos los campos de todos los tipos de usuarios podrán ser modificados por sus respectivos propietarios en cualquier momento.
Derecho de Cancelación	Los usuarios podrán borrar total o parcialmente sus datos.
Derecho de Oposición	Ya que no usamos estos datos más que para autenticar a los usuarios si un usuario ejerce este derecho, su cuentas se cancelará y se borrarán todos sus datos.

Cuadro 9: Cómo resolvemos los derechos del RGPD para los datos de usuarios

Derecho del RGPD	Cómo resolverlo
Derecho al Olvido	Para cumplir con este derecho no guardaremos ningún tipo de dato cuando una cuenta de usuario se borre.
Derecho a la Portabilidad	Para cumplir este derecho proporcionaremos un mecanismo que permita a los usuarios obtener sus datos guardados en la aplicación.
Límite en el tiempo de conservación de datos	Los datos que recogemos de los usuarios serán utilizados hasta que éstos se den de baja.

Cuadro 10: Cómo resolvemos los derechos del RGPD para los datos de usuarios(Continuación)

5.1.4. Breve descripción del sistema legado

El sistema a actualizar es una implementación básica del Generic Enabler Orion de FIWARE donde se encuentran los datos referentes al padrón del ayuntamiento así como los datos de vehículos matriculados, a esta Implementación de Orion se conectan directamente las aplicaciones del ayuntamiento como se observa en la figura siguiente.

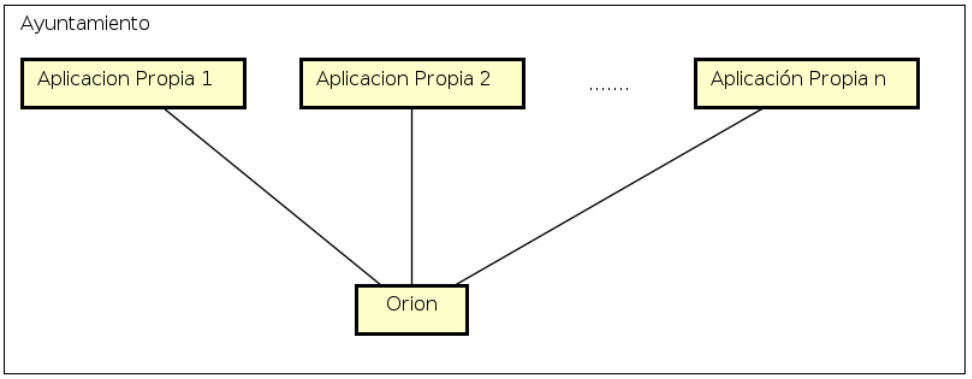

Figura 8: Esquema del sistema inicial

Por cuestiones de compatibilidad con esas aplicaciones, esta implementación de Orion no se puede modificar, por lo que se nos asignado el desarrollo de un módulo de apoyo a Orion al que llamaremos Mímir que se encargue de las cuestiones referentes a la RGPD que no tiene en cuenta Orion. Mímir deberá ademas ser compatible con aplicaciones que ya utilicen Orion por lo que deberá desarrollarse como una API REST.

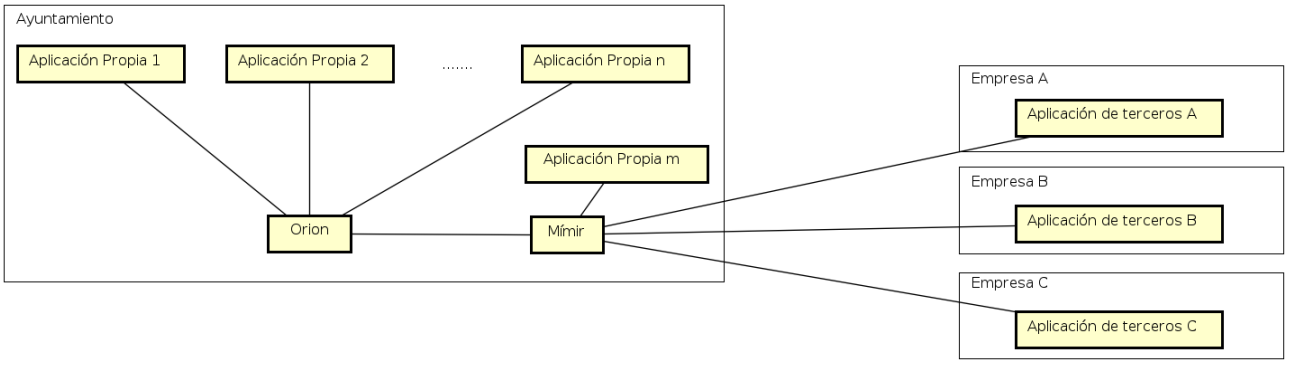

Figura 9: Esquema del sistema objetivo final

Como podemos ver en la figura anterior las aplicaciones de las empresas concesionarias se conectarán a Orion a través de Mímir en vez de directamente. Se tiene planeado que las futuras aplicaciones del ayuntamiento se conecten de la misma forma.

5.1.5. Funcionalidades básicas de Mímir

Mímir se desarrollará como un módulo intermediario entre las aplicaciones y Orion que permita el uso de este únicamente a las aplicaciones registradas y solamente a los datos para los que tienen permisos, asegurando así la seguridad de los datos.

Mímir permitirá añadir, borrar, modificar y consultar los datos de las aplicaciones. Permitirá a dichas aplicaciones añadir, consultar, modificar o borrar los datos almacenados en Orion según los permisos asignados tratándolos cuando sea necesario.

Para poder registrar aplicaciones, las empresas deberán estar previamente registradas en la base de datos de Mímir por el administrador. Mímir permitirá a las empresas registradas acceder a sus datos guardados y su cancelación en cualquier momento, cumpliendo así con los derechos de transparencia y cancelación recogidos en la RGPD. Además cumpliendo con lo dispuesto en la RGPD, estos datos sobre las empresas se guardarán en Mímir mientras las aplicaciones de dichas empresas usen nuestro servicio, hasta la petición de cancelación por la propia empresa o hasta un máximo de 2 años tras su último uso de nuestra aplicación previo aviso de cancelación por nuestra parte.

5.1.6. Datos de Orion

Los datos recogidos en la implementación de Orion del ayuntamiento a la que tendrán acceso las empresas concesionarias, son los referentes a los datos del padrón y vehículos, siendo estos últimos provistos a Orion por la Dirección General de Tráfico y por tanto de ámbito nacional.

En el padrón se recogen obligatoriamente los siguientes datos:

- Nombre
- Apellidos
- Sexo
- Domicilio habitual
- Nacionalidad
- Fecha de nacimiento
- Lugar de Nacimiento
- DNI o NIE

Además de estos datos Orion contiene un campo a mayores con el número de tarjeta para personas de movilidad reducida. En caso de no tener dicha tarjeta el campo se establece en cero.

La DGT nos proporciona los siguientes datos sobre los vehículos:

- Matrícula
- Tipo de combustible
- Marca
- Modelo
- Caballos fiscales
- Color
- DNI del propietario

Los campos tipo de combustible y color se rigen por los siguientes códigos dispuestos por la DGT.

Códigos para tipos de combustible

- “G” para gasolina,
- “D” para Diesel,
- “E” para Eléctrico,
- “GLP” para Gas Licuado del Petróleo (en los mixtos se pondrán los dos combustibles),
- “GNC” para gas natural comprimido y
- “GNL” para gas natural licuado,
- “H” para hidrógeno,
- “BM” Biometanol,
- “ET” Etanol,
- “BD” Biodiesel.

Colores basicos utilizados:

- blanco,
- amarillo,
- naranja,
- rojo,
- púrpura/violeta,
- azul,
- verde,
- gris,
- marrón,
- negro,
- En caso de varios colores: multicolor

5.1.7. Formato de los datos de Orion

Orion emplea como formato para sus datos, un esquema específico en el que todos los atributos excepto id y *type* de una entidad deben estar formados por un JSON en el que se encuentran los campos *value* conteniendo el valor del atributo y *type* indicando que tipo de dato es.

La implementación de Orion del ayuntamiento tiene dos tipos de entidades *Person* y *Vehicle* a continuación se mostrará un ejemplo de cada una de las entidades. En cada uno de ellos el id es el DNI y la Matrícula respectivamente.

- ***Person***

```
{
    "address": {
        "metadata": {},
        "type": "String",
        "value": "Calle Brezo, 3, 4B, Valladolid"
    },
    "birthDate": {
        "metadata": {},
        "type": "String",
        "value": "1995-12-25"
    },
    "birthPlace": {
        "metadata": {},
        "type": "String",
        "value": "Sevilla, Sevilla"
    },
    "familyName": {
        "metadata": {},
        "type": "String",
        "value": "Ramos"
    },
    "gender": {
        "metadata": {},
        "type": "String",
        "value": "Masculino"
    },
    "givenName": {
        "metadata": {},
        "type": "String",
        "value": "Alfonso"
    },
    "id": "786532140",
    "nationality": {
        "metadata": {},
        "type": "String",
        "value": "española"
```

```
    },
    "parkingCard": {
        "metadata": {},
        "type": "Integer",
        "value": 741852
    },
    "type": "Person"
},
```

- ***Vehicle***

```
{
    "brand": {
        "metadata": {},
        "type": "String",
        "value": "Suzuki"
    },
    "color": {
        "metadata": {},
        "type": "String",
        "value": "azul"
    },
    "fiscalPower": {
        "metadata": {},
        "type": "Number",
        "value": 11.6
    },
    "fuelType": {
        "metadata": {},
        "type": "String",
        "value": "D"
    },
    "id": "7619LGA",
    "model": {
        "metadata": {},
        "type": "String",
        "value": "Vitara"
    },
    "owner": {
        "metadata": {},
        "type": "String",
        "value": "98765432B"
    },
    "type": "Vehicle"
},
```

5.1.8. Datos de Mímir

Mímir guardará los datos concernientes a las empresas que hagan uso de él así como las diferentes aplicaciones que estas empresas registren.

- Datos guardados de las empresas autorizadas.
  - NIF
  - Email
  - Contraseña
  - Aplicaciones registradas
- Datos de las aplicaciones.
  - Id aplicación
  - NIF de la empresa
  - Descripción
  - Permisos

5.1.9. Permisos soportados

Cada una de las aplicaciones pueden tener concedidos uno o varios de los siguientes permisos en cualquiera de sus variantes y combinaciones.

Mímir proporcionará permisos CRUD (Creación, Lectura, Actualización y Borrado por sus siglas en inglés) tanto para entidades completas como para los campos de dichas entidades.

En el prototipo a desarrollar se crearán los permisos de lectura, creación y borrado para entidades completas, mientras para cada uno de los campos de las entidades, se crearán los permisos de lectura y actualización.

Mímir proporcionará acceso a los dos tipos de entidades recogidas en Orion, Person y Vehicle.

Por último se concederá acceso en modo lectura a los campos de Person y Vehicle en dos modos, directo, que permite acceder al dato tal y como está guardado en Orion o procesado, que transmite una representación.

- De Persona.
  - **id**: Acceso directo.
  - **name**: Acceso directo.
  - **surname**: Acceso directo.
  - **gender**: Acceso directo.
  - **addres**: Acceso directo.
  - **nationality**: Acceso directo.
  - **birthday**: Acceso directo o procesado.

- **birthplace**: Acceso directo.
- **parking-card**: Acceso directo o procesado.

- De Vehiculo.
  - **id**: Acceso directo
  - **brand**: Acceso directo
  - **combustible**: Acceso directo o procesado
  - **Caballos fiscales**: Acceso directo o procesado
  - **colour**: Acceso directo
  - **owner**: Acceso directo

Para el desarrollo del prototipo se ha concretado el acceso procesado a los campos birthday como boolean que indique si se es mayor de edad, al campo parking-card como boolean que indique si se tiene dicha tarjeta de las entidades Person y el acceso procesado a los campos combustible indicando si es y caballos fiscales indicando al grupo al que pertenece de los siguientes

- 1: De menos de 8 caballos fiscales
- 2: De 8 hasta 11.99 caballos fiscales
- 3: De 12 hasta 15.99 caballos fiscales
- 4: De 16 hasta 19.99 caballos fiscales
- 5: De 20 caballos fiscales en adelante

5.1.10. Requisitos Funcionales

- **RF-01** El sistema permitirá al usuario identificarse.
- **RF-02** El sistema permitirá a los usuarios obtener sus token de acceso asignados.
- **RF-03** El sistema permitirá crear usuarios con rol Empresa o Aplicación.
- **RF-04** El sistema permitirá activar las cuentas con rol Aplicación.
- **RF-05** El sistema permitirá modificar los permisos asignados a los usuarios con rol Aplicación.
- **RF-06** El sistema permitirá modificar los datos de los usuarios.
- **RF-07** El sistema permitirá borrar cuentas de usuario.
- **RF-08** El sistema permitirá a los usuarios empresa leer sus propios datos y los de sus aplicaciones creadas.
- **RF-09** El sistema permitirá a los usuarios con rol empresa modificar y borrar los datos de sus aplicaciones asociadas.

- **RF-10** El sistema permitirá leer, modificar y borrar los datos de cualquier tipo de usuario al administrador.
- **RF-11** El sistema permitirá crear, leer, modificar y borrar los datos guardados en Orion a usuarios con rol Aplicación según sus permisos.
- **RF-12** El sistema deberá poder modificar y/o filtrar los datos pedidos por los usarios en función de los permisos asignados.
- **RF-13** El sistema permitirá crear, leer, borrar y modificar los permisos que se pueden asignar a los usuario con rol aplicación.

5.1.11. Requisitos no Funcionales

Los siguientes requisitos detallan las funcionalidades que debe proporcionar el sistema. Como prerrequisito a cualquiera de ellos el usuario deberá estar logueado en el sistema.

- **RNF-01** Al ser una API REST el sistema debe tener una disponibili- dad lo más cercana posible a 24/7.
- **RNF-02** El sistema será fácilmente escalable.
- **RNF-03** El sistema será capaz de seguir funcionando mientras se hacen las operaciones de mantenimiento.
- **RNF-04** El acceso al sistema debe estar restringido para usuarios no registrados.
- **RNF-05** El sistema será accesible tanto via comando como por naveador web.
- **RNF-06** El sistema será capaz de recuperarse de errores en menos de 24 horas.

5.1.12. Roles

De los requisitos funcionales podemos extraer una serie de roles que diferenciarán a los distintos tipos de usuarios según las funcionalidades a las que tengan acceso. Estos roles son: **Administrador**, **Empresa** y **Aplicación**. El Administrador podrá acceder a todas las vistas y se encargará tanto de la creación de nuevas cuentas de Empresas como de activar las cuentas de Aplicaciones así como de su gestión. La Empresa podrá crear Aplicaciones y las Aplicaciones por su parte podrán acceder a las vistas que proporcionan los datos guardados en Orion.

5.1.13. Posibles *Generic Enablers* a utilizar

Para asegurar la seguridad y privacidad de los datos es necesario poder identificar el origen y filtrar las peticiones que recibamos, por ello nos planteamos la utilización de los siguientes GEs de FIWARE del capítulo de seguridad.

- **KeyRock**: KeyRock nos permitiría gestionar cuentas de usuario y su identificación de un modo sencillo pero por desgracia no permite tener diferentes tipos de usuarios por lo que no podríamos cumplir con los requisitos funcionales planteados.

- **Willma + AuthZForce**: Con AuthZForce podríamos gestionar diferentes permisos para los usuarios y con Willma filtrar las peticiones según dichos permisos. Willma únicamente permite o deniega el paso de las peticiones a los servicios protegidos, por lo que nos sería imposible filtrar dichas peticiones para mostrar solamente los campos a los que tiene acceso el usuario y tampoco podríamos tratar dichos campos para responder si dichos datos cumplen una condición.

Debido a estas limitaciones en los GEs, esenciales para nuestro sistema, se decidió no utilizar ninguno de los GEs citados y en su lugar implementar un módulo propio.

5.1.14. Casos de Uso

Los casos de uso son el conjunto de escenarios que indican como debería interaccionar el sistema con el usuario para llevar a cabo una determinada funcionalidad.

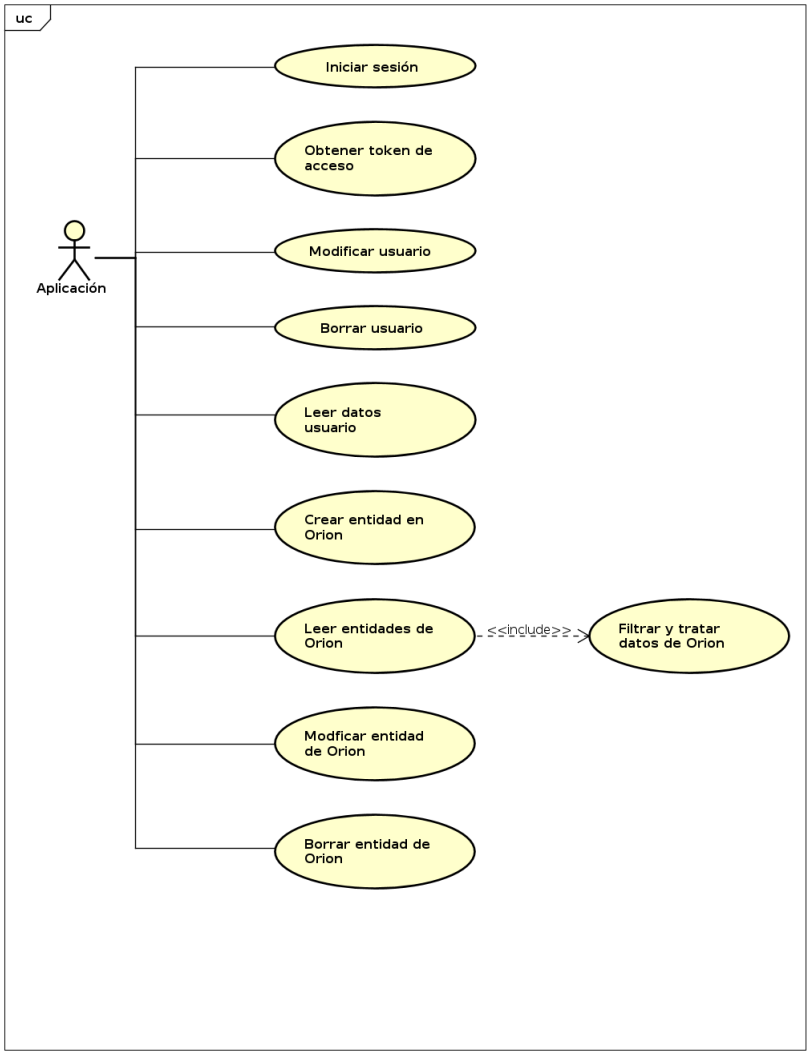

Figura 10: Diagrama de casos de uso del actor Aplicación

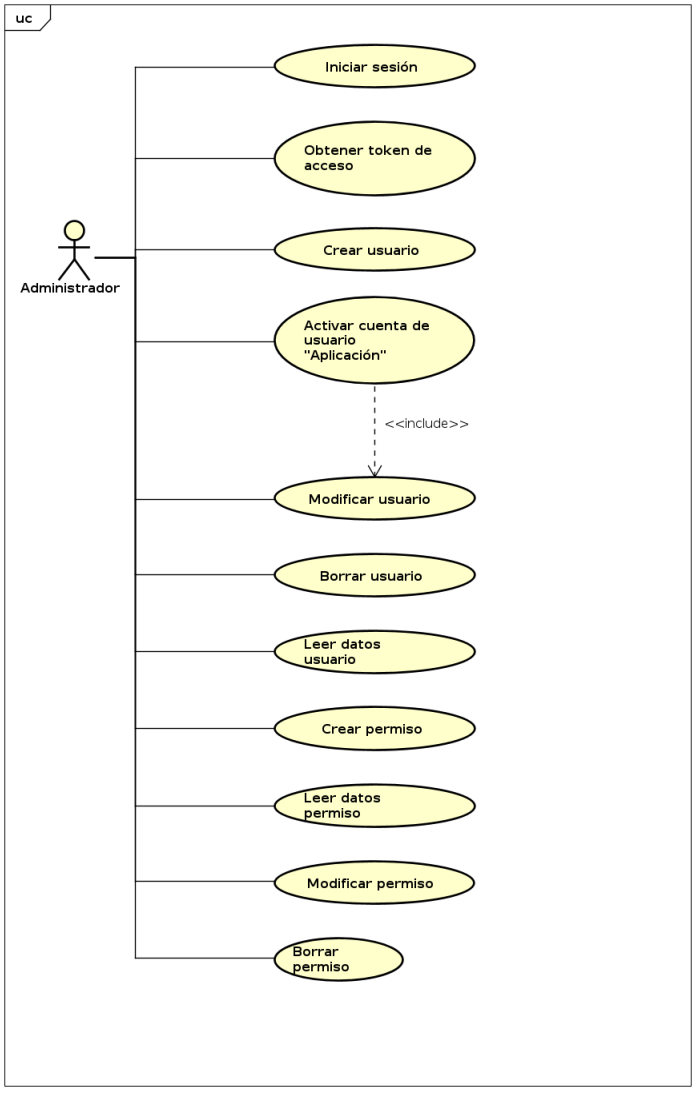

Figura 11: Diagrama de casos de uso del actor Administrador

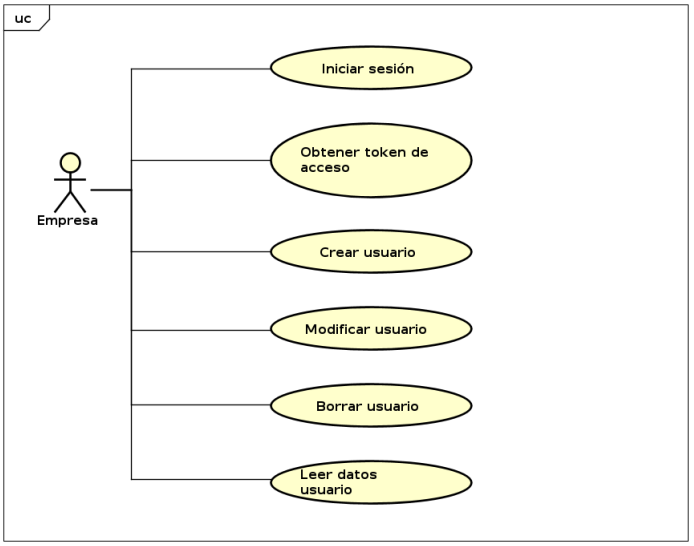

Figura 12: Diagrama de casos de uso del actor Empresa

En las figuras 10, 11 y 12 se muestran los casos de uso para cada uno de los roles que pueden tomar los usuarios del sistema. Cada uno de estos casos de uso tienen asociados una serie de acciones necesarias para poder llevar a cabo una funcionalidad del sistema. En los cuadros del 11 al 26 están detallados dichos pasos junto con las precondiciones necesarias para que se puedan llevar a cabo y las excepciones que pueden suceder en la realización de dichas acciones.

CU-01	Iniciar sesión	
Descripción	El sistema deberá permitir al usuario iniciar sesión.	
Precondición	El usuario debe acceder mediante navegador web	
Secuencia Normal	**Paso**	**Acción**
	1	El sistema solicita las credenciales de acceso, nombre de usuario y contraseña.
	2	El actor usuario introduce las credenciales.
	3	El sistema comprueba con las credenciales si el usuario existe en el sistema.
Excepciones	**Paso**	**Acción**
	2a	El actor usuario cancela la operación y el caso de uso queda sin efecto.
	3a	Si las credenciales no son correctas, muestra un mensaje de error y vuelve al paso 1.

Cuadro 11: Caso de Uso: Iniciar sesión

CU-02	Obtener token de acceso	
Descripción	El sistema deberá permitir al usuario obtener un token de acceso.	
Precondición	El usuario debe acceder mediante comando.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método POST las credenciales.
	2	El sistema comprueba con las credenciales si existe el usuario en el sistema.
	3	El sistema responde con el token de acceso del usuario.
Excepciones	**Paso**	**Acción**
	2a	Si las credenciales no son correctas, el sistema devuelve un mensaje de error.

Cuadro 12: Caso de Uso: Obtener token de acceso

CU-03	Crear usuario	
Descripción	El sistema permitirá crear usuarios con rol Empresa o Aplicación.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método POST los datos referentes al usuario a crear.
	2	El sistema comprueba que los datos recibidos se corresponden con los datos requeridos para crear el usuario.
	3	El sistema crea un nuevo usuario.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario intenta crear un nuevo usuario de un tipo no permitido para su rol, el sistema devuelve un mensaje de error.
	2a	Si los datos no son validos, el sistema devuelve un mensaje de error.
Comentarios	Las cuentas de usuario con rol Aplicación son creadas desactivadas (pueden hacer login pero no pueden acceder a los datos de Orion).	

Cuadro 13: Caso de Uso: Crear usuario

CU-04	Activar cuenta de usuario con rol Aplicación	
Descripción	El sistema permitirá activar las cuentas con rol Aplicación.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El sistema lista las cuentas de usuario con rol aplicación desactivadas.
	2	El usuario elige la cuenta a activar.
	3	Se lanza el caso de uso "Modificar usuarioz el caso de uso finaliza
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene rol Administrador, el sistema devuelve un mensaje de error.
	2a	Si los datos no son validos, el sistema devuelve un mensaje de error.

Cuadro 14: Caso de Uso: Activar cuenta de usuario con rol Aplicación

CU-05	Modificar usuario	
Descripción	El sistema permitirá modificar los datos de los usuarios.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método POST los nuevos datos del usuario a actualizar.
	2	El sistema comprueba que los campos existen para el usuario y que tienen el formato adecuado.
	3	El sistema actualiza los datos del usuario.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene rol Administrador, es el propio usuario o es el dueño del usuario, el sistema devuelve un mensaje de error.
	2a	Si los datos no son validos, el sistema devuelve un mensaje de error.

Cuadro 15: Caso de Uso: Modificar usuario

CU-06	Borrar usuario	
Descripción	El sistema permitirá borrar usuarios.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método DELETE el usuario a eliminar.
	2	El sistema comprueba si el usuario a eliminar existe.
	3	El sistema borra el usuario y el caso de uso finaliza.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene rol Administrador, es el propio usuario o es el dueño del usuario, el sistema devuelve un mensaje de error.
	2a	Si el usuario no existe, el sistema devuelve un mensaje de error.

Cuadro 16: Caso de Uso: Iniciar sesión

CU-07	Leer datos usuario	
Descripción	El sistema permitirá a los usuarios leer sus propios datos y los de sus usuarios creados.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pide los datos de un usuario.
	2	El sistema comprueba si el usuario existe.
	3	El sistema devuelve los datos del usuario.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene rol Administrador, no es el propio usuario o es el dueño del usuario, el sistema devuelve un mensaje de error.
	2a	Si el usuario no existe, el sistema devuelve un mensaje de error.

Cuadro 17: Caso de Uso: Leer datos usuario

CU-08	Crear entidad en Orion	
Descripción	El sistema permitirá crear una nueva entidad en Orion a usuarios con rol Aplicación según sus permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método POST los datos para crear una nueva entidad en Orion.
	2	El sistema comprueba el formato de los datos pasasdos.
	3	El sistema realiza una petición de creación de una entidad a Orion con los datos pasados por el usuario.
	4	El sistema responde al usuario con la respuesta recibida de Orion.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene el rol Aplicación o no tiene los permisos necesarios, el sistema devuelve un mensaje de error.
	2a	Si los datos no tienen el formato adecuado, el sistema devuelve un mensaje de error.

Cuadro 18: Caso de Uso: Crear entidad en Orion

CU-09	Leer entidades de Orion	
Descripción	El sistema permitirá leer entidades guardadas en Orion de un tipo determinado a usuarios con rol Aplicación según sus permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pide al sistema los datos de una o varias entidades del mismo tipo guardadas en Orion.
	2	El sistema pide a Orion los datos de las entidades indicadas por el usuario.
	3	Se lanza el caso de uso "Filtrar y tratar datos de Orion".
	4	El sistema responde al usuario con la respuesta recibida de Orion.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene el rol Aplicación o no tiene los permisos necesarios, el sistema devuelve un mensaje de error.
	2a	Si los datos no tienen el formato adecuado, el sistema devuelve un mensaje de error.

Cuadro 19: Caso de Uso: Leer entidades de Orion

CU-10	Modificar entidad de Orion	
Descripción	El sistema permitirá modificar entidades ya existentes en Orion a usuarios con rol Aplicación según sus permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método PUT los datos a modificar de una entidad ya existente en Orion.
	2	El sistema comprueba el formato de los datos pasados.
	3	El sistema realiza una petición de actualización de una entidad a Orion con los datos pasados por el usuario.
	4	El sistema responde al usuario con la respuesta recibida de Orion.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene rol Administrador, no es el propio usuario o es el dueño del usuario, el sistema devuelve un mensaje de error.
	1b	Si el usuario no tiene el rol Aplicación o no tiene los permisos necesarios, el sistema devuelve un mensaje de error.
	2a	Si los datos no tienen el formato adecuado, el sistema devuelve un mensaje de error.

Cuadro 20: Caso de Uso: Modificar entidad de Orion

CU-11	Borrar entidad de Orion	
Descripción	El sistema permitirá borrar los datos guardados en Orion a usuarios con rol Aplicación según sus permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método DELETE el usuario a eliminar.
	2	El sistema realiza una petición de borrado de la entidad a Orion.
	3	El sistema responde al usuario con la respuesta recibida de Orion.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene rol Administrador, no es el propio usuario o es el dueño del usuario, el sistema devuelve un mensaje de error.

Cuadro 21: Caso de Uso: Borrar entidad de Orion

CU-12	Filtrar y tratar datos de Orion	
Descripción	El sistema podrá filtrar y tratar los datos de Orion pedidos por usuarios con rol Aplicación según sus permisos.	
Precondición	El usuario previamente ha solicitado realizar alguna operación sobre Orion.	
Secuencia Normal	**Paso**	**Acción**
	1	El sistema comprueba los permisos del usuario para el tipo de la entidad.
	2	El sistema elimina los campos de la entidad a los que no tiene acceso el usuario.
	3	El sistema trata los campos de la entidad que así lo requieran.
	4	El sistema devuelve los datos.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene ningún permiso para entidades de ese tipo, el sistema devuelve un mensaje de error.

Cuadro 22: Caso de Uso: Filtrar y tratar datos de Orion

CU-13	Crear permiso	
Descripción	El sistema permitirá crear permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método POST los datos referentes al permiso a crear.
	2	El sistema comprueba que los datos recibidos se corresponden con los datos requeridos para crear el permiso.
	3	El sistema crea un nuevo permiso.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario que intenta crear un permiso no es del rol administrador, el sistema devuelve un mensaje de error.
	2a	Si los datos no son validos, el sistema devuelve un mensaje de error.
Comentarios	Se permiten permisos CRUD tanto para entidades de Orion Completas como para cada uno de sus atributos.	

Cuadro 23: Caso de Uso: Crear permiso

CU-14	Leer datos permiso	
Descripción	El sistema permitirá leer los datos de los permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pide los datos de un permiso.
	2	El sistema comprueba si el permiso existe.
	3	El sistema devuelve los datos del permiso.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene el rol administrador, el sistema devuelve un mensaje de error.
	2a	Si permiso no existe, el sistema devuelve un mensaje de error.

Cuadro 24: Caso de Uso: Leer datos permiso

CU-15	Modificar permiso	
Descripción	El sistema permitirá modificar los datos de los permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía un método POST con los nuevos datos del permiso a actualizar.
	2	El sistema comprueba que los campos existen para el permiso y que tienen el formato adecuado.
	3	El sistema actualiza los campos del permiso.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene el rol Administrador, el sistema devuelve un mensaje de error.
	2a	Si los datos no son válidos, el sistema devuelve un mensaje de error.

Cuadro 25: Caso de Uso: Modificar permiso

CU-16	Borrar permiso	
Descripción	El sistema permitirá borrar permisos.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario envía mediante un método DELETE el permiso a eliminar.
	2	El sistema comprueba si el permiso a eliminar existe.
	3	El sistema borra el permiso.
Excepciones	**Paso**	**Acción**
	1a	Si el usuario no tiene el rol Administrador, el sistema devuelve un mensaje de error.
	2a	Si el permiso no existe, el sistema devuelve un mensaje de error.

Cuadro 26: Caso de Uso: Borrar permiso

5.1.15. Modelo de Dominio

A partir de los casos de uso y los requisitos funcionales y no funcionales podemos extraer una primera aproximación de como se tienen que implementar las clases del sistema.

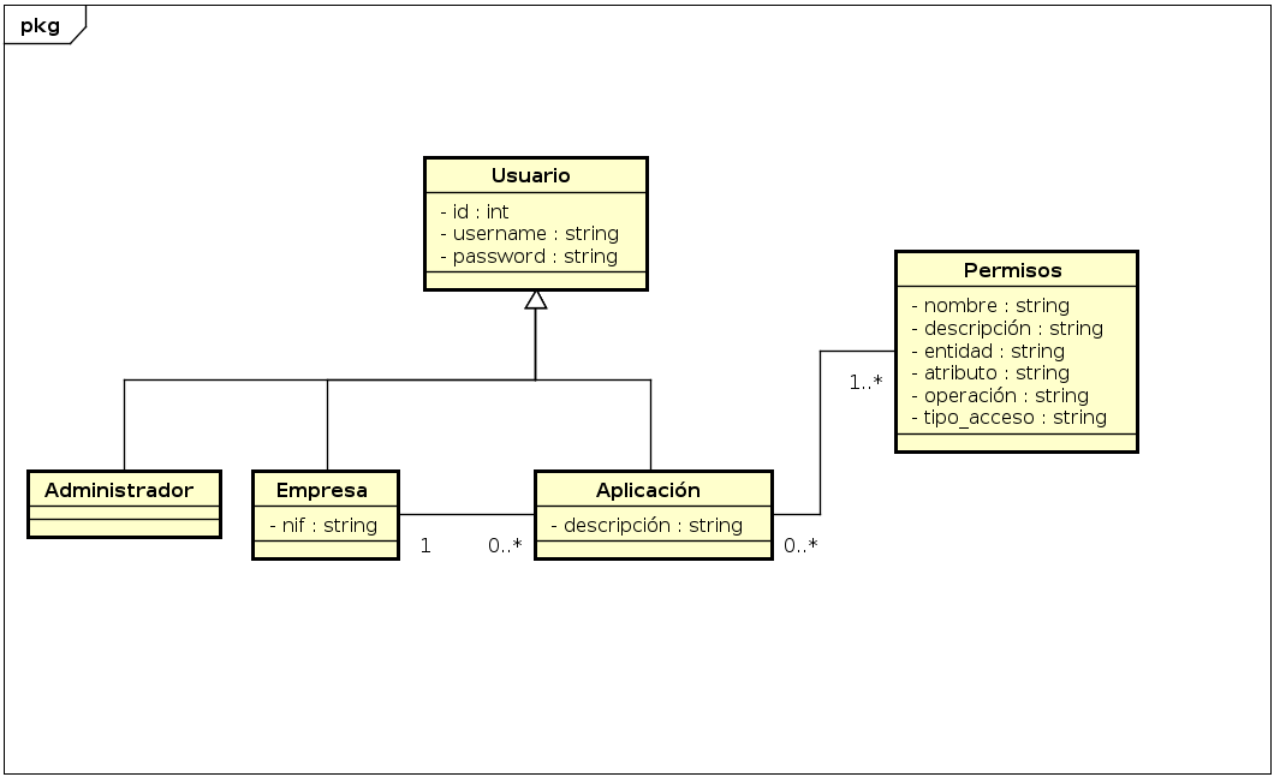

Figura 13: Modelo de dominio inicial

En el figura 13 se muestran las clases mínimas que deberá contener el sistema. A continuación procederemos a describir cada una de ellas.

- **Usuario**: Clase que representa a los usuarios que interaccionarán con el sistema. Como hemos descrito anteriormente necesitamos de diferentes tipos de usuarios que tendrán acceso a distintas funcionalidades, esto se consigue con la relación de generalización/especialización entre la clase general Usuario y las clases hijas Administrador, Empresa y Aplicación.
- **Administrador**: Clase que representa al administrador del sistema. Se encargará de crear las cuentas de usuario de las empresas y de activar las cuentas de aplicaciones. Es un especialización de Usuario.
- **Empresa**: Clase que representa a las empresas. Se podrá gestionar cuentas de aplicaciones. Estará relacionada con las aplicaciones que cree. Es un especialización de Usuario.
- **Aplicación**: Clase que representa a las aplicaciones. Se encargará de acceder a los datos guardados en Orion. Estará relacionada con la empresa que la creó y tendrá asignados unos permisos. Es un especialización de Usuario.
- **Permisos**: Clase que representa los permisos que pueden tener las aplicaciones. Gracias a ella se filtrarán y tratarán los datos de Orion pedidos por las aplicaciones.

5.1.16. Diagramas de secuencia

En este apartado vamos a mostrar los diagramas de secuencia para los casos de uso Crear entidad en Orion y Leer entidades de Orion. En estos diagramas se mostrarán las interacciones entre los actores y el sistema de manera simplificada.

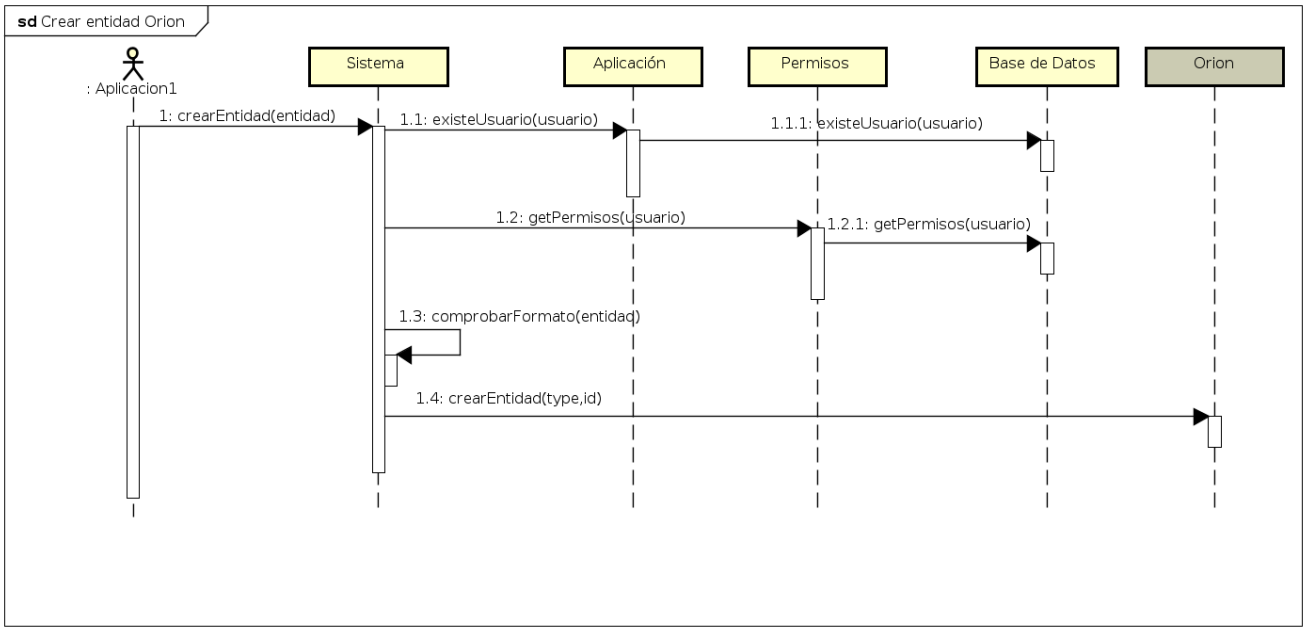

Figura 14: Diagrama de secuencia crear entidad en Orion

La figura 14 se corresponde con el diagrama de secuencia para el caso de uso Crear entidad en Orion. En la secuencia de mensajes se puede ver como el actor envía los datos de la entidad de Orion a crear, el sistema comprueba si el actor existe en el sistema y si tiene los permisos necesarios para llevar a cabo la acción. Tras esto comprueba el formato de los datos y envía la petición de creación de entidad a Orion.

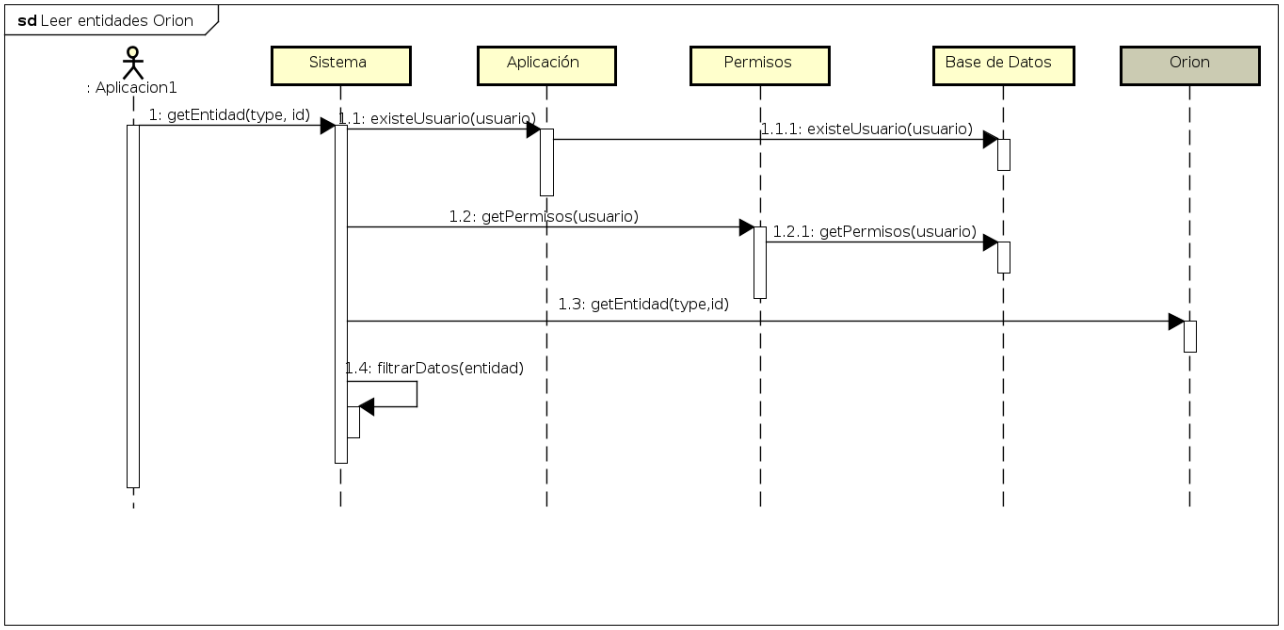

Figura 15: Diagrama de secuencia leer entidad de Orion

La figura 15 se corresponde con el diagrama de secuencia para el caso de uso Leer entidad de Orion, es bastante similar al anterior pero en vez de pasar la entidad a crear se pasan los datos referentes al id de la entidad y el tipo, tras hacer las mismas comprobaciones que en el caso anterior sobre el actor y sus permisos, el sistema pide la entidad a Orion y tras esto filtra los campos recibidos para adecuarlos a los permisos del actor.

5.2. Diseño

5.2.1. Introducción

En este capítulo procederemos a detallar las cuestiones de diseño principales que tendremos en cuenta a la hora de desarrollar el prototipo de nuestra aplicación. En las secciones siguientes expondremos una visión general de la arquitectura que utilizaremos, cuestiones de seguridad de los datos en el diseño que tendremos en cuenta, la estructura final de la base de datos y los patrones utilizados.

5.2.2. Arquitectura

La arquitectura de software es uno de los elementos centrales de toda aplicación. Mediante la relación de distintos patrones y paquetes, se conforma la estructura que permitirá la implementación de la aplicación.

- **Arquitectura REST**

  Para mantener la compatibilidad de nuestro sistema con aplicaciones en producción del ayuntamiento que utilizan Orion que en el futuro podrían migrar a nuestro sistema, seguiremos la arquitectura REST implementada en Orion.

La arquitectura REST está dividida en varios niveles según el tipo de peticiones soportadas.[6]

- **Nivel 0 (POX)**
  En arquitectura de este nivel el cliente accede al servidor únicamente mediante métodos POST y el intercambio de información está formato XML. En este nivel estarían incluidas las aplicaciones tipo SOAP (Simple Object Access Protocol) y estrictamente hablando no podría ser consideradas aplicaciones REST.

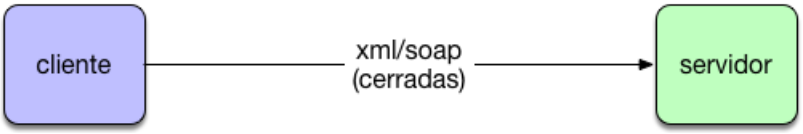

Figura 16: Arquitectura REST Nivel 0. Fuente: Arquitectura Java

- **Nivel 1 (Recursos)**
  Este nivel se define por la apricición del concepto de Recurso para facilitar el acceso a los datos al definir las operaciones que se realizan sobre ellos. El problema de este nivel viende dado por la inconsistencia de su nomenclatura, cada acción viene definida en la url del recurso y nada impide que diferentes recursos tengan diferentes nombres para las mismas acciones.

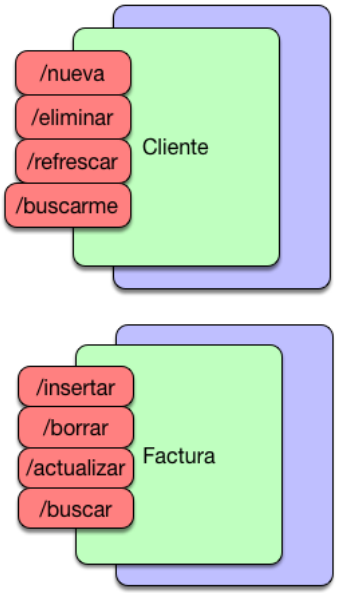

Figura 17: Arquitectura REST Nivel 1. Fuente: Arquitectura Java

- **Nivel 2 (Verbos Http)**
  En nivel 2 evoluciona el concepto anterior al homogeneizar las urls y los verbos que se utilizan para cada operación. En este nivel las urls se definen por el nombre del recurso y los verbos están limitados a GET para obtener los datos, POST para añadir nuevos elementos, PUT para actualizar los datos y DELETE para borrarlos.
  Este nivel es el ofrecido por Orion y por lo tanto es el que emplearemos.

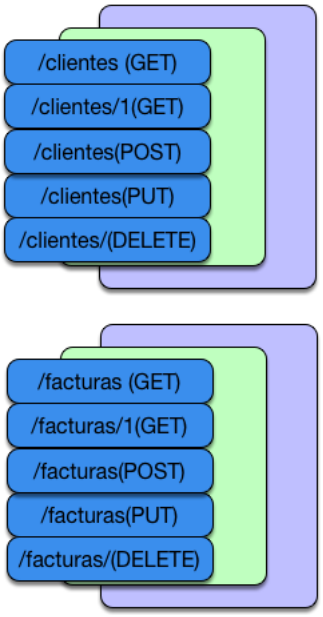

Figura 18: Arquitectura REST Nivel 2. Fuente: Arquitectura Java

- **Nivel 3 (HATEOTAS)**
  Este es el último nivel y su cambio con respecto al nivel anterior radica en la relación entre recursos mediante el uso de enlaces de ahí su nombre "Hypermedia as the Engine of Application State".

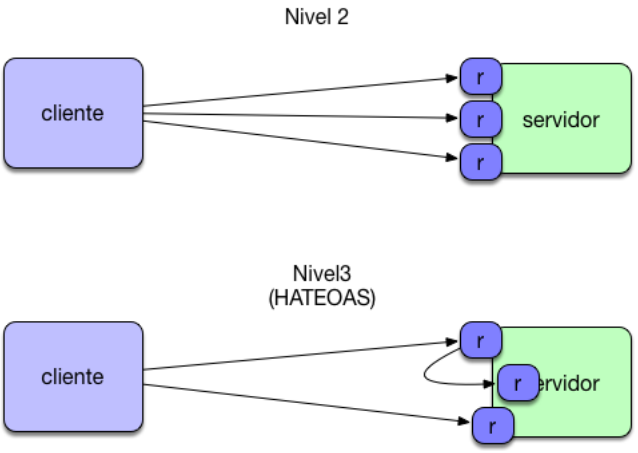

Figura 19: Arquitectura REST Nivel 3. Fuente: Arquitectura Java

- **Arquitectura de Django REST Framework**

  Django REST Framework en adelante DRF es un paquete del framework Django que permite el desarrollo de APIs REST de un forma rápida y sencilla. Django y por tanto DRF se basa en su propia visión del patrón MVC[13] pero con ciertas diferencias en la nomenclatura y funciones. Un patrón MVC se basa en la separación de la representación de los datos de la aplicación (Modelo) de la representación de dichos datos (Vista) y de la gestión y

adaptación de los datos entre estos dos elementos (Controlador). Por otra parte Django hace una interpretación en la que la vista se encarga de que datos se representan, mientras que un template se encarga de como los datos se presentan al usuario, mientras que el modelo es igual que en MVC. Las tareas que realizadas por el controlador en el MVC en Django son realizadas por el propio framework. Podríamos definir a Django como un framework MTV en contraposición con MVC pero como hemos explicado no hay una correspondencia uno a uno entre ellos.

5.2.3. Seguridad desde el diseño

La motivación de esta memoria se centra en la seguridad y privacidad de los datos según el RGPD, una de las recomendaciones consiste en el diseño de las aplicaciones desde el punto de vista de la seguridad y privacidad de los datos. Es por ello que vamos a tener en cuenta ciertas cuestiones en materia de seguridad a la hora de diseñar el sistema.

- Uno de los puntos críticos en una aplicación web a la hora de mantener la privacidad y seguridad de sus datos radica en el momento de la autenticación del usuario contra el sistema. Hay que tener especial cuidado en este punto ya que el usuario debe mandar sus credenciales al sistema y estas pueden ser interceptadas. Es por ello que la autenticación del usuario nunca se hará mediante un método GET en el cual el único modo de enviar parámetros es a través de la propia url.

```
1        http://localhost:8000/login?username=user1&password=1234
```

  Este problema lo evitaremos usando los backends de autenticación y end-points de login de Django REST Framework ya que están preparados y probados para asegurar la privacidad de las credenciales del usuario durante el proceso y el uso exclusivo de HTTPS en la aplicación.

- Otro de los riesgos a los que están expuestas las aplicaciones es la ejecución de código SQL no previsto en el sistema, conocido como inyección SQL. para evitarlo, las peticiones al sistema deben ser filtradas y escapadas.

  Este problema también nos lo resuelve DRF ya que los accesos a la base de datos los realiza el propio framework a través de los modelos que definamos y por lo tanto es el propio framework el que se encarga de filtrar y escapar los datos introducidos.

5.2.4. Diseño de la Base de Datos

Para el diseño de la Base de datos vamos a tener en cuenta los mismos principios a los que aspiramos con la creación de esta aplicación. Estos son, el requerimiento de los mínimos datos necesarios de los usuarios para poder realizar las funcionalidades del sistema, limitación del uso de los datos exclusivamente al fin para el que fueron recogidos y por último asegurar la exactitud, integridad y confidencialidad de los datos. Para este punto por ejemplo permitiremos la actualización de los datos por parte de los usuarios para así mantener su exactitud y la contraseña (dato más sensible a guardar ya que es la puerta de entrada al resto de los datos) la guardaremos tras ser encriptada.

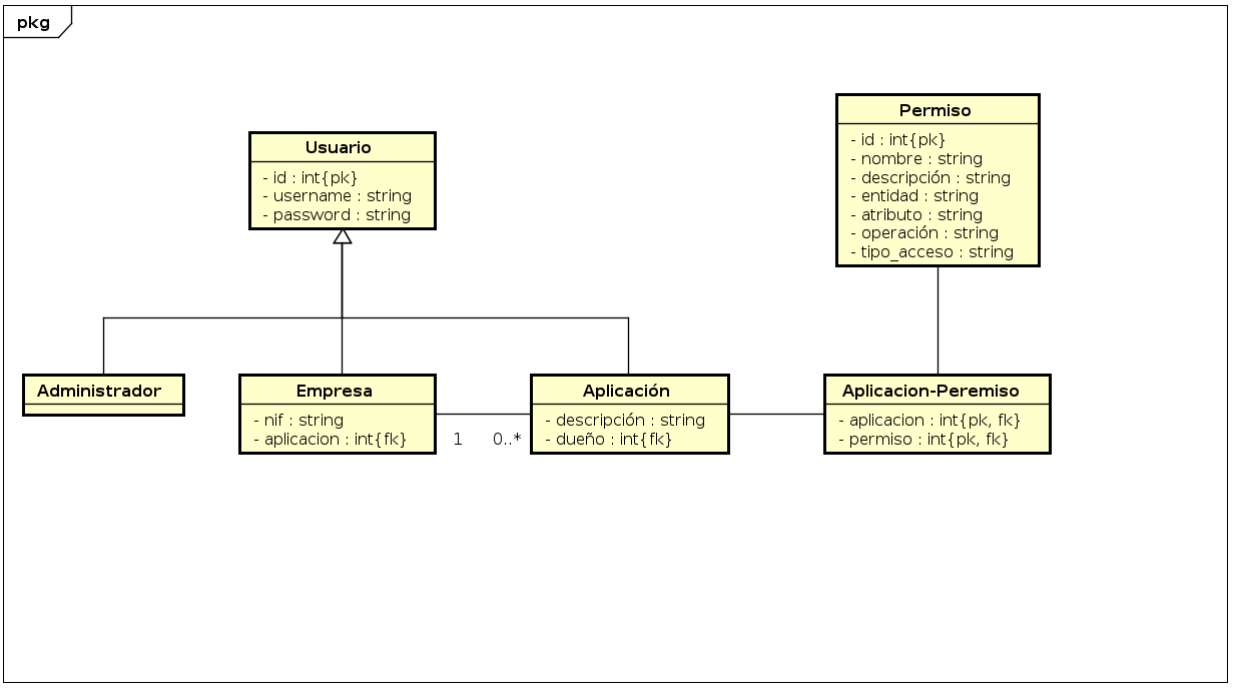

Figura 20: Modelo conceptual de los datos

La figura 20 no ha cambiado mucho desde la fase de análisis, pero en su paso hasta el modelo conceptual de la base de datos hemos tenido que añadir una clase intermedia entre Aplicación y Permiso para poder mantener la relación varios a varios.

5.3. Implementación

5.3.1. Introducción

En este capítulo están detallados los aspectos específicos de implementación del sistema planteado en los capítulos anteriores de análisis y diseño, aplicado a las características especificas de Django REST Framework. Además mostraremos porciones de código de ejemplo de cada uno de los elementos que componen el sistema.

5.3.2. Descripción de los recursos utilizados

Para la implementación del sistema hemos utilizado los siguientes elementos:

- Python como lenguaje para el backend.
- El framework Django para la creación de la aplicación web. Django permite el desarrollo de aplicaciones web en python de una forma sencilla y estructurada.
- El paquete Django REST Framework para crear una API REST accesible via navegador. Este paquete nos permite la creación de una API REST en un proyecto Django de forma rápida, además de auto generar las vistas que permiten que dicha API sea accesible desde el navegador.

Django recomienda separar los proyectos por apps, conjuntos de funcionalidades relacionadas. Siguiendo esta recomendación, dividiremos nuestro proyecto en dos app "accounts" que se encargará de la gestión de las cuentas de usuario y "request" que se encargará de las interacciones con Orion.

Este modo de dividir el proyecto nos permite por ejemplo la reutilización de "accounts" en otros proyectos con necesidades de gestión de usuarios similares.

5.3.3. Models

En Django los modelos son la representación de las tablas de la base de datos. Un modelo es una clase de Python que define los campos que contiene la tabla de la base de datos.

En Django solamente se puede tener un modelo que actue como usuario, para lograr tener varios tipos de usuarios existen dos aproximaciones diferentes:

- **Perfiles**: Este método se basa en crear pequeños modelos que contengan las características específicas de cada tipo de usuario. Estos modelos tedrían una relación uno a uno con el modelo usuario
- **Herencia**: Este método se basa en el uso de la herencia para crear diferentes modelos que hereden de la clase *User* o *AbstractUser* y por lo tanto se puedan identificar contra el sistema.

Hemos decidido implementar la segunda opción ya que de esta forma podemos autenticar a los diferentes tipos de usuarios cada uno con sus datos, sin tener que crear una entrada en la tabla padre para cada usuario. En el siguiente fragmento de código se muestran el campo *permissions* del modelo *AppUser*, modelo representativo de los usuarios con rol aplicación. Este campo define la relación varios a varios comentada en la sección de diseño.

```
1     permissions = models.ManyToManyField(Permission, related_name='apps')
```

En *related_name* se define el nombre del campo en *Permissions* que referencia a este modelo.

Para el rol administrador nos hemos decantado por usar la implementación de superusuario de Django ya que contiene todo lo necesario para este rol. Para crear un administrador utilizamos el siguiente comando:

```
1 python3 manage.py createsuperuser
```

El modelo padre de los usuarios "CustomUser" heredará a su vez de "AbstractUser" para así mantener todas las funcionalidades de autenticación de usuario que proporciona Django pero permitiendo que en el futuro se pudiera modificar el comportamiento de la esta clase.

Aparte de los modelos para los usuarios tenemos otro modelo para los diferentes tipos de permisos. Estos permisos contienen los siguientes campos:

Los permisos por su parte constarán de los siguientes campos:

- Nombre
- Descripción

- Tipo de la entidad de Orion
- Atributo dentro de esa entidad.
- Tipo de operación permitida
- Tipo de acceso permitido, directo o tras procesar el dato
- Aplicaciones a las que se les ha concedido dicho permiso

Cada uno de los campos de los campos de un permiso excepto los campos nombre y descripción, están acotados por las operaciones CRUD que se pueden llevar a cabo en Orion y por el formato de los datos guardados en él. Es por eso que hemos definido una serie de constantes en el fichero *constants.py* ubicado en el directorio raíz del proyecto, para delimitar las opciones que pueden tomar estos datos, impidiendo la creación de permisos para atributos o tipos de entidades inexistentes. Además de esta manera DRF puede generar en la vista web un formulario con los campos ya acotados, facilitando la creación de permisos. En el siguiente fragmento de código se muestran las opciones disponibles para las operaciones a realizar.

```
CREATE = 'Create'
READ = 'Read'
UPDATE = 'Update'
DELETE = 'Delete'
TYPE = (
    ( CREATE, 'Create' ),
    ( READ, 'Read' ),
    ( UPDATE, 'Update' ),
    ( DELETE, 'Delete' ),
)
```

Con este modelo se podrán añadir o modificar permisos de ser necesario y ampliar las opciones posibles simplemente añadiéndolas a *constants.py*.

En la app "request" hemos seguido un enfoque diferente. En esta app los datos a modelar, no se encuentran en una base de datos accesible directamente, si no que están accesibles a través de Orion. Por lo tanto hemos creado un modelo por cada tipo de entidad en Orion no asociado a una base de datos. De esta manera, podemos usar *serializers* con los datos provenientes de Orion. En el fragmento de código siguiente se puede ver como el modelo para *Person* únicamente implementa el método *__init__()*.

```
class Person(object):
    def __init__(self, **kwargs):
        for field in ('id','type','givenName','familyName','gender','address',
                      'nationality','birthDate','birthPlace','parkingCard'
        ):
            setattr(self, field, kwargs.get(field,None))
```

5.3.4. Serializers

Los serializers son unas clases propias de DRF que permiten convertir las instancias de los modelos y las queryset (colecciones de objetos provenientes de la base de datos) en tipos nativos de Python que puedan ser renderizados en JSON o XML y viceversa.

Como tal necesitamos un *serializer* por cada modelo que queramos representar en los *end-points*. El *serializer* define que campos del modelo va a serializar/deserializar y en él se definen las operaciones que se realizan sobre las instancias del modelo. En nuestro caso nos ha sido necesario redefinir los métodos *create()* y *update()* para que en el campo password quede guardado el hash de la contraseña en vez de la contraseña en texto plano.

En el siguiente fragmento de código podemos ver el método *create()* de *AppSerializer*, *serializer* para el modelo *AppUser*

```
def create(self, validated_data):
    owner = BusinessUser.objects.get( username=self.context['request'].user )
    app, created = AppUser.objects.update_or_create(
        username=validated_data['username'],
        description=validated_data['description'],
        password='1234',
        owner=owner,
        active=False,
    )
    app.permissions.set(validated_data['permissions'])
    app.set_password(validated_data['password'])
    app.save()
    return app
```

Este método se corresponde con el *end-point* para crear aplicaciones al que tendrán acceso las cuentas empresa. Es por ello que el campo *owner* se establece automáticamente al usuario que ha hecho la petición, evitando así que una empresa pueda crear aplicaciones cuyo dueño sea otra empresa. Además en este método se relacionan los permisos con la aplicación creada y se guarda el hash de la contraseña.

Los serializers de la app *"request"* son similares a los de *"accounts"* pero ya que el modelo no define los tipos de datos, es en el serializer donde debemos indicar el tipo de datos esperado. Por otra parte Orion proporciona dos modos de formateo de los datos, completa que muestra los datos con el tipo y los metadatos o compacta que únicamente muestra el valor, para lograr esta misma funcionalidad hemos creado dos serializers diferentes para cada tipo de entidad, uno para la salida completa y otro para la compacta. Además ya que necesitamos filtrar los datos según los permisos de la aplicación usuario, debemos modificar la representación de los serializers para que eliminen los campos no requeridos.

5.3.5. Views

Como indicamos anteriormente la definición de vista es diferente en MVC y Django. En Django las vistas se encargan de que datos se presentan en vez de como se presentan, dejando esta últia tarea al template. En nuestro caso al tratarse nuestro sistema de una API REST, hemos dejado

los templates por defecto de DRF centrandonos en las vistas y que datos devuelven cada una de ellas.

DRF implementa los llamados *ViewSets* clases que aglutinan las diferentes vistas (Verbos HTTP) para un mismo recurso en métodos. Entre ellos destaca *ModelViewSet* ya que implementa todas las vistas posibles para un recurso, por ello nuestras vistas están definidas como *ViewSets* que heredan de esta clase. Así solamente hemos tenido que redefinir algunos métodos para adecuar su funcionalidad a los requisitos específicos de nuestro proyecto.

En las vistas se definen que datos se muestran para cada uno de los verbos HTTP. Uno de los problemas que nos hemos encontrado a la hora de generar las vistas de un elemento concreto, es que en la URL de un elemento este es identificado por su id, campo auto generado por Django en la base de datos. Para poder identificar las instancias de los modelos por su nombre, hemos tenido que redefinir cada uno de los métodos *retrieve()*. En el siguiente fragmento de código se muestra el método *retrieve()* de la vista *PermissionViewSet* encargada de las vistas concernientes a los permisos.

```
1 def retrieve(self, request, pk=None):
2     permission = Permission.objects.get(name=pk)
3     serializer = PermissionSerializer(instance=permission, many=False )
4     return Response(serializer.data)
```

De esta forma la URL referente al permiso *readPerson* se define como:
localhost:8000/permissions/readPerson
en vez de:
localhost:8000/permissions/1

6. Pruebas

6.1. Introducción

En la siguiente sección se detallarán las pruebas llevadas a cabo para detectar los fallos con respecto a los requisitos planteados anteriormente en la memoria. Para ellos vamos a realizar pruebas de caja negra que nos servirán para realizar pruebas funcionales del sistema. Los casos de prueba listados a continuación se realizarán para cada uno de los end-points del sistema basándonos en los casos de uso detallados anteriormente. Al tratarse de un prototipo se ha tratado de cumplir con los requisitos de la mejor manera posible, pero habrá muchos escenarios que no se hayan tenido en cuenta y estas pruebas nos llevarán a encontrar algunos de ellos, aunque deberían hacerse muchas más pruebas de elementos específicos para lograr una aplicación que pueda pasar a producción y todavía mas pruebas hasta llegar a un sistema totalmente robusto.

6.2. Pruebas generales

- admin

CP-01	Crear una empresa	
Descripción	Crear un nuevo usuario empresa	
Entrada	**End-point**	/user-admin/
	Método	POST
	Payload	username=Test taxID=A1234567A email=b1@b.com password=qazwsxed
Salida esperada	**Código**	201
	Payload	Campos de la empresa creada en formato JSON
Salida real	Salida esperada	

Cuadro 27: CP-01: Crear empresa

CP-02	Crear una empresa ya existente	
Descripción	Crear un nuevo usuario empresa con un nombre de usuario ya existente	
Entrada	**End-point**	/user-admin/
	Método	POST
	Payload	username=Test taxID=A1234567A email=b1@b.com password=qazwsxed
Salida esperada	**Código**	400
	Payload	“username”:[“A user with that username already exists”]
Salida real	Salida esperada	

Cuadro 28: CP-02: Crear una empresa ya existente

CP-03	Actualizar una empresa	
Descripción	Actualizar un usuario empresa existente	
Entrada	**End-point**	/user-admin/
	Método	POST
	Payload	username=TestTest taxID=B1234567B email=b1@b.com password=1234
Salida esperada	**Código**	200
	Payload	Campos de la entidad actualizada en formato JSON. Campo password con el HASH del password
Salida real	**Código**	200
	Payload	Campos de la entidad actualizada en formato JSON. Campo password con el password en texto plano

Cuadro 29: CP-03: Actualizar una empresa

La contraseña se ha guardado como texto plano en vez de su HASH. Para solucionar este problema debemos sobre escribir el método *update()* de BusinessUserSerializer para que al igual que en el métoco *create()*, se genere y guarde el HASH de la contraseña. Este cambio lo hemos tenido que realizar tanto para BusinessUserSerializer como para AppUserSerializer y AppUserAdminSerializer.

CP-04	Listar todas las empresas	
Descripción	Devuelve todos los datos de todas las empresas	
Entrada	**End-point**	/user-admin/
	Método	GET
Salida esperada	**Código**	200
	Payload	Lista de todas las empresas con todos sus campos en formato JSON
Salida real	Salida esperada	

Cuadro 30: CP-04: Listar todas las empresas

CP-05	Leer datos de una empresa	
Descripción	Devuelve todos los datos de una empresa	
Entrada	**End-point**	/user-admin/TestTest/
	Método	GET
Salida esperada	**Código**	200
	Payload	Datos de la empresas en formato JSON
Salida real	Salida esperada	

Cuadro 31: CP-05: Leer datos de una empresa

CP-06	Eliminar una empresa	
Descripción	Elimina una empresa determinada	
Entrada	**End-point**	/user-admin/TestTest/
	Método	DELETE
Salida esperada	**Código**	204
Salida real	Salida esperada	

Cuadro 32: CP-06: Eliminar una empresa

Para las siguientes pruebas existe un usuario Business2 de tipo empresa al que asociaremos la aplicación creada.

CP-07	Crear una aplicación	
Descripción	Crear un nuevo usuario aplicación	
Entrada	**End-point**	/app-admin/
	Método	POST
	Payload	username=TestApp description="App Test" password=qazwsxed owner=Business2 permissions:='["readPerson","createPerson"]'
Salida esperada	**Código**	201
	Payload	Campos de la aplicación creada en formato JSON
Salida real	Salida esperada	

Cuadro 33: CP-07: Crear una aplicación

- empresa

  Para llevar a cabo los siguientes casos de prueba deben existir en el sistema los usuarios de tipo empresa Business1 y Business2.

  Para los casos de prueba de las tablas 45 a 49 debemos estar identificados como Business2, mientras que para los casos de prueba de las tablas 50 a 62 debemos estar identificados como Business1

CP-08	Crear una aplicación ya existente	
Descripción	Crear un nuevo usuario aplicación con un nombre de usuario ya existente	
Entrada	**End-point**	/app-admin/
	Método	POST
	Payload	username=TestApp description=“App Test” password=qazwsxed owner=Business2 permissions:=‘[“readPerson”,“createPerson”]’
Salida esperada	**Código**	400
	Payload	“username”:[“A user with that username already exists”]
Salida real	Salida esperada	

Cuadro 34: CP-08: Crear una aplicación ya existente

CP-09	Listar todas las aplicaciones	
Descripción	Devuelve todos los datos de todas las aplicaciones	
Entrada	**End-point**	/app-admin/
	Método	GET
Salida esperada	**Código**	200
	Payload	Lista de todas las aplicaciones con active = False con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 35: CP-09: Listar todas las aplicaciones

CP-10	Actualizar una aplicación	
Descripción	Actualizar un usuario aplicación existente	
Entrada	**End-point**	/app-admin/TestApp
	Método	PUT
	Payload	username=TestAppTest description=“App Test” password=qazwsxed owner=Business2 permissions:=‘[“readPerson”,“createPerson”]’ active:=true
Salida esperada	**Código**	200
	Payload	Aplicación con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 36: CP-10: Actualizar una aplicación

CP-11	Leer datos de una aplicación	
Descripción	Devuelve todos los datos de una aplicación determinada	
Entrada	**End-point**	/app-admin/TestAppTest
	Método	GET
Salida esperada	**Código**	200
	Payload	Aplicación con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 37: CP-11: Leer datos de una aplicación

CP-12	Eliminar una aplicación	
Descripción	Elimina una aplicación determinada	
Entrada	**End-point**	/app-admin/TestAppTest
	Método	DELETE
Salida esperada	**Código**	204
Salida real	Salida esperada	

Cuadro 38: CP-12: Eliminar una aplicación

CP-13	Crear un permiso	
Descripción	Crear un nuevo permiso	
Entrada	**End-point**	/permissions/
	Método	POST
	Payload	name=TestPermission description="Permission Test" entity=Vehicle attribute=owner type=Read access_type=Processed
Salida esperada	**Código**	201
	Payload	Campos del permiso creado en formato JSON
Salida real	Salida esperada	

Cuadro 39: CP-13: Crear una aplicación

CP-14	Crear un permiso ya existente	
Descripción	Crear un nuevo permiso con un nombre ya existente	
Entrada	**End-point**	/permissions/
	Método	POST
	Payload	name=TestPermission description="Permission Test" entity=Vehicle attribute=owner type=Read access_type=Processed
Salida esperada	**Código**	400
	Payload	"name":["A user with that username already exists"]
Salida real	Salida esperada	

Cuadro 40: CP-14: Crear un permiso ya existente

CP-15	Actualizar un permiso	
Descripción	Actualizar un permiso existente	
Entrada	**End-point**	/permissions/TestPermission
	Método	PUT
	Payload	name=TestPermissionTest description="Permission Test mod" entity=Vehicle attribute=brand type=Read access_type=Processed
Salida esperada	**Código**	200
	Payload	Permiso con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 41: CP-15: Actualizar un permiso

CP-16	Listar todos los permisos	
Descripción	Devuelve todos los datos de todas las aplicaciones	
Entrada	**End-point**	/permissions/
	Método	GET
Salida esperada	**Código**	200
	Payload	Lista de todos los permisos con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 42: CP-16: Listar todos los permisos

CP-17	Leer datos de un permiso	
Descripción	Devuelve todos los datos de un permiso determinado	
Entrada	**End-point**	/permissions/TestPermissionTest/
	Método	GET
Salida esperada	**Código**	200
	Payload	Permiso con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 43: CP-17: Leer datos de un permiso

CP-18	Eliminar un permiso	
Descripción	Elimina un permiso determinado	
Entrada	**End-point**	/permissions/TestPermissionTest/
	Método	DELETE
Salida esperada	**Código**	204
Salida real	Salida esperada	

Cuadro 44: CP-18: Eliminar un permiso

CP-19	Crear una empresa	
Descripción	Crear un nuevo usuario empresa	
Entrada	**End-point**	/users/
	Método	POST
	Payload	username=Test taxID=A1234567A email=b1@b.com password=qazwsxed
Salida esperada	**Código**	405
	Payload	{“description”:“Method not allowed}
Salida real	Salida esperada	

Cuadro 45: CP-19: Crear empresa

CP-20	Actualizar una empresa	
Descripción	Actualizar datos propios	
Entrada	**End-point**	/users/Business2
	Método	PUT
	Payload	username=Business2 taxID=X1234569X email=b5@b.com password=qazwsxed
Salida esperada	**Código**	200
	Payload	Campos de la entidad actualizada en formato JSON. Campo password con el HASH del password
Salida real	Salida esperada	

Cuadro 46: CP-20: Actualizar una empresa

CP-21	Listar todas las empresas	
Descripción	Devuelve todos los datos de todas las empresas	
Entrada	**End-point**	/users/
	Método	GET
Salida esperada	**Código**	200
	Payload	Datos propios en formato JSON
Salida real	Salida esperada	

Cuadro 47: CP-21: Listar todas las empresas

CP-22	Leer datos de una empresa	
Descripción	Devuelve todos los datos de una empresa	
Entrada	**End-point**	/users/Business2/
	Método	GET
Salida esperada	**Código**	200
	Payload	Datos propios en formato JSON
Salida real	Salida esperada	

Cuadro 48: CP-22: Leer datos de una empresa

CP-23	Eliminar una empresa	
Descripción	Elimina una empresa determinada	
Entrada	**End-point**	/users/Business2/
	Método	DELETE
Salida esperada	**Código**	204
0 **Salida real**	Salida esperada	

Cuadro 49: CP-23: Eliminar una empresa

CP-24	Crear una aplicación	
Descripción	Crear un nuevo usuario aplicación	
Entrada	**End-point**	/apps/
	Método	POST
	Payload	username=TestApp description="App Test" password=qazwsxed permissions:='["readPerson","createPerson"]'
Salida esperada	**Código**	201
	Payload	Campos de la aplicación creada en formatoJSON
Salida real	Salida esperada	

Cuadro 50: CP-24: Crear una aplicación

CP-25	Crear una aplicación ya existente	
Descripción	Crear un nuevo usuario aplicación con un nombre de usuario ya existente	
Entrada	**End-point**	/apps/
	Método	POST
	Payload	username=TestApp description="App Test" password=qazwsxed permissions:='["readPerson","createPerson"]'
Salida esperada	**Código**	400
	Payload	"username":["A user with that username already exists"]
Salida real	Salida esperada	

Cuadro 51: CP-25: Crear una aplicación ya existente

CP-26	Actualizar una aplicación	
Descripción	Actualizar un usuario aplicación existente	
Entrada	**End-point**	/apps/TestApp
	Método	PUT
	Payload	username=TestAppTest description=."App Test mod"password=qazwsxed permissions:='["readPerson","createPerson"]', owner=Business2
Salida esperada	**Código**	200
	Payload	Aplicación con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 52: CP-26: Actualizar una aplicación

CP-27	Listar todas las aplicaciones	
Descripción	Devuelve todos los datos de todas las aplicaciones de la empresa	
Entrada	**End-point**	/apps/
	Método	GET
Salida esperada	**Código**	200
	Payload	Lista de todas las aplicaciones del usuario con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 53: CP-27: Listar todas las aplicaciones

CP-28	Leer datos de una aplicación	
Descripción	Devuelve todos los datos de una aplicación determinada	
Entrada	**End-point**	/apps/TestAppTest
	Método	GET
Salida esperada	**Código**	200
	Payload	Aplicación con sus datos en formato JSON
Salida real	Salida esperada	

Cuadro 54: CP-28: Leer datos de una aplicación

CP-29	Elmiminar una aplicación	
Descripción	Elimina una aplicación determinada	
Entrada	**End-point**	/apps/TestAppTest
	Método	DELETE
Salida esperada	**Código**	204
Salida real	Salida esperada	

Cuadro 55: CP-29: Elmiminar una aplicación

- aplicación

  Para llevar a cabo los siguientes casos de prueba debe existir en el sistema un usuario App1 con los siguientes permisos:

  - createPerson
  - deletePerson
  - updatePerson
  - namePerson
  - birthDatePersonProcessed
  - parkingCardPersonProcessed

CP-30	Crear una entidad de Orion	
Descripción	Crear una nueva entidad de Orion	
Entrada	**End-point**	/v2/entities/
	Método	POST
	Payload	Datos de una nueva entidad de Orion en formato JSON
Salida esperada	**Código**	201
	Payload	Campos de la entidad creada en formato JSON
Salida real	Salida esperada	

Cuadro 56: CP-30: Crear una entidad de Orion

CP-31	Crear una entidad de Orion ya existente	
Descripción	Crear una nueva entidad de Orion con un id ya existente	
Entrada	**End-point**	/v2/entities/
	Método	POST
	Payload	Datos de una nueva entidad de Orion en formato JSON
Salida esperada	**Código**	422
	Payload	{"error": "Already exists"}
Salida real	Salida esperada	

Cuadro 57: CP-31: Crear una entidad de Orion ya existente

CP-32	Listar todas las entidades de Orion de un tipo determinado	
Descripción	Lista todas los entidades de Orion de un tipo determinado con sus datos según los permisos del usuario	
Entrada	**End-point**	/v2/entities/?type=Person
	Método	GET
Salida esperada	**Código**	200
	Payload	Lista de todas las entidades en formato compacto mostrando los campos accesibles según los permisos del usuario.
Salida real	Salida esperada	

Cuadro 58: CP-32: Listar todas las entidades de Orion de un tipo determinado

CP-33	Leer datos de una entidad de Orion	
Descripción	Devuelve los datos de una entidad de Orion determinado	
Entrada	**End-point**	/v2/entities/12345678D/?type=Person
	Método	GET
Salida esperada	**Código**	200
	Payload	Datos de la entidad en formato compacto mostrando los campos accesibles según los permisos del usuario.
Salida real	Salida esperada	

Cuadro 59: CP-33: Leer datos de una entidad de Orion

CP-34	Leer atributo de una entidad de Orion	
Descripción	Devuelve un único dato de una entidad de Orion determinado	
Entrada	**End-point**	/v2/entities/12345678D/attrs/givenName?-type=Person
	Método	GET
Salida esperada	**Código**	200
	Payload	Atributo en formato JSON
Salida real	Salida esperada	

Cuadro 60: CP-34: Leer atributo de una entidad de Orion

CP-35	Actualizar un atributo de una entidad de Orion	
Descripción	Actualiza un único atributo de una entidad de Orion determinada	
Entrada	**End-point**	/v2/entities/12345678D/attrs/givenName?-type=Person
	Método	PUT
	Payload	Nuevo dato de la entidad correctamente formateado en JSON
Salida esperada	**Código**	204
Salida real	Salida esperada	

Cuadro 61: CP-35: Actualizar un atributo de una entidad de Orion

CP-36	Eliminar una entidad de Orion	
Descripción	Elimina una entidad de Orion determinada	
Entrada	**End-point**	/v2/entities/12345678D/?type=Person
	Método	DELETE
Salida esperada	**Código**	204
Salida real	Salida esperada	

Cuadro 62: CP-36: Eliminar una entidad de Orion

7. Conclusiones y Trabajo Futuro

7.1. Conclusiones

El RGPD no significa un giro de 180 grados a la hora de cómo se protegen los datos de usuario en las aplicaciones, con respecto a los dispuesto hasta ahora por la LOPD. Redefine y amplia los datos considerados sensibles y refuerza los derechos de los usuarios permitiéndole un mayor conocimiento y control sobre el uso que se le da a sus datos.

FIWARE proporciona muchos de los elementos básicos necesarios para crear una aplicación para *smart cities*. Es destacable señalar que, teniendo en cuenta lo relacionados que están los datos de usuarios con este tipo de aplicaciones, no exista un GE específico que permita adecuar al RGPD las aplicaciones basadas en FIWARE.

Orion no ofrece ninguna capacidad de autenticación por lo que la seguridad de sus datos debe ser manejada de forma externa y Orion debe ser accesible únicamente a través de estos elementos externos. El desarrollo del proyecto nos ha servido para ver de primera mano las dificultades de adaptar un sistema en producción a nuevas características no contempladas durante sus diseño, en este caso su adecuación a lo dispuesto en el RGPD. En especial el no poder modificar el formato de los datos guardados en Orion.

Me he dado cuenta durante el desarrollo de la aplicación de lo importante que es detallar desde el primer momento el alcance y funcionalidades del sistema a desarrollar. En mi caso, no definir estos elementos desde un principio, hizo que las funcionalidades fueran aumentando hasta llegar un punto en el que el desarrollo sería inviable en el tiempo dado y por lo tanto me vi obligado a reducir sus funcionalidades y definirlas claramente en los requisitos funcionales mostrados en el documento.

7.1.1. Objetivos alcanzados

- Se han analizado las diferencias entre la LOPD y el RGPD y cómo afectan a las empresas y usuarios.
- Se ha estudiado la estructura de FIWARE y su viabilidad a la hora de cumplir con los derechos recogidos en el RGPD mediante el desarrollo de una aplicación.
- Se ha desarrollado una aplicación que adapta a el RGPD a un sistema ya en producción.
- La aplicación desarrollada es fácilmente ampliable a nuevos tipo de entidades que se quieran introducir y a la creación de los permisos que derivarían de ello.
- Gracias el uso del *framework* Django, la aplicación desarrollada ofrece unas garantías mínimas de seguridad y estabilidad.

7.2. Trabajo Futuro

- Lograr la integración completa con Orion, permitiendo suscripciones y acceso a todos los métodos ofrecidos por este.
- Personalizar la interfaz de usuario para la versión navegable.

- Habilitar la notificación vía correo electrónico a los usuarios cuando sus cuentas van a ser borradas.
- Habilitar HTTPS para toda la aplicación.
- Eliminar de la versión navegable el formulario para el método POST del end-point */users/* ya que este método no esta permitido en esa vista.
- Tratar el problema de cuando en un método POST la URL no acaba en '/'. Cuando una petición llega sin '/' al final está se redirige automáticamente a la URL con la barra añadida. Surge un problema cuando se tratan de métodos POST ya que los datos no se redirigen y por tanto se pierden.
- Modificar la página inicial de la versión navegable para que muestre la página de login si no está identificado contra el sistema o los *end-points* accesibles para su rol.
- Preparar la aplicación para su despliegue en un entorno de producción.

Referencias

[1] Alvaro Alonso. «PEP Proxy - Wilma». En: (14 de nov. de 2017). URL: https://catalogue.fiware.org/enablers/pep-proxy-wilma.

[2] Joaquín Salvachúa y Álvaro Alonso. «Identity Management – KeyRock». En: (10 de nov. de 2017). URL: https://catalogue.fiware.org/enablers/identity-management-keyrock.

[3] UPM y Álvaro Alonso. «FIWARE Open specification, Identity management». En: (10 de nov. de 2017). URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.IdentityManagement.

[4] UPM y Álvaro Alonso. «FIWARE Open specification, PEP Proxy». En: (14 de nov. de 2017). URL: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.PEPProxy.

[5] Stephan Neuhaus (Zürcher Hochschule der Angewandten Wissenschaften). «FIWARE Open specification, Privacy». En: (14 de nov. de 2017). URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.Privacy.

[6] Cecilio Álvarez Caules. En: (5 de jul. de 2018). URL: https://www.arquitecturajava.com/arquitecturas-rest-y-sus-niveles/.

[7] Criteo.com. En: (1 de jul. de 2018). URL: https://www.criteo.com/es/insights/datos-sensibles-o-no-segun-el-rgpd-una-distincion-que-marca-la-diferencia/.

[8] THALES y Cyril Dangerville. «FIWARE Open specification, Authorization PDP». En: (13 de nov. de 2017). URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.AuthorizationPDP.

[9] Cyril Dangerville. «Authorization PDP - AuthZForce». En: (13 de nov. de 2017). URL: https://catalogue.fiware.org/enablers/authorization-pdp-authzforce.

[10] Expansión. «Las dudas más frecuentes de la adaptación al RGPD». En: (9 de jun. de 2018). URL: https://amp.expansion.com/juridico/actualidad-tendencias/2018/06/04/5b157bb022601da73f8b461b.html.

[11] Torres de Fenicia. En: (8 de jul. de 2018). URL: http://torresdefenicia.tripod.com/sitebuildercontent/sitebuilderfiles/MP%5C_2.06.2%5C_Formulario_Tabla%5C_Definicion%5C_Administracion%5C_riesgos.pdf.

[12] FIWARE. «FIWARE About us». En: (27 de dic. de 2017). URL: https://www.fiware.org/about-us/.

[13] Django Software Foundation. En: (5 de jul. de 2018). URL: https://docs.djangoproject.com/en/2.0/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names.

[14] Juanjo Hierro. «Objetivo previsto de una plataforma de smart city». En: (19 de jun. de 2018). URL: https://es.slideshare.net/JuanjoHierro/fiware-a-standard-platform-for-smart-cities.

[15] Patricia Lalanda. «Análisis de las novedades introducidas por el RGPD Loyra Abogados». En: (6 de nov. de 2017). URL: https://www.loyra.com/reglamento-general-de-proteccion-de-datos-analisis-sobre-las-novedades-que-impone-a-las-empresas/.

[16] LegalToday. En: (3 de jul. de 2018). URL: http://www.legaltoday.com/blogs/transversal/blog-ilp-abogados/plazos-para-conservacion-y-cancelacion-de-datos-personales.

[17] Stephan Neuhaus. «Privacy». En: (14 de nov. de 2017). URL: https://catalogue.fiware.org/enablers/privacy.

[18] Blog de Protección de Datos para empresas y autónomos. En: (1 de jul. de 2018). URL: https://protecciondatos-lopd.com/empresas/datos-especialmente-protegidos-sensibles/.

[19] Blog de Protección de Datos para empresas y autónomos. En: (3 de jul. de 2018). URL: https://protecciondatos-lopd.com/empresas/conservarcion-datos-plazo/#Plazos_de_conservacion.

[20] Blog de Protección de Datos para empresas y autónomos. «Derechos ARCO ¿Qué son? Nuevos derechos del RGPD». En: (16 de jun. de 2018). URL: https://protecciondatos-lopd.com/empresas/derechos-arco-que-son/.

[21] Peter Salhofer. «Evaluating the FIWARE Platform». En: *51st Hawaii International Conference on System Sciences, HICSS 2018, Hilton Waikoloa Village, Hawaii, USA, January 3-6, 2018*. 2018.

A. Manual de Usuario y programador de Mímir

A.1. Introducción

El presente manual adopta un enfoque práctico basado en ejemplos para que los usuarios logren familiarizarse con Mímir de un modo rápido y ameno.

Este documento está divido para los dos roles de usuarios admitidos por Mímir: las empresas y sus aplicaciones.

A.2. Manual para la empresa

A.2.1. Antes de empezar

Como la gestión de las cuentas de empresa es realizada comúnmente por personas, Mímir implementa una API navegable para que las operaciones listadas en este manual puedan ser realizadas de una forma sencilla. Por ello se proporcionará tanto la vista de navegador como los comandos para realizar todas las operaciones.

Para poder hacer uso de Mímir es necesario tener una cuenta en el sistema. Esta cuenta le será proporcionada por el ayuntamiento tras concretar su contrato con dicha institución.

Los datos de usuario le serán proporcionados junto con la dirección de acceso a Mímir tras la firma del contrato.

Para los ejemplos de este manual supondremos que la url de acceso es localhost en el puerto 8000, el usuario Business1 y la contraseña qazwsxed

Mímir utiliza JSON tanto para los datos en las solicitudes como para las respuestas.

ATENCIÓN!: Recuerde que todas las urls deben terminar en '/'

A.2.2. Identificarse en Mímir

Desde la página inicial de Mímir haga click en el botón login ubicado arriba a la derecha, tras lo cual será redirigido a la página de login donde deberá meter sus credenciales. Una vez hecho esto ya estará identificado en el sistema.

Mediante comando la única forma de loguearse en el sistema es mediante un token de acceso. Para obtenerlo ejecute el siguiente comando:

```
1    http POST localhost:8000/api-token-auth/ username=Business1 password=qazwsxed
```

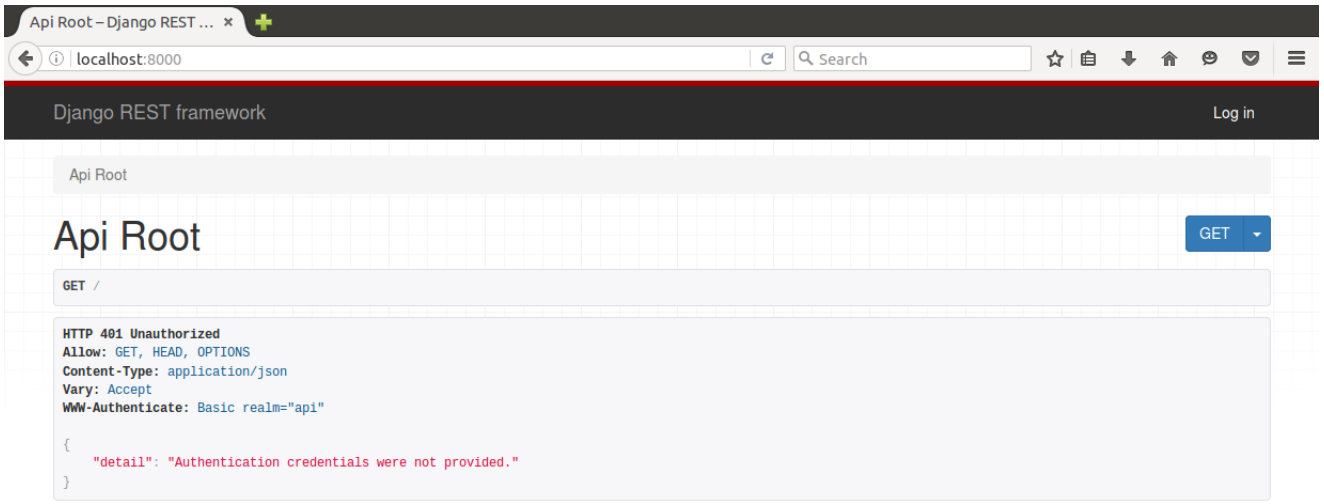

Figura 21: Página inicial

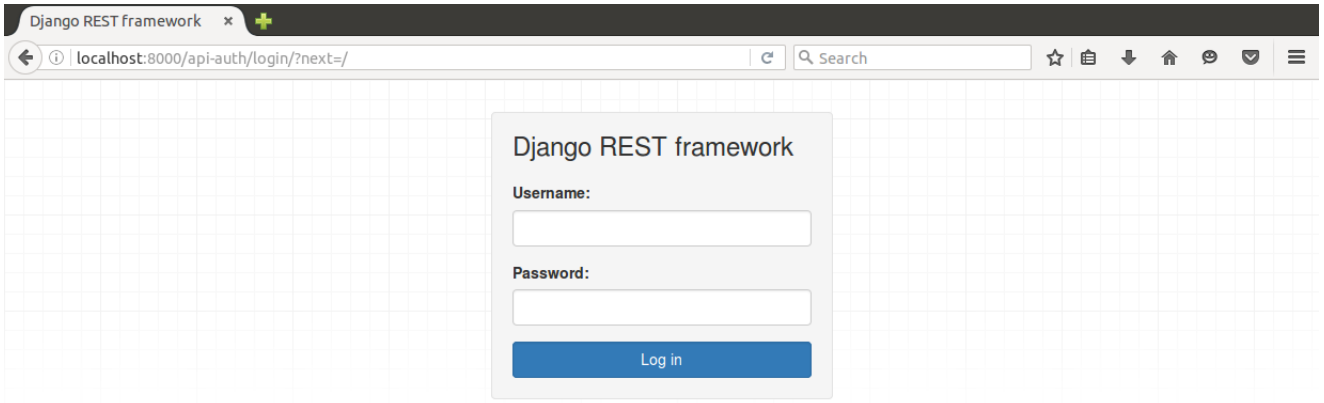

Figura 22: Página de login

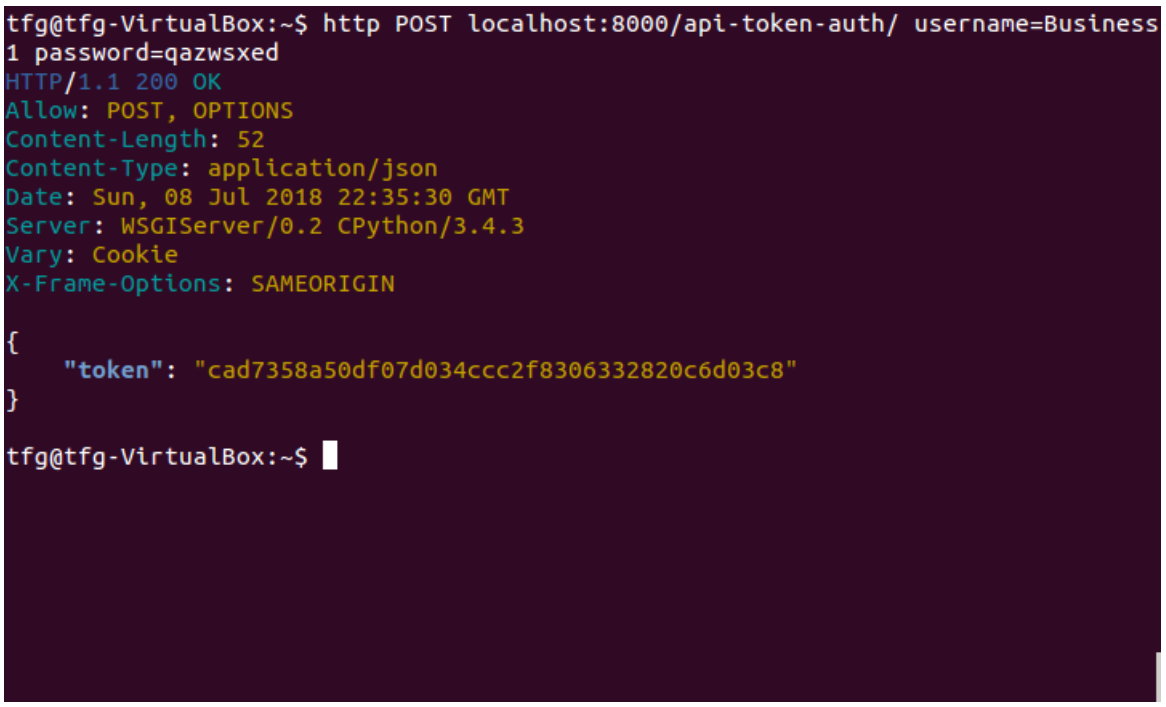

Figura 23: Obtener token de acceso

Como respuesta recibirá el token de acceso que deberá pasar en el resto de peticiones a Mímir

A.2.3. Crear una nueva aplicación

Diríjase a la página ***localhost:8000/apps/*** en ella se listarán todas sus aplicaciones y podrá crear nuevas.

Rellene los datos del formulario poniendo especial atención en que los permisos escogidos deben ser compatibles con los acordados con el ayuntamiento. Para este ejemplo escogeremos los permisos createPerson que nos permite crear nuevas entidades de tipo persona, updatePerson para modificar sus datos, deletePerson para borrar la entidad y givenNamePerson para acceder al atributo nombre de la entidad.

El comando equivalente es el siguiente:

```
1  http POST localhost:8000/apps/
2  "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
3  username=App1 description="App 1" password=qazwsxed
4  permissions:='["createPerson","deletePerson","updatePerson","namePerson"]'
```

Si la aplicación se crea correctamente recibirá una respuesta con código 201 y los datos de la aplicación creada. Una vez creada una aplicación deberá esperar a que le den de paso los permisos escogidos y le activen la cuenta de la aplicación.

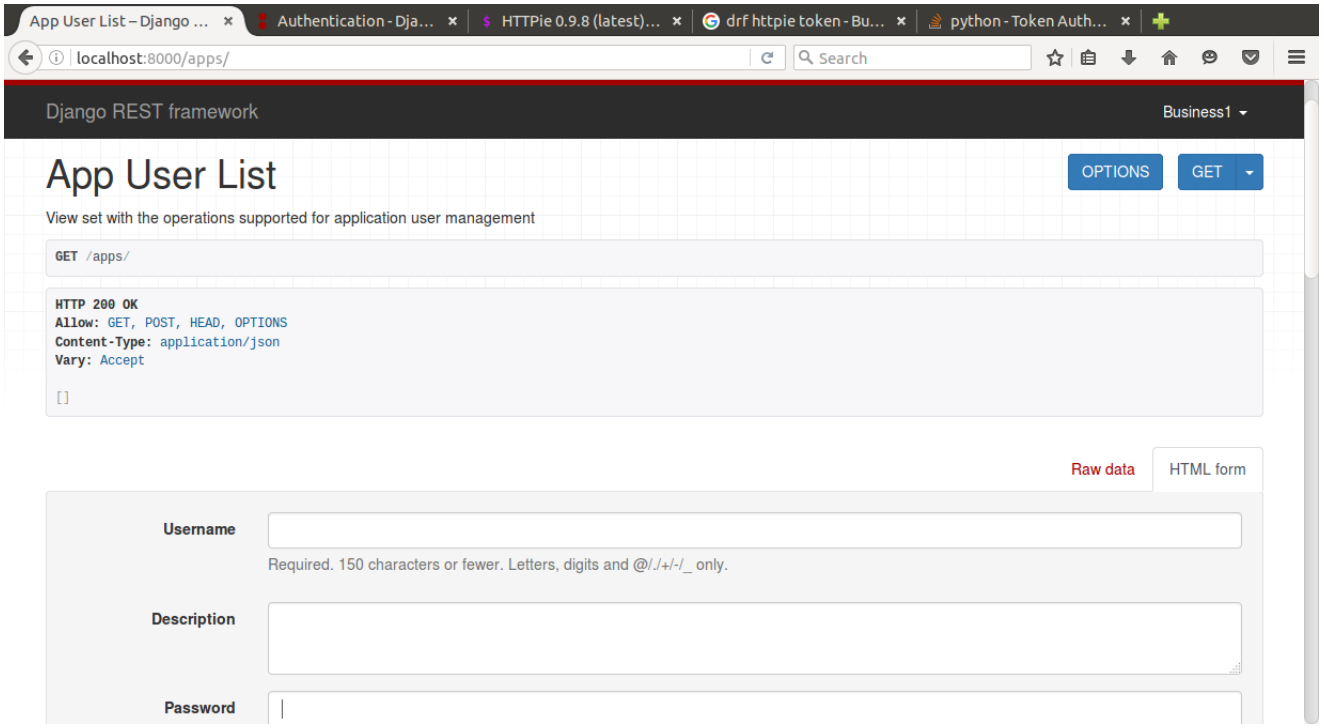

Figura 24: Pagina de listado y creación de aplicaciones

```
tfg@tfg-VirtualBox:~$ http POST localhost:8000/apps/ "Authorization: Token cad73
58a50df07d034ccc2f8306332820c6d03c8" username=App1 description="App 1" password=
qazwsxed permissions:='["createPerson","deletePerson","updatePerson","namePerson
"]'
HTTP/1.1 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 242
Content-Type: application/json
Date: Sun, 08 Jul 2018 22:59:29 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "active": false,
    "description": "App 1",
    "owner": "Business1",
    "password": "pbkdf2_sha256$100000$28US1yko123q$MwztFNWdMx0ujoQ2qDxGawV8s6Enz
einODCSEpjj1RI=",
    "permissions": [
        "createPerson",
        "deletePerson",
        "updatePerson",
        "namePerson"
    ],
    "username": "App1"
}

tfg@tfg-VirtualBox:~$
```

Figura 25: Crear nueva aplicación

A.2.4. Modificar y ver detalles de la aplicación

Diríjase a la página ***localhost:8000/apps/App1*** para ver los detalles de la aplicación. Desde esta página podrá además modificar los datos de la aplicación. Tenga en cuenta que una vez modificada la cuenta de la aplicación volverá al estado inactivo tendrá que ser activada de nuevo.

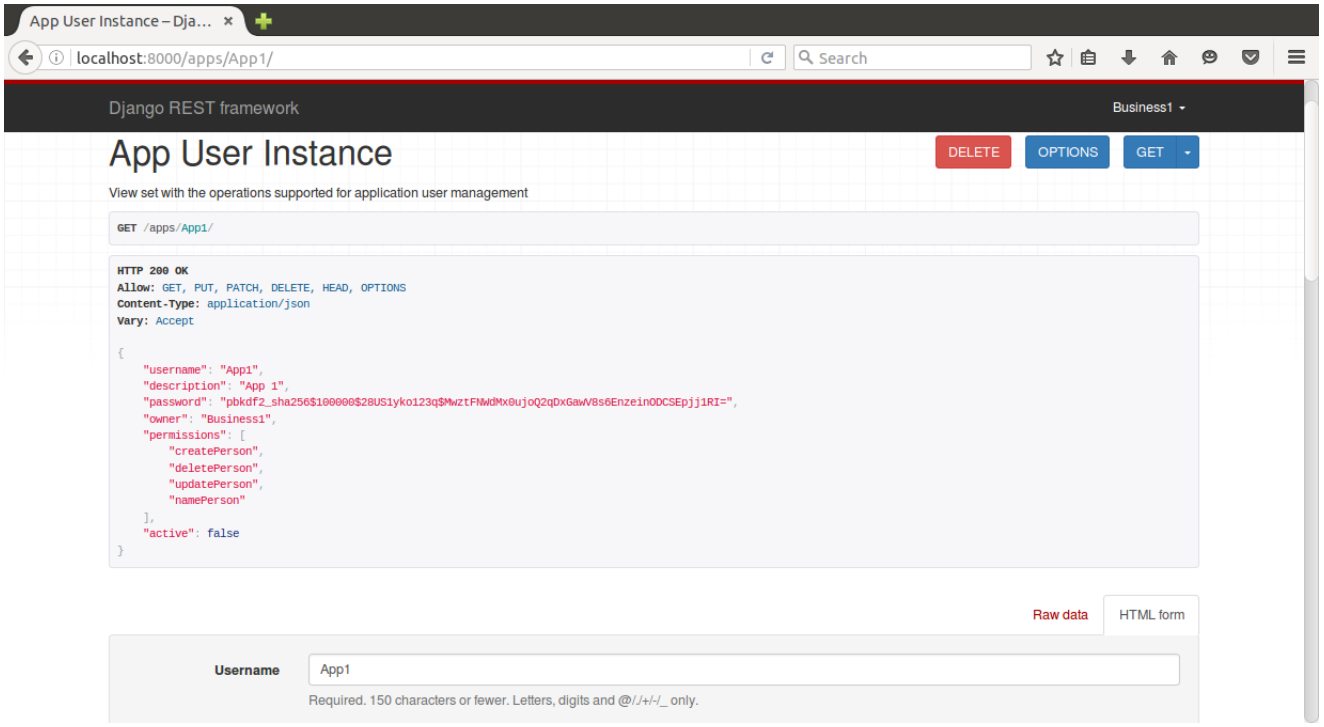

Figura 26: Pagina detalles y actualización de aplicaciones

Los comandos equivalentes son los siguientes:
Para ver detalles

```
1  http GET localhost:8000/apps/App1/
2  "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
```

Este comando devolverá una respuesta con código 200 y los datos de la aplicación
Para modificar

```
1  http PUT localhost:8000/apps/App1/
2  "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
3  username=App1 description="App 1 modificada" password=qazwsxed
4  permissions:='["createPerson","deletePerson","updatePerson","namePerson"]'
```

Si la aplicación se modifica correctamente recibirá una respuesta con código 200 y los nuevos datos de la aplicación.

```
tfg@tfg-VirtualBox:~$ http GET localhost:8000/apps/App1/ "Authorization: Token c
ad7358a50df07d034ccc2f8306332820c6d03c8"
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 253
Content-Type: application/json
Date: Sun, 08 Jul 2018 23:45:48 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "active": false,
    "description": "App 1 modificada",
    "owner": "Business1",
    "password": "pbkdf2_sha256$100000$P1uolG5O9CqI$Q1FdxCFUbOpcpcl5PyINn8tqOj6li
4QOxmnujjw2kmM=",
    "permissions": [
        "createPerson",
        "deletePerson",
        "updatePerson",
        "namePerson"
    ],
    "username": "App1"
}

tfg@tfg-VirtualBox:~$
```

Figura 27: Ver detalles de aplicación

```
tfg@tfg-VirtualBox:~$ http PUT localhost:8000/apps/App1/ "Authorization: Token c
ad7358a50df07d034ccc2f8306332820c6d03c8" username=App1 description="App 1 modifi
cada" password=qazwsxed permissions:='["createPerson","deletePerson","updatePers
on","namePerson"]'
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 253
Content-Type: application/json
Date: Sun, 08 Jul 2018 23:31:16 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "active": false,
    "description": "App 1 modificada",
    "owner": "Business1",
    "password": "pbkdf2_sha256$100000$P1uolG5O9CqI$Q1FdxCFUbOpcpcl5PyINn8tqOj6li
4QOxmnujjw2kmM=",
    "permissions": [
        "createPerson",
        "deletePerson",
        "updatePerson",
        "namePerson"
    ],
    "username": "App1"
}

tfg@tfg-VirtualBox:~$
```

Figura 28: Modificar aplicación

A.2.5. Borrar aplicación

Desde la página ***localhost:8000/apps/App1*** también podrá borrar la cuenta de la aplicación. Para ello únicamente deberá pulsar en el botón rojo de la izquierda y confirmar en la ventana emergente.

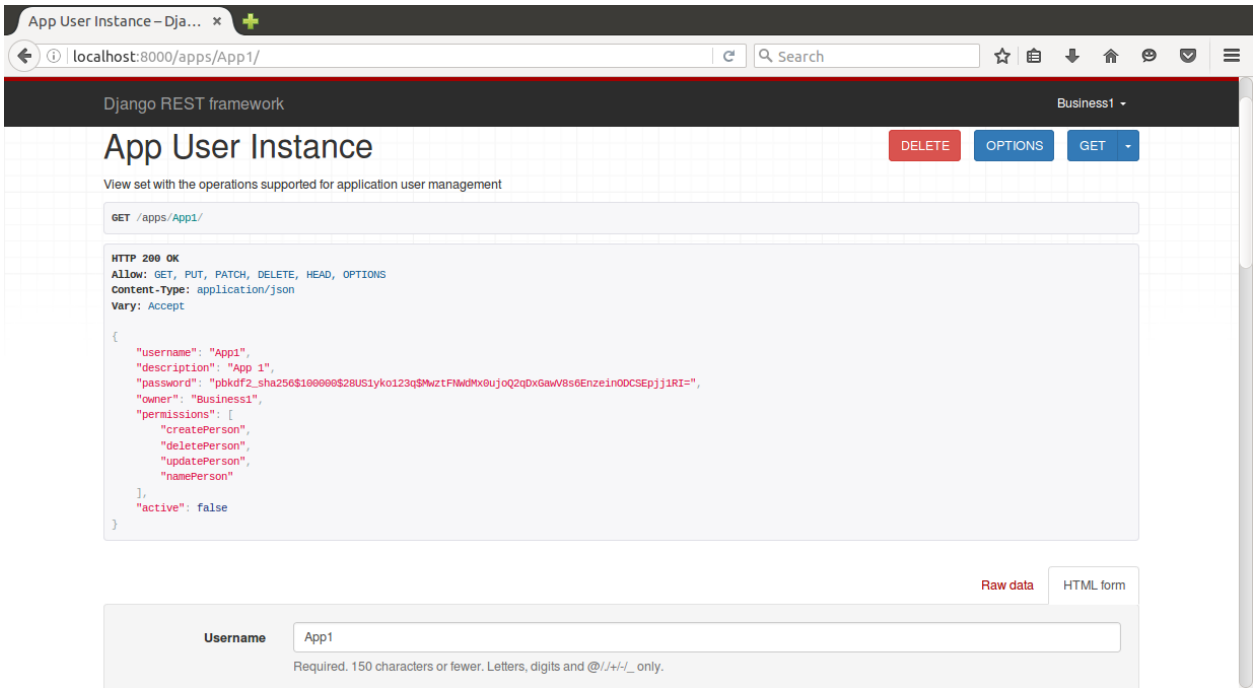

Figura 29: Borrar aplicacion

El comando equivalente es el siguiente:

```
1  http DELETE localhost:8000/apps/App1/
2  "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
```

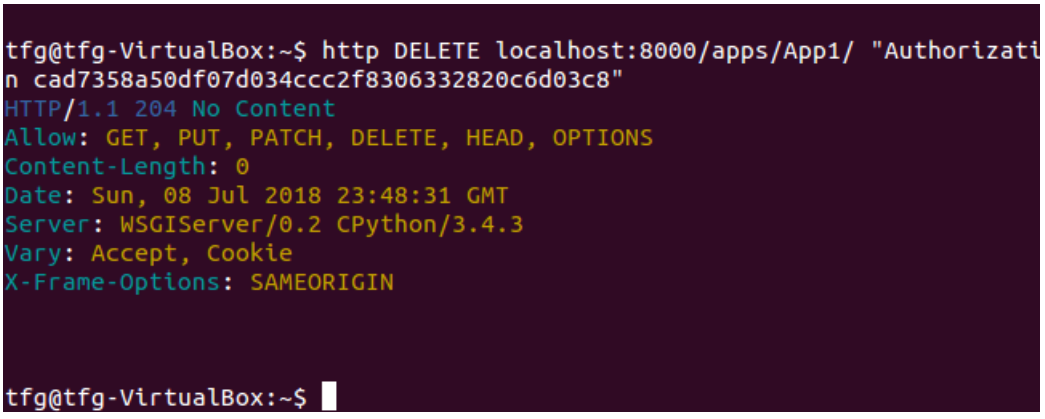

Figura 30: Comando para borrar aplicación

Si la aplicación se borra correctamente recibirá una respuesta con código 204.

A.2.6. Ver y modificar detalles de tu cuenta

Diríjase a la página ***localhost:8000/users/Business1*** para ver los detalles de su cuenta. Desde esta página podrá además modificar sus datos.

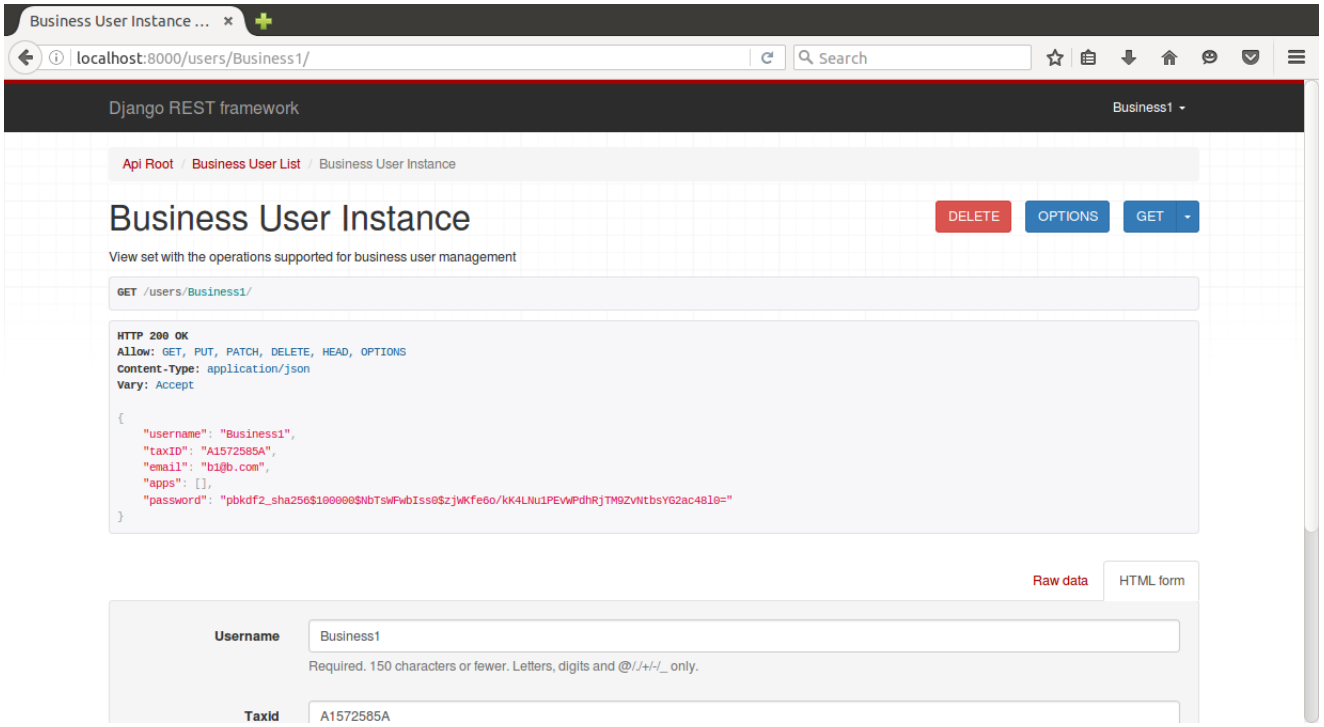

Figura 31: Ver y modificar tu cuenta

Los comandos equivalentes son los siguientes:
Para ver detalles

```
1    http GET localhost:8000/users/Business1/
2    "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
```

Este comando devolverá una respuesta con código 200 y los datos de tu cuenta
Para modificar

```
1    http PUT localhost:8000/users/Business1/
2    "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
3    username=Business1 email=modificado@mod.com password=qazwsxed taxID=B1234567B
```

Si su cuenta se modifica correctamente recibirá una respuesta con código 200 y los nuevos datos de la aplicación.

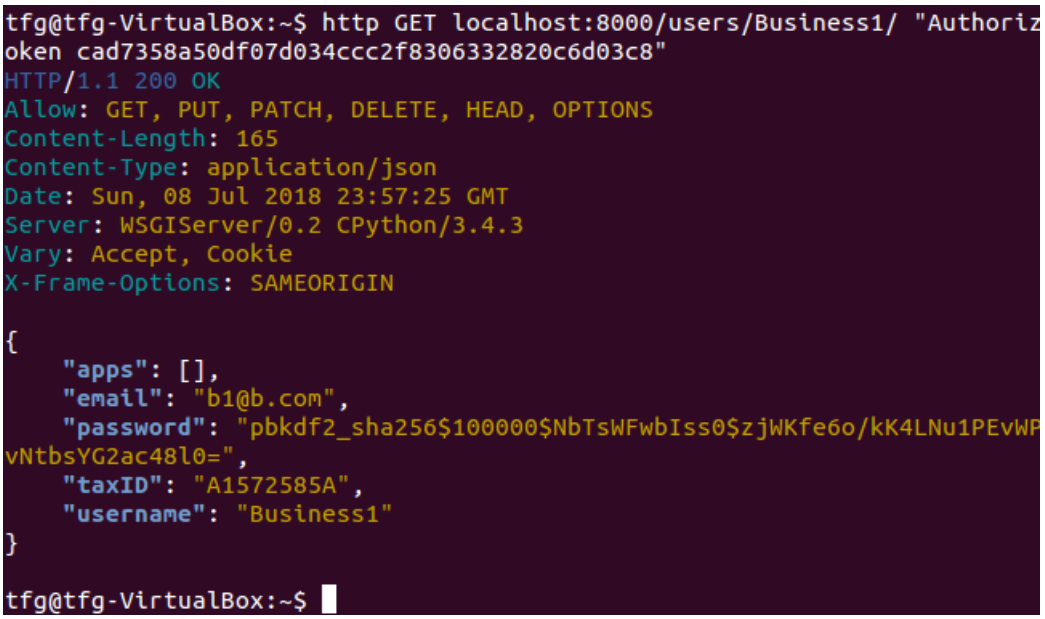

Figura 32: Ver detalles de tu cuenta

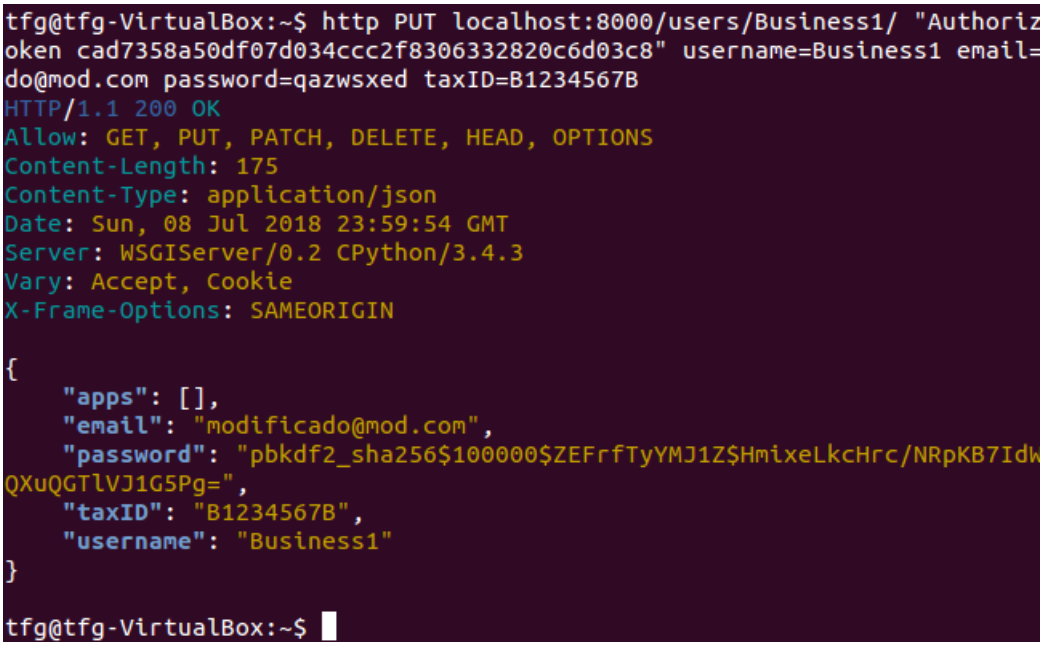

Figura 33: Modificar datos de tu cuenta

A.2.7. Borrar cuenta

Desde la página ***localhost:8000/users/Business1*** también podrá borrar su cuenta de usuario. Para ello únicamente deberá pulsar en el botón rojo de la izquierda y confirmar en la ventana emergente.

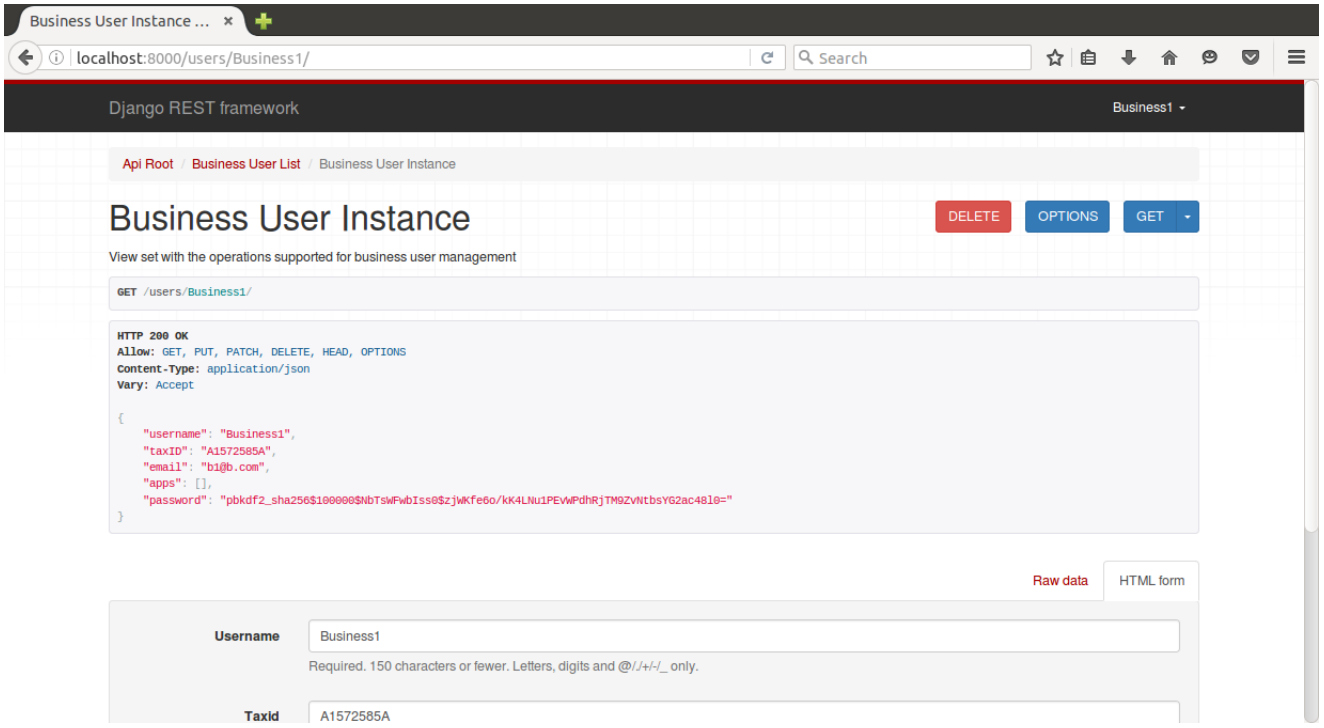

Figura 34: Borrar tu cuenta

El comando equivalente es el siguiente:

```
1  http DELETE localhost:8000/users/Business1/
2  "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
```

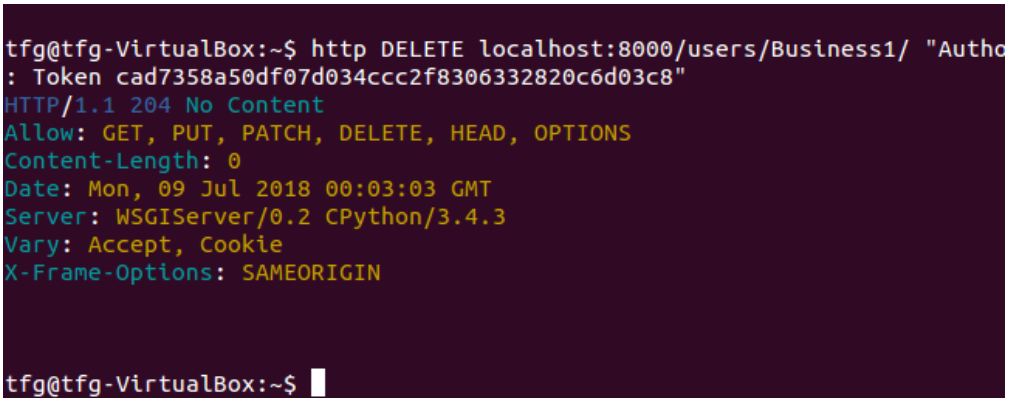

Figura 35: Borrar tu cuenta

Si la cuanta de usuario se borra correctamente recibirá una respuesta con código 204.

A.3. Manual de uso de la cuenta aplicación

A.3.1. Antes de empezar

Para poder seguir los pasos detallados en este manual es necesario tener una cuenta de aplicación activada en el sistema.

El usuario objetivo de esta sección del manual es el programador de la aplicación que hará uso de esta cuenta, por lo tanto en estos ejemplos se presentarán únicamente las acciones via comando.

Para los ejemplos de este manual supondremos que la url de acceso es localhost en el puerto 8000, el usuario App1 y la contraseña qazwsxed

App1 tendrá los permisos necesarios para crear actualizar y borrar entidades de tipo persona así como el permiso para acceder al atributo *name* de las entidades de tipo persona. Mímir utiliza JSON tanto para los datos en las solicitudes como para las respuestas.

ATENCIÓN!: Recuerde que todas las urls deben terminar en '/'

A.3.2. Obtener token de acceso

Mímir utiliza como modo de autenticación via comando tokens de acceso. Para obtener un token de acceso ejecute el siguiente comando.

```
1    http POST localhost:8000/api-token-auth/ username=App1 password=qazwsxed
```

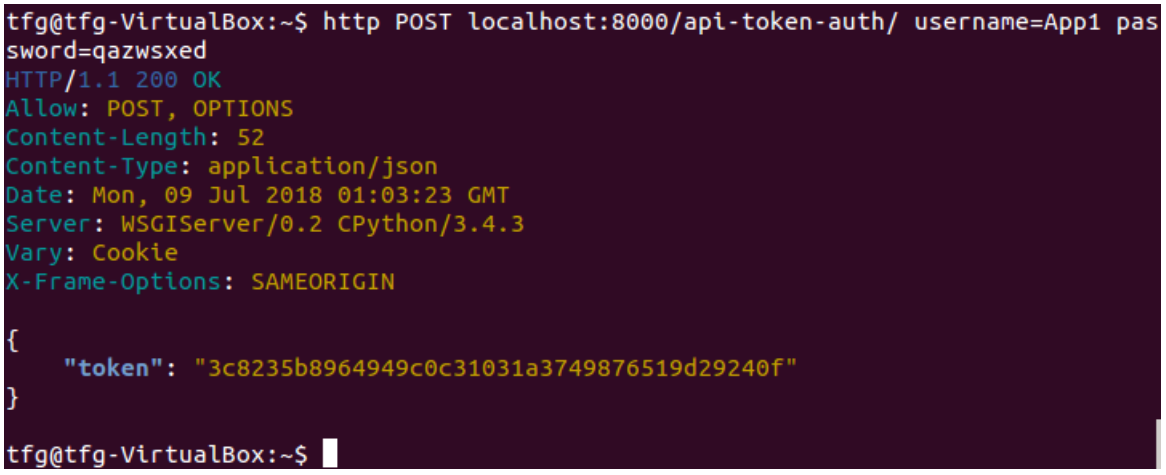

Figura 36: Obtener token de acceso

Como respuesta recibirá el token de acceso que deberá pasar en el resto de peticiones a Mímir.

A.3.3. Crear nueva entidad

Para crear una nueva entidad utilice el siguiente comando con el mismo formato para los datos que el mostrado.

```
http POST localhost:8000/v2/entities/
"Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f"
id=12345678D type=Person
givenName:='{"value":"TEST","type":"String"}'
familyName:='{"value":"TEST TEST","type":"String"}'
birthDate:='{"value":"1999-02-15","type":"String"}'
birthPlace:='{"value":"Valladolid, Valladolid","type":"String"}'
address:='{"value":"TEST","type":"String"}'
gender:='{"value":"Masculino","type":"String"}'
nationality:='{"value":"española","type":"String"}'
parkingCard:='{"value":0,"type":"Integer"}'
```

Figura 37: Crear nueva entidad

Si la aplicación se crea correctamente recibirá una respuesta con código 201 y los datos de la entidad creada.

A.3.4. Listar entidades

Para listar todas las entidades de un determinado tipo, utilice el siguiente comando. Utilice type para seleccionar el tipo de las entidades a listar. No es posible listar entidades de diversos tipos a la vez.

```
1  http GET localhost:8000/v2/entities/
2  "Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f"
3  type==Person
```

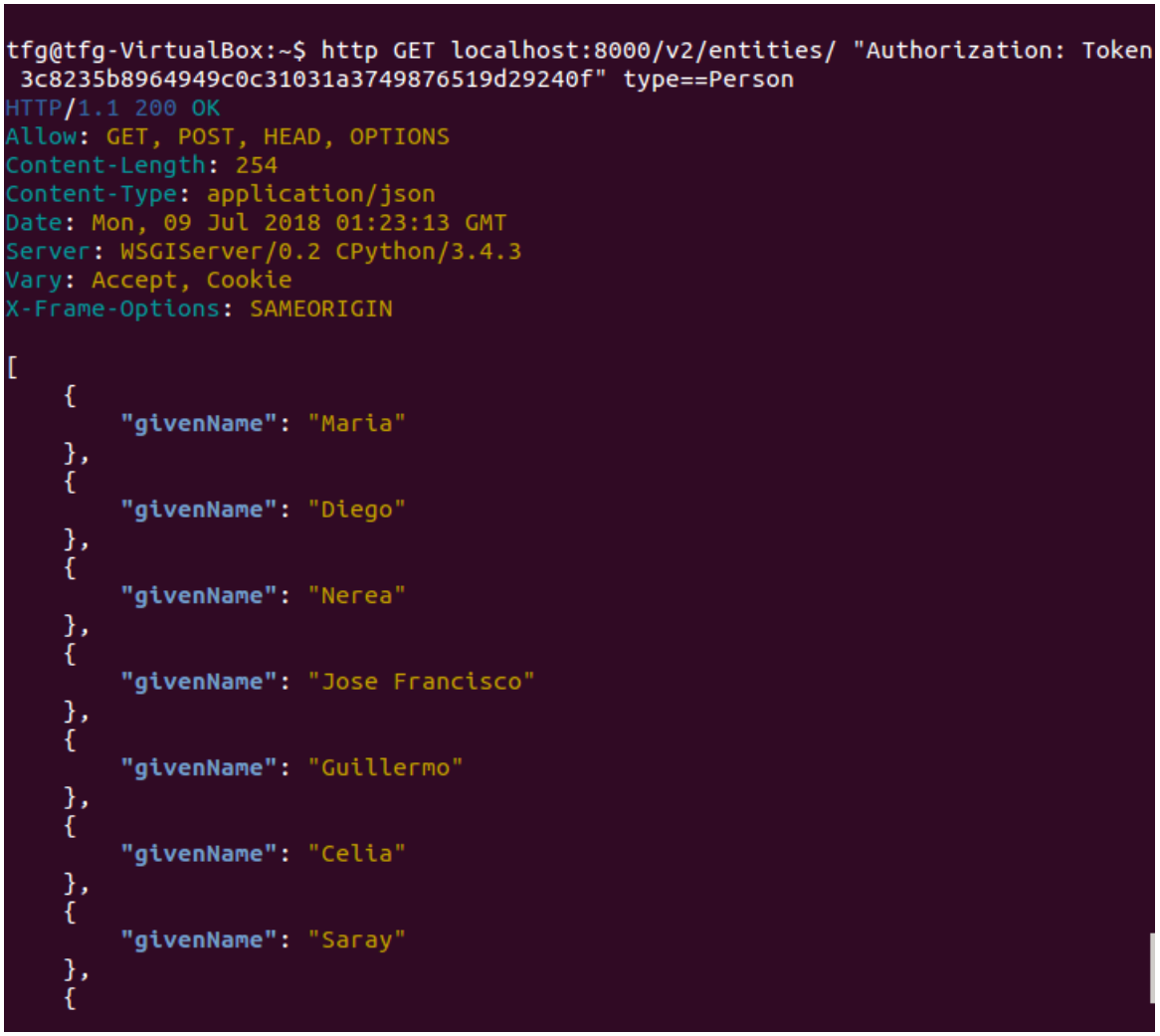

Figura 38: Listar entidades de un determinado tipo

Recibirá una respuesta con código 200 y los datos de la entidad a los que tenga acceso. Como podemos ver, App1 recibe únicamente el campo ***givenName*** ya que es el único al que tiene permitido el acceso

A.3.5. Acceder a un atributo de una entidad

Para acceder a un atributo de una entidad, utilice el siguiente comando. Es obligatorio el uso de type para seleccionar el tipo de la entidad. Opcionalmente puede usar options=keyValues para mostrar únicamente el valor del atributo.

```
1    http GET localhost:8000/v2/entities/12345678D/attrs/givenName
2    "Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f"
3    type==Person
```

Figura 39: Acceder atributo de una entidad

Si el atributo existe dentro de la entidad y tienes permiso para verlo recibirá una respuesta con código 200 y el atributo.

A.3.6. Obtener datos de una entidad

Para acceder a los datos una única entidad, utilice el siguiente comando. Es obligatorio el uso de type para seleccionar el tipo de la entidad.

```
1  http GET localhost:8000/v2/entities/12345678D/
2  "Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f"
3  type==Person
```

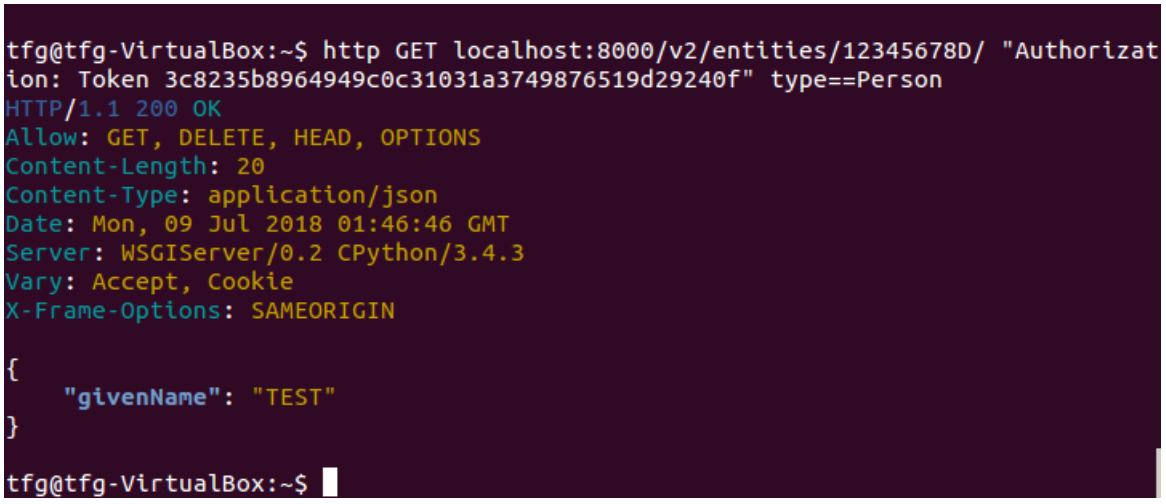

Figura 40: Obtener datos de una entidad

Recibirá una respuesta con código 200 y los atributos de la entidad a los que tenga acceso.

A.3.7. Modificar un campo de una entidad

Para modificar un campo de una entidad, utilice el siguiente comando. Es obligatorio el uso de type para seleccionar el tipo de la entidad.

```
1  http PUT localhost:8000/v2/entities/12345678D/attrs/givenName/
2  "Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f"
3  type==Person
4  givenName:='{"value":"Mod","type":"String"}'
```

Recibirá una respuesta con código 204 si el campo se ha modificado satisfactoriamente.

```
tfg@tfg-VirtualBox:~$ http PUT localhost:8000/v2/entities/12345678D/attrs/g
ame/ "Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f" type==
n givenName:='{"value":"Mod","type":"String"}'
HTTP/1.1 204 No Content
Allow: GET, PUT, HEAD, OPTIONS
Content-Length: 0
Date: Mon, 09 Jul 2018 01:56:41 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN


tfg@tfg-VirtualBox:~$
```

Figura 41: Modificar datos de una entidad

A.3.8. Borrar entidad

Para borrar una entidad existente, utilice el siguiente comando. Es obligatorio el uso de type para seleccionar el tipo de la entidad.

```
1    http DELETE localhost:8000/v2/entities/12345678D/
2    "Authorization: Token 3c8235b8964949c0c31031a3749876519d29240f"
3    type==Person
```

```
tfg@tfg-VirtualBox:~$ http DELETE localhost:8000/v2/entities/12345678D/ "Au
zation: Token 3c8235b8964949c0c31031a3749876519d29240f" type==Person
HTTP/1.1 204 No Content
Allow: GET, DELETE, HEAD, OPTIONS
Content-Length: 0
Date: Mon, 09 Jul 2018 02:00:16 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN


tfg@tfg-VirtualBox:~$
```

Figura 42: Borrar una entidad

Recibirá una respuesta con código 204 si la entidad se ha borrado satisfactoriamente.

B. Manual de administración de Mímir

B.1. Introducción

Como administrador tendrá acceso a todas las vistas para la gestión de cuentas de usuario. Sin embargo, se recomienda usar siempre las vistas de administrador ya que proporcionan funcionalidades adicionales con respecto a las vistas normales.

En este manual se proporcionarán tanto las vistas de navegador como los comandos para realizar todas las operaciones.

Para los siguientes ejemplos se supondrá que la url de acceso es localhost en el puerto 8000 el nombre de usuario es admin y la contraseña qazwsxed

ATENCIÓN!: Recuerde que todas las urls deben terminar en '/'

B.2. Identificarse en Mímir

Para loguearse en Mímir deberá introducir sus credenciales desde la página de login si usa la versión navegable.

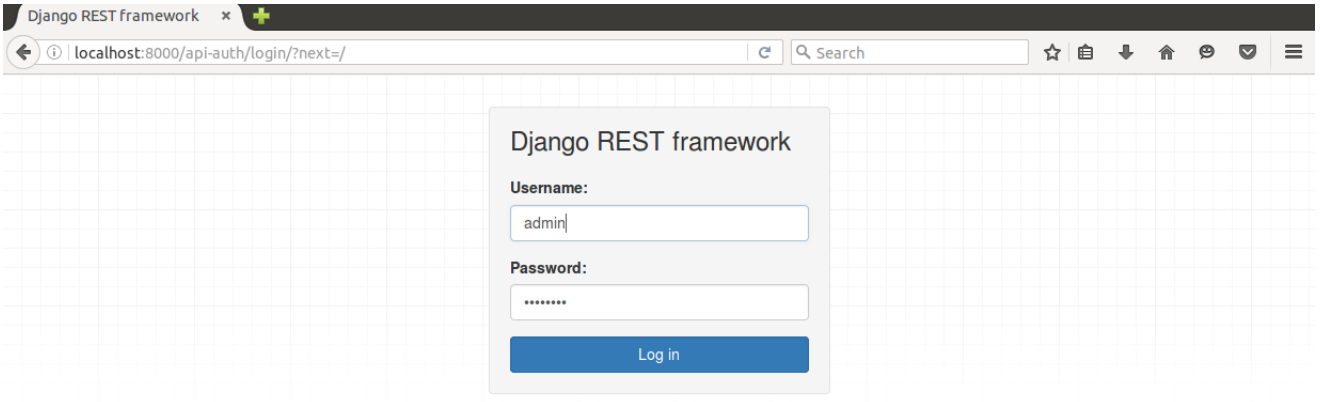

Figura 43: Página de login

Mediante comando la única forma de loguearse en el sistema es mediante un token de acceso. Para obtenerlo ejecute el siguiente comando:

```
1    http POST localhost:8000/api-token-auth/ username=admin password=qazwsxed
```

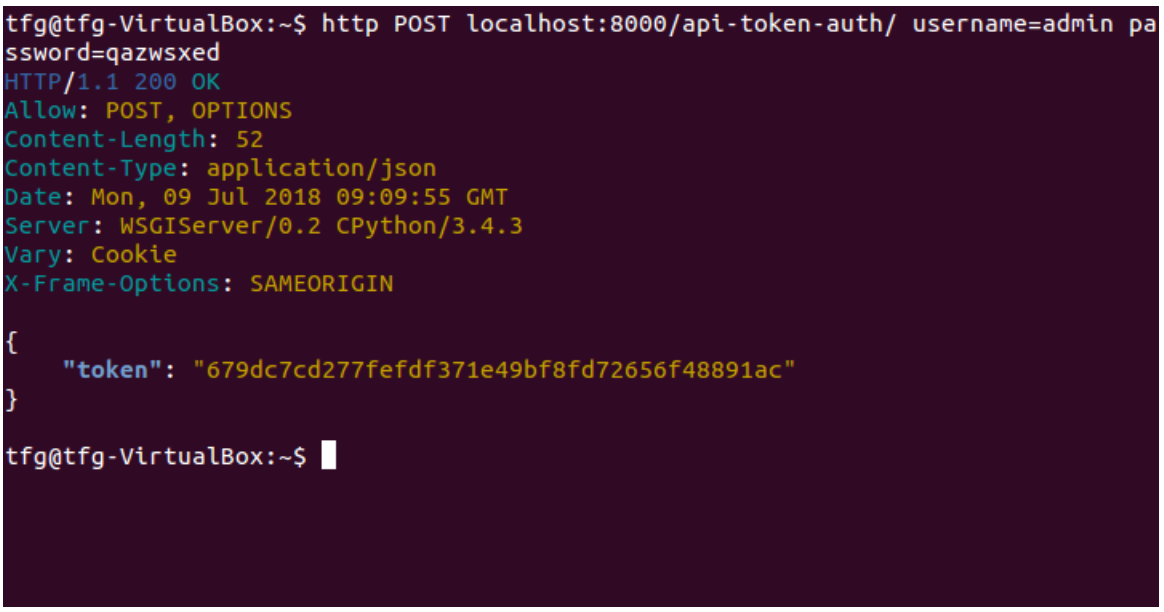

Figura 44: Obtener token de acceso (admin)

Como respuesta recibirá el token de acceso que deberá pasar en el resto de peticiones a Mímir

B.3. Crear empresa

Puede crear nuevas cuentas de empresa desde la página ***localhost:8000/user-admin/*** en ella se listarán las cuentas de empresas y podrá crear nuevas.

En nuestro caso ya tenemos una cuenta llamada Business1 en el sistema.

El comando equivalente es el siguiente:

```
1  http POST localhost:8000/user-admin/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
3  username=Manual taxID=A1234567A
4  email=b1@b.com password=qazwsxed
```

Si la cuenta de empresa se crea correctamente recibirá una respuesta con código 201 y los datos de la cuenta creada.

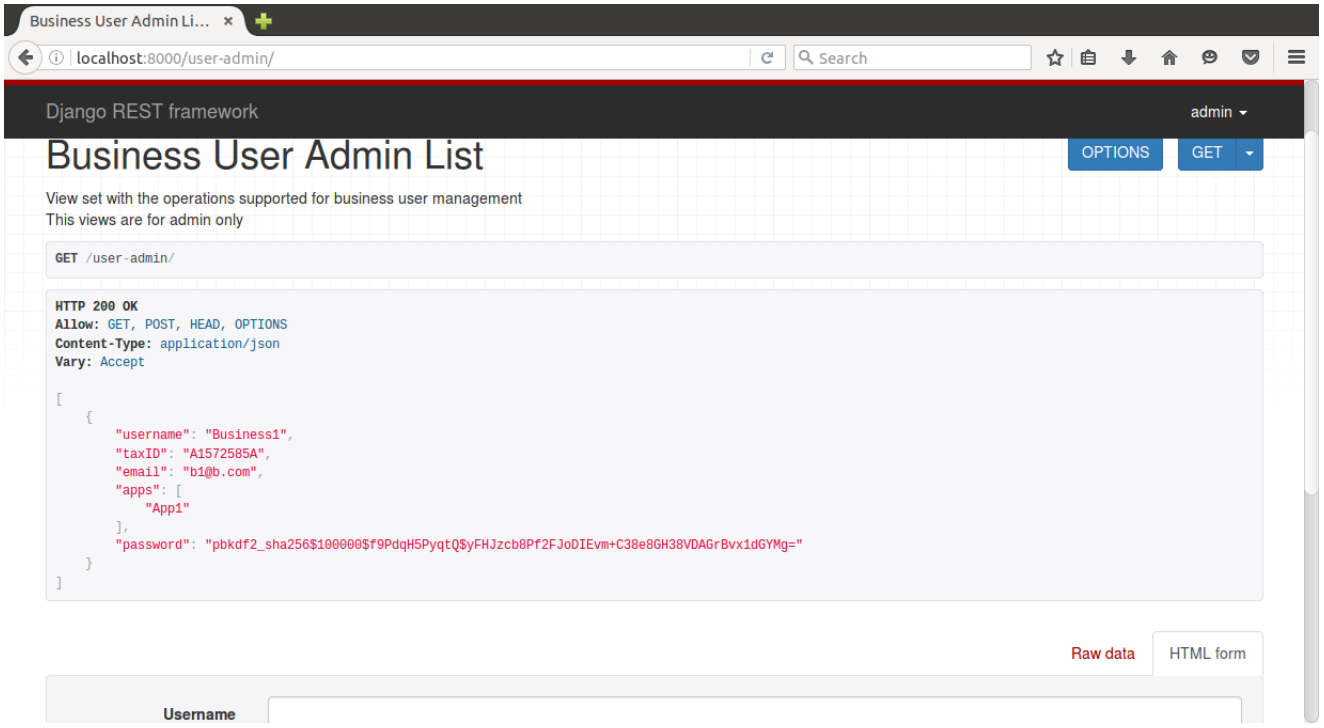

Figura 45: Pagina de listado y creación de cuentas de empresa para admin

```
tfg@tfg-VirtualBox:~$ http POST localhost:8000/user-admin/ "Authorization: Token
 679dc7cd277fefdf371e49bf8fd72656f48891ac" username=Manual taxID=A1234567A email
=manual@m.com password=qazwsxed
HTTP/1.1 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 166
Content-Type: application/json
Date: Mon, 09 Jul 2018 09:29:29 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "apps": [],
    "email": "manual@m.com",
    "password": "pbkdf2_sha256$100000$8xNFkGnar54F$+asvR9+1oZK1RdaKruNizBoLQkHox
zKIaQK3dFgBrKM=",
    "taxID": "A1234567A",
    "username": "Manual"
}

tfg@tfg-VirtualBox:~$
```

Figura 46: Crear nueva cuenta empresa

B.4. Modificar y ver detalles de la empresa

Desde la página ***localhost:8000/user-admin/Manual/*** puede ver los detalles de la empresa. Desde esta página podrá además modificar sus datos.

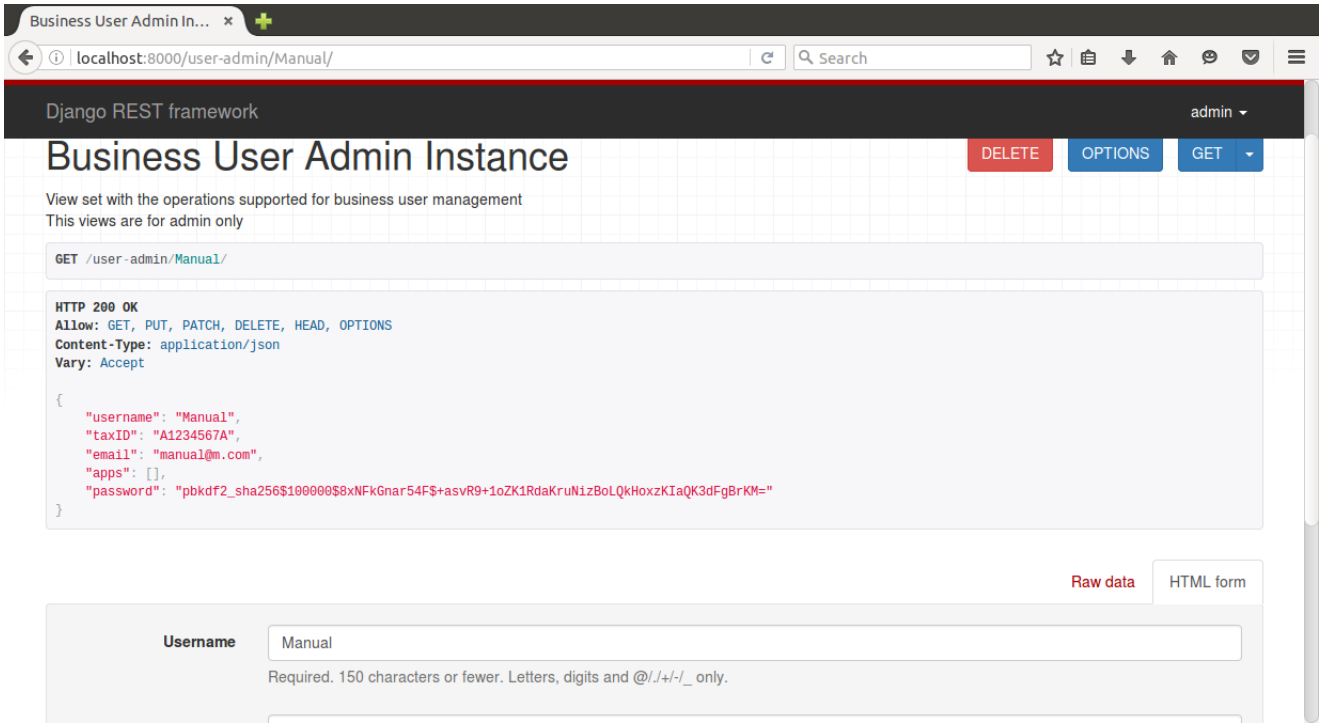

Figura 47: Pagina detalles y actualización de empresas para admin

Los comandos equivalentes son los siguientes:
Para ver detalles

```
1  http GET localhost:8000/user-admin/Manual/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
```

Este comando devolverá una respuesta con código 200 y los datos de la empresa
Para modificar

```
1  http PUT localhost:8000/apps/App1/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
3  username=Manual  taxID=B9876543B
4  email=manual@mod.com password=qazwsxed
```

Si la cuenta se modifica correctamente recibirá una respuesta con código 200 y los nuevos datos de la cuenta.

```
tfg@tfg-VirtualBox:~$ http GET localhost:8000/user-admin/Manual/ "Authorization:
 Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 166
Content-Type: application/json
Date: Mon, 09 Jul 2018 09:44:24 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "apps": [],
    "email": "manual@m.com",
    "password": "pbkdf2_sha256$100000$8xNFkGnar54F$+asvR9+1oZK1RdaKruNizBoLQkHox
zKIaQK3dFgBrKM=",
    "taxID": "A1234567A",
    "username": "Manual"
}

tfg@tfg-VirtualBox:~$
```

Figura 48: Ver detalles de la empresa

```
tfg@tfg-VirtualBox:~$ http PUT localhost:8000/user-admin/Manual/ "Authorization:
 Token 679dc7cd277fefdf371e49bf8fd72656f48891ac" username=Manual taxID=B9876543B
 email=manual@mod.com password=qazwsxed
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 168
Content-Type: application/json
Date: Mon, 09 Jul 2018 09:49:19 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "apps": [],
    "email": "manual@mod.com",
    "password": "pbkdf2_sha256$100000$a7qJXksNh7WK$3vnT/wsNy3JIaTT0OMM2FsedZyaYb
xFBj5UlnHHcJ1U=",
    "taxID": "B9876543B",
    "username": "Manual"
}

tfg@tfg-VirtualBox:~$
```

Figura 49: Modificar cuenta empresa

B.5. Borrar empresa

Desde la página ***localhost:8000/user-admin/Manual/*** también podrá borrar la cuenta de la empresa. Para ello únicamente deberá pulsar en el botón ***DELETE*** y confirmar.

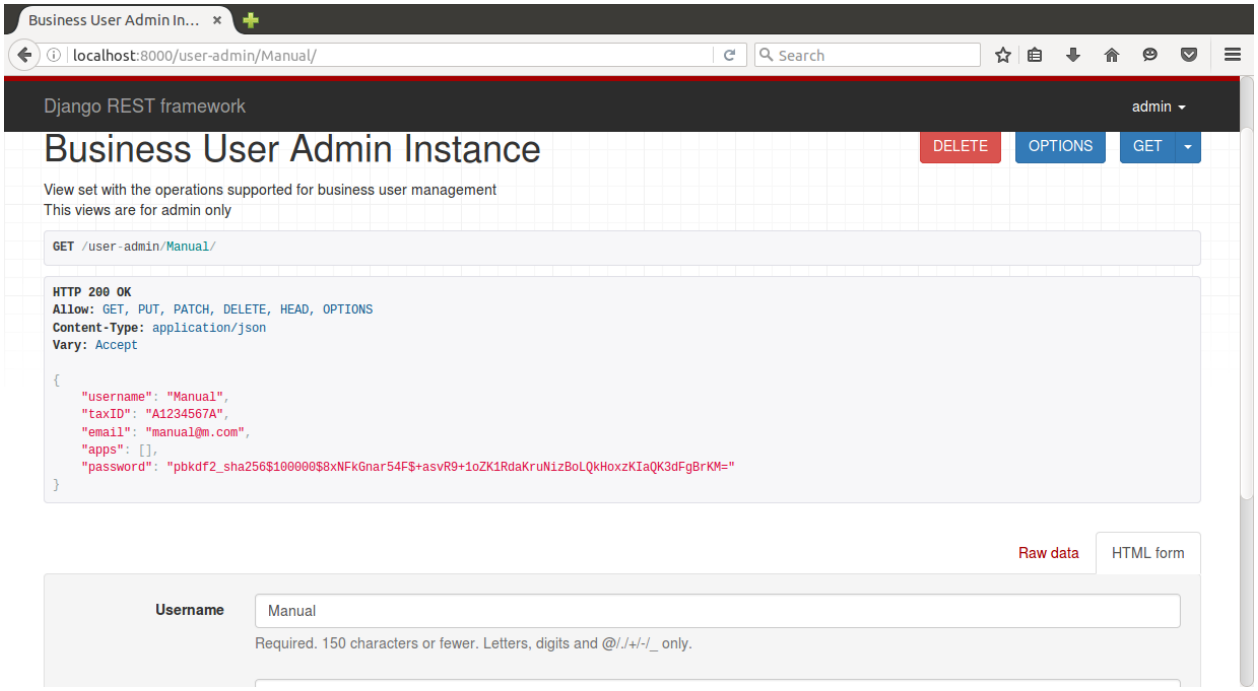

Figura 50: Borrar empresa

El comando equivalente es el siguiente:

```
1  http DELETE localhost:8000/user-admin/Manual/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
```

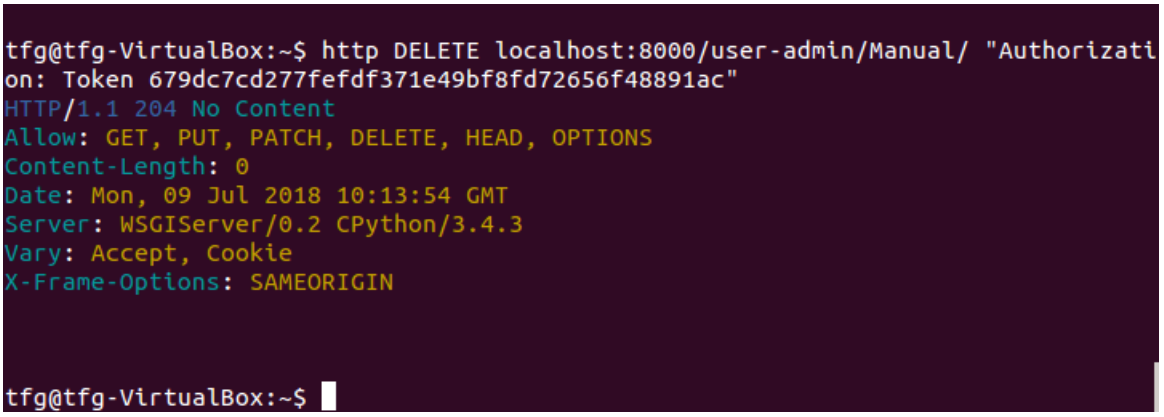

Figura 51: Comando para borrar una empresa

Si la empresa se borra correctamente recibirá una respuesta con código 204.

B.6. Crear aplicacion

En la página ***localhost:8000/app-admin/*** se listarán todas las aplicaciones inactivas y podrá crear nuevas.

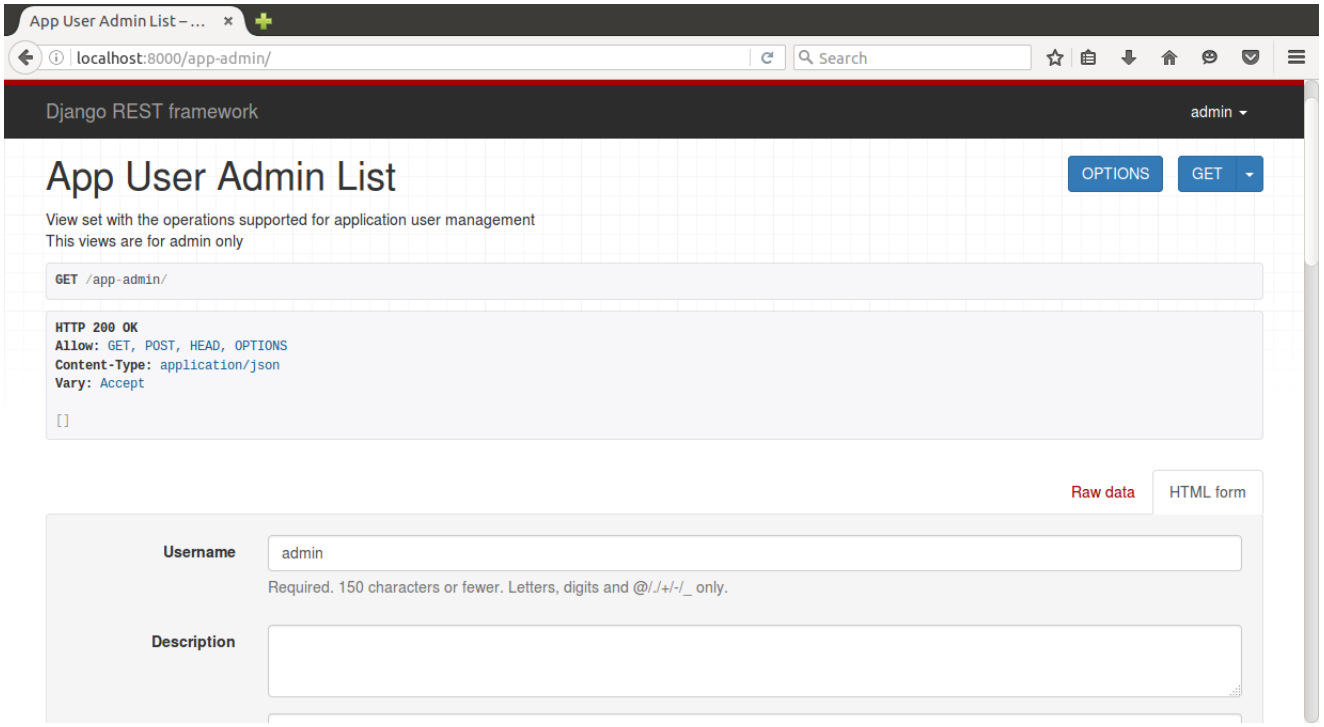

Figura 52: Pagina de listado y creación de aplicaciones para admin

El comando para crear una nueva aplicación:

```
1  http POST localhost:8000/app-admin/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
3  username=ManualApp description="App del manual" password=qazwsxed
4  permissions:='["createPerson","deletePerson"]' ower=Business1
```

Tenga en cuenta que puede crear y activar una aplicación de una vez añadiendo active=true al comando anterior. Para activar aplicaciones creadas por las empresas debe modificar dicha empresa y poner el campo active=true

```
tfg@tfg-VirtualBox:~$ http POST localhost:8000/app-admin/ "Authorization: Token
679dc7cd277fefdf371e49bf8fd72656f48891ac"    username=ManualApp description="App
 del manual" password=qazwsxed     permissions:='["createPerson","deletePerson"]
' owner=Business1
HTTP/1.1 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 228
Content-Type: application/json
Date: Mon, 09 Jul 2018 10:26:19 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "active": false,
    "description": "App del manual",
    "owner": "Business1",
    "password": "pbkdf2_sha256$100000$XWfQzRedCkZK$VKR2m1yJHP4xGBWyJ2jNnTKsz5rs3
YrPRY0SkR0Lw5Q=",
    "permissions": [
        "createPerson",
        "deletePerson"
    ],
    "username": "ManualApp"
}

tfg@tfg-VirtualBox:~$
```

Figura 53: Crear nueva aplicación

B.7. Modificar y ver detalles de la aplicacion

Desde la página ***localhost:8000/app-admin/ManualApp*** podrá ver los detalles de la aplicación. Desde esta página podrá además modificar sus datos.

Los comandos equivalentes son los siguientes:

Para ver detalles

```
1    http GET localhost:8000/app-admin/ManualApp/
2    "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
```

Este comando devolverá una respuesta con código 200 y los datos de la aplicación

Para modificar

```
1    http PUT localhost:8000/apps/ManualApp/
2    "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
3    username=ManualApp description="App  modificada" password=qazwsxed
4    permissions:='["createPerson","deletePerson"]'
```

Si la aplicación se modifica correctamente recibirá una respuesta con código 200 y los nuevos datos de la aplicación.

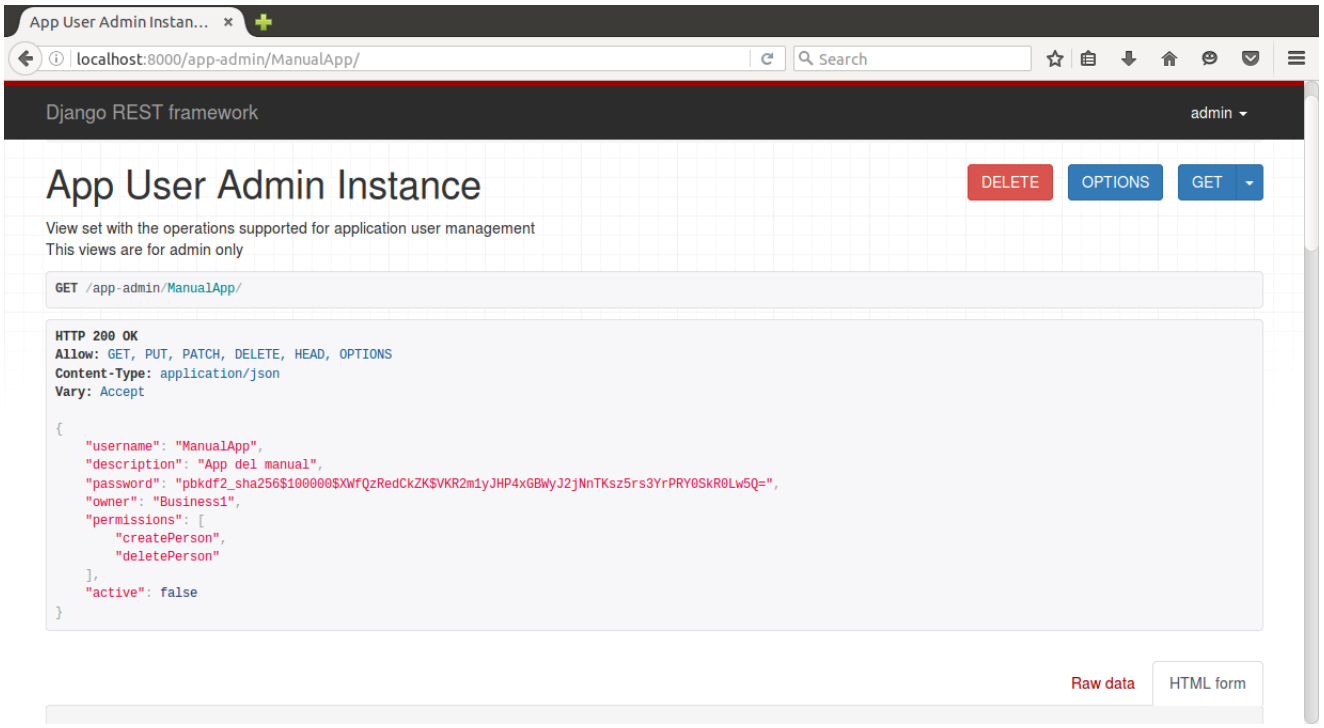

Figura 54: Página detalles y actualización de aplicaciones para admin

```
tfg@tfg-VirtualBox:~$ http GET localhost:8000/app-admin/ManualApp/ "Au
n: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 228
Content-Type: application/json
Date: Mon, 09 Jul 2018 10:59:21 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "active": false,
    "description": "App del manual",
    "owner": "Business1",
    "password": "pbkdf2_sha256$100000$XWfQzRedCkZK$VKR2m1yJHP4xGBWyJ2j
YrPRY0SkR0Lw5Q=",
    "permissions": [
        "createPerson",
        "deletePerson"
    ],
    "username": "ManualApp"
}

tfg@tfg-VirtualBox:~$
```

Figura 55: Ver detalles de aplicación

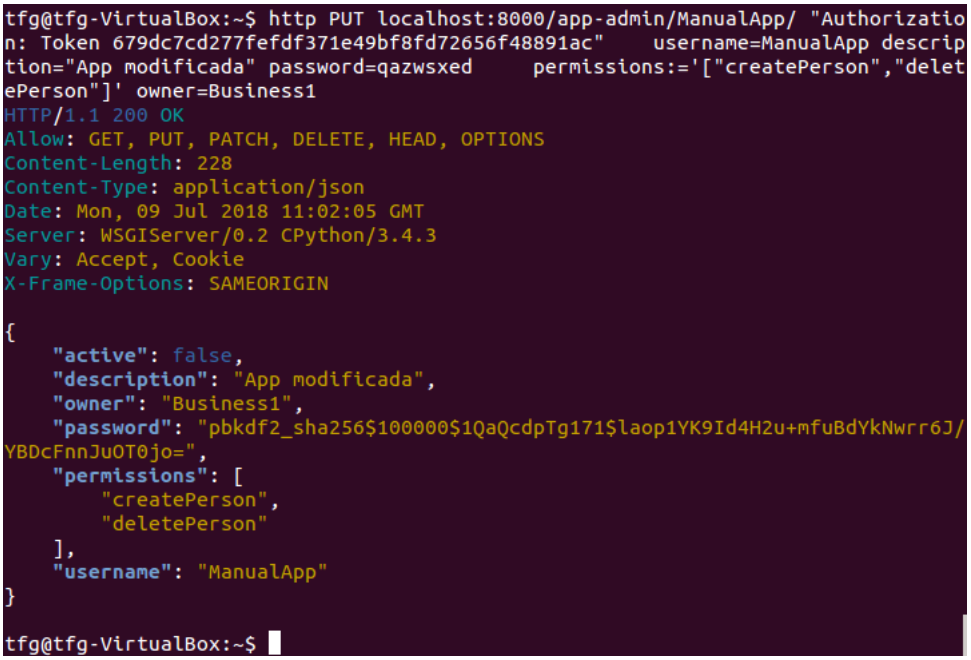

Figura 56: Modificar aplicación

B.8. Borrar aplicación

Desde la página ***localhost:8000/app-admin/ManualApp*** también podrá borrar la cuenta de la aplicación. Para ello únicamente deberá pulsar en el botón rojo de la izquierda y confirmar en la ventana emergente.

El comando equivalente es el siguiente:

```
1  http DELETE localhost:8000/app-admin/ManualApp/
2  "Authorization: Token cad7358a50df07d034ccc2f8306332820c6d03c8"
```

Si la aplicación se borra correctamente recibirá una respuesta con código 204.

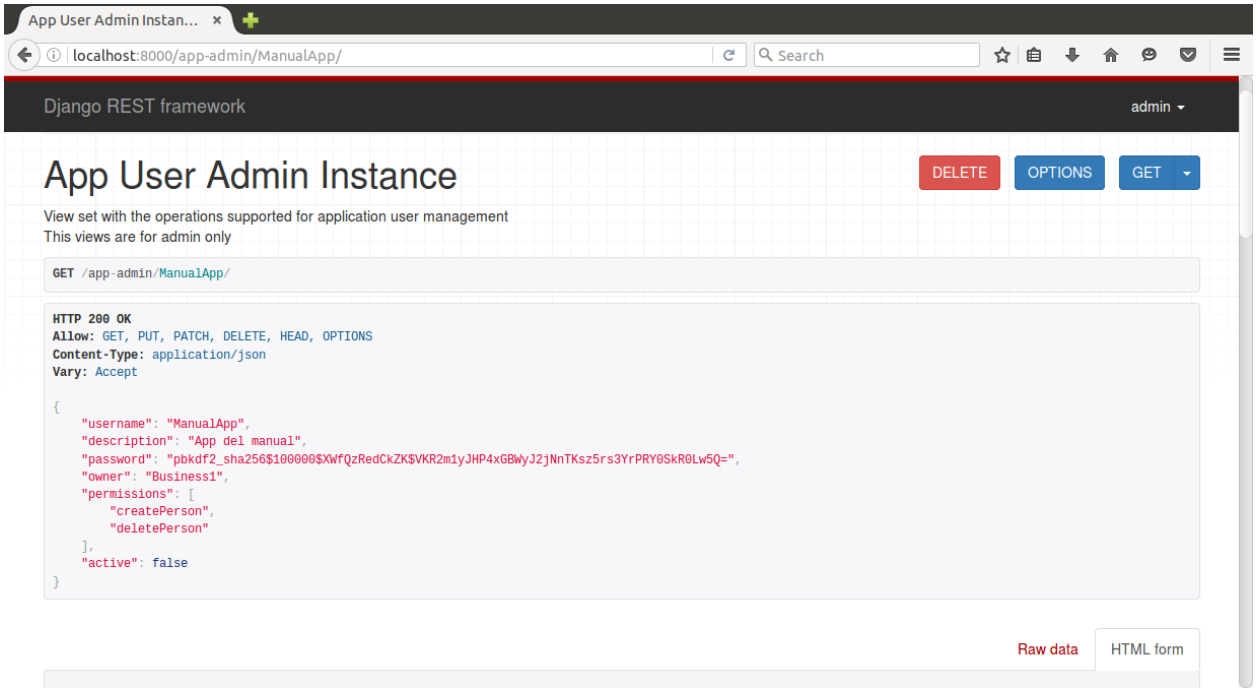

Figura 57: Borrar aplicación

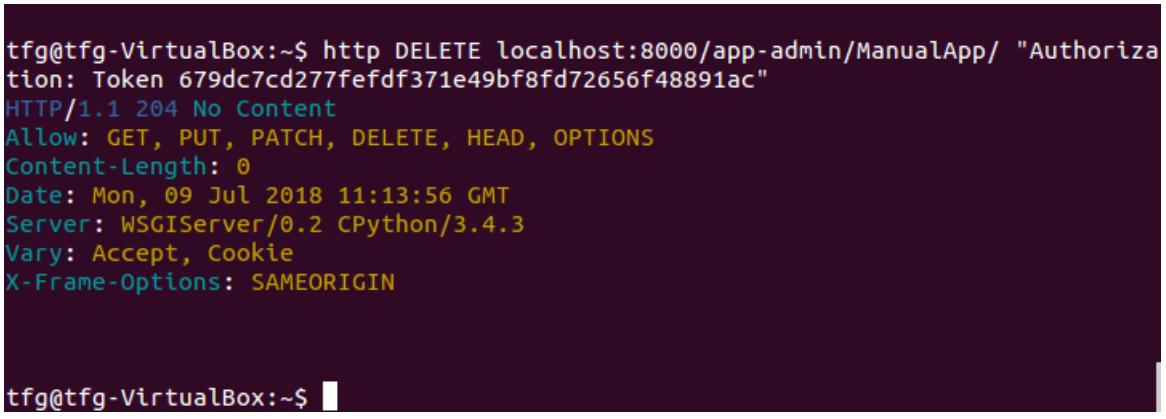

Figura 58: Comando para borrar aplicación

B.9. Crear permiso

En la página ***localhost:8000/permissions/*** se listarán todas los permisos y podrá crear nuevos.

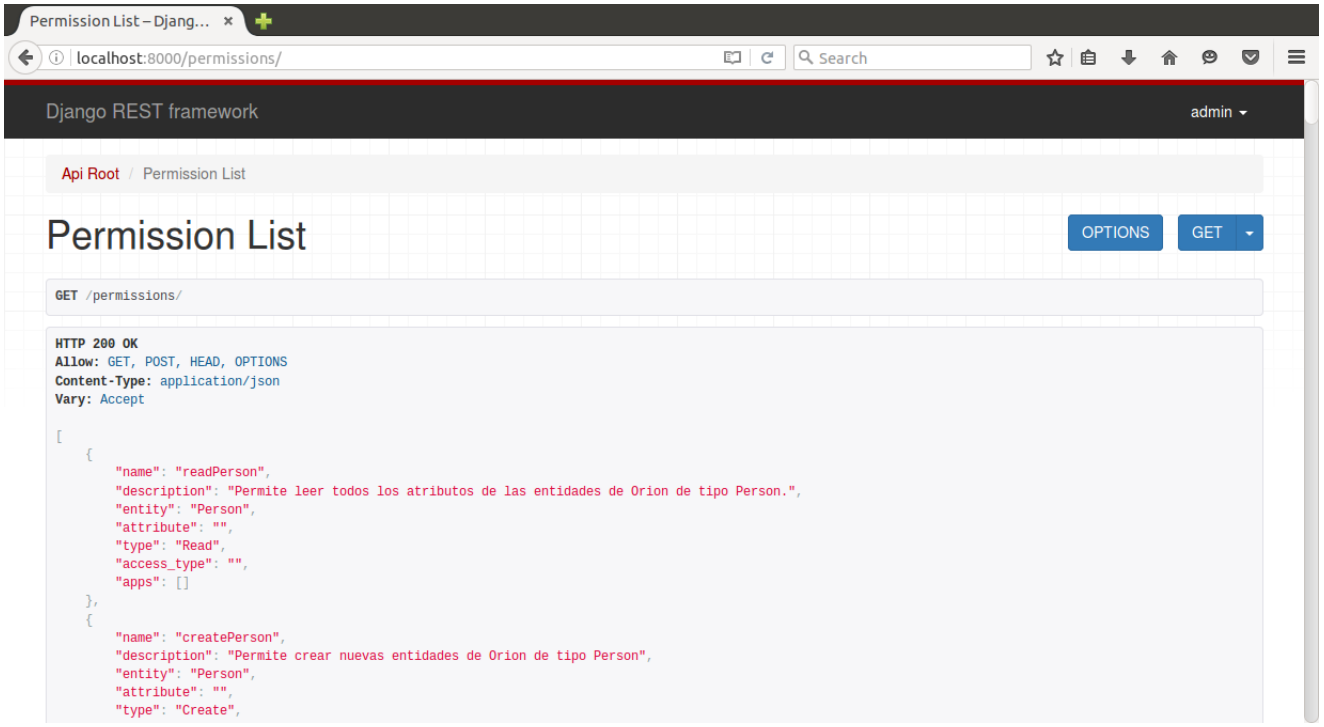

Figura 59: Pagina de listado y creación de permisos

El comando para crear una nuevo permiso:

```
1  http POST localhost:8000/permissions/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
3  name=ManualPer description="Permission del Manual"
4  entity=Vehicle attribute=owner
5  type=Read access_type=Processed
```

```
tfg@tfg-VirtualBox:~$ http  POST localhost:8000/permissions/ "Authorization: Tok
en 679dc7cd277fefdf371e49bf8fd72656f48891ac" name=ManualPer description="Permiss
ion del Manual"  entity=Vehicle attribute=owner type=Read access_type=Processed
HTTP/1.1 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 147
Content-Type: application/json
Date: Mon, 09 Jul 2018 11:22:29 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "access_type": "Processed",
    "apps": [],
    "attribute": "owner",
    "description": "Permission del Manual",
    "entity": "Vehicle",
    "name": "ManualPer",
    "type": "Read"
}

tfg@tfg-VirtualBox:~$
```

Figura 60: Crear nuevo permiso

B.10. Modificar y ver detalles del permiso

Desde la página ***localhost:8000/permissions/ManualPer*** podrá ver los detalles del permiso. Desde esta página podrá además modificar sus datos.

Los comandos equivalentes son los siguientes:

Para ver detalles

```
1  http GET localhost:8000/permissions/ManualPer/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
```

Este comando devolverá una respuesta con código 200 y los datos de la aplicación

Para modificar

```
1  http PUT localhost:8000/permissions/ManualPer
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
3  name=ManualPer description="Permiso modificado"
4  entity=Vehicle attribute=owner type=Read access_type=Processed
```

Si el permiso se modifica correctamente recibirá una respuesta con código 200 y los nuevos datos.

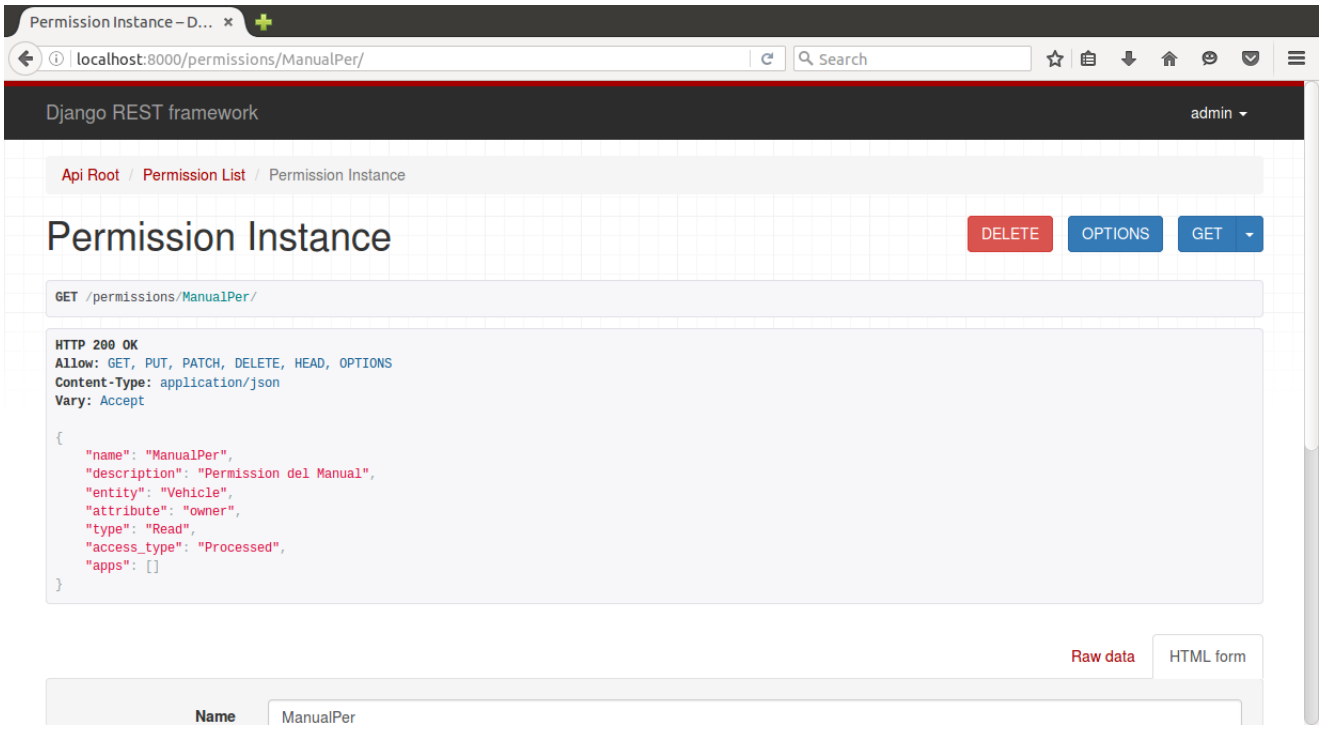

Figura 61: Página detalles y actualización de permisos

```
tfg@tfg-VirtualBox:~$ http  GET localhost:8000/permissions/ManualPer/ "Authoriza
tion: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 147
Content-Type: application/json
Date: Mon, 09 Jul 2018 11:30:36 GMT
Server: WSGIServer/0.2 CPython/3.4.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "access_type": "Processed",
    "apps": [],
    "attribute": "owner",
    "description": "Permission del Manual",
    "entity": "Vehicle",
    "name": "ManualPer",
    "type": "Read"
}

tfg@tfg-VirtualBox:~$
```

Figura 62: Ver detalles de permiso

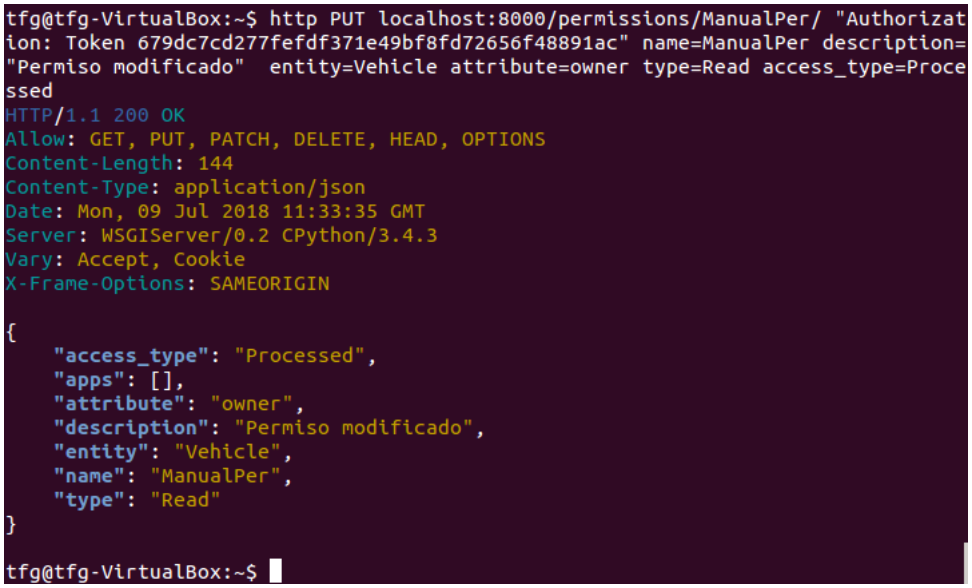

Figura 63: Modificar permiso

B.11. Eliminar permiso

Desde la página ***localhost:8000/permissions/ManualPer*** también podrá borrar el permiso. Para ello únicamente deberá pulsar en el botón rojo de la izquierda y confirmar en la ventana emergente.

El comando equivalente es el siguiente:

```
1  http DELETE localhost:8000/permissions/ManualPer/
2  "Authorization: Token 679dc7cd277fefdf371e49bf8fd72656f48891ac"
```

Si el permiso se borra correctamente recibirá una respuesta con código 204.

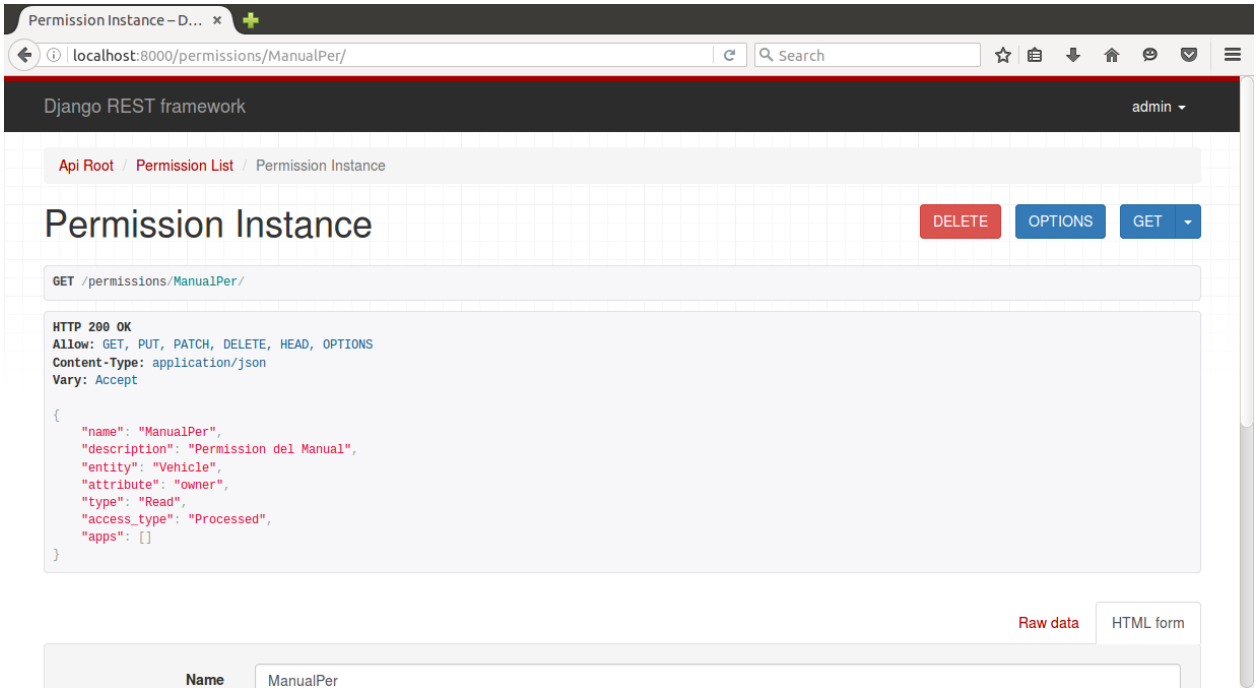

Figura 64: Borrar permiso

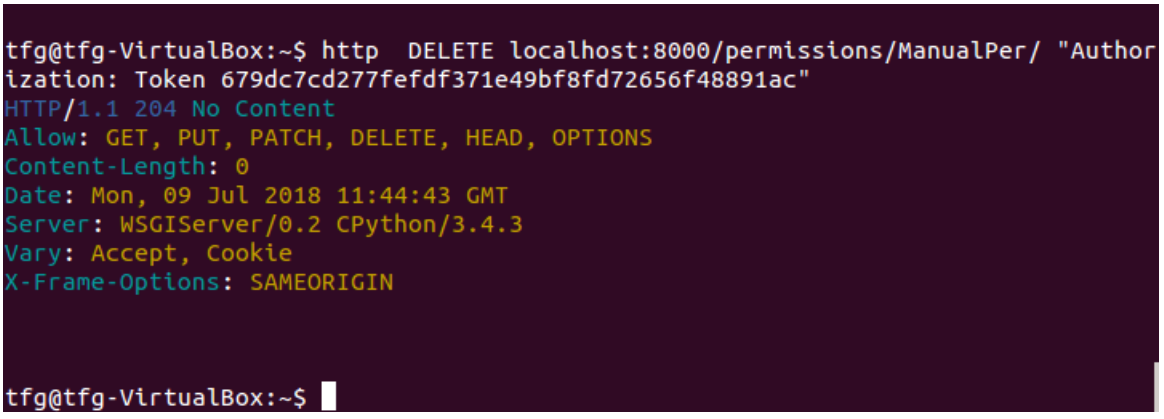

Figura 65: Comando para borrar permiso

C. Manual de instalación

Mímir es una aplicación creada usando Django, un framework para aplicaciones web en python. Para poder ejecutar la aplicación es necesario tener instalado un servidor web compatible con python.

C.1. Prerrequisitos

- Sistema UNIX (probado en Ubuntu 14.04)
- python 3.4
- docker-compose

C.2. Instalación

Desde el directorio raiz de la aplicación
Instalamos el gestor de paquetes python pip.

```
1  apt-get install python3-pip
```

Instalamos Virtualenv, una herramienta para crear entornos Python aislados.

```
1  pip3 install virtualenv
```

Creamos un nuevo entorno virtual

```
1  virtualenv env
```

Accedemos al entorno virtual

```
1  source env/bin/activate
```

Sabremos que estamos en el entorno virtual porque saldrá "(env)" al inicio de la linea de comandos.

Una vez en el entorno virtual instalaremos las dependencias recogidas en el fichero *requirements.txt*.

Para ello primero accederemos a la carpeta mimir

```
1  cd mimir
2  pip3 install -r requirements.txt
```

Tras esto Mimir ya estará instalado y listo para arrancar.

Antes de poder iniciar el servidor debemos levantar el servicio Orion que emplea Mímir. Para este paso es necesario tener instalado docker-compose, puedes encontrar los pasos necesarios en este enlace `https://docs.docker.com/compose/install/` en otro terminal vaya a la carpeta fiware ubicada en el directorio raíz levante Orion. La primera vez se descargarán los dockers de Orion y mongo y se lanzarán.

```
cd fiware
docker-compose up
```

C.3. Iniciar Mímir por primera vez

Por defecto Django incorpora un servidor para poder probar las aplicaciones durante el desarrollo. Para arrancar este servidor de pruebas basta con ejecutar el siguiente comando de **manage.py** generado automáticamente cuando se crea un proyecto Django.

```
python3 manage.py runserver
```

Como servidor de producción nos hemos decantado por Gunicorn un servidor HTTP WSGI ligero para UNIX. Si se han seguido los pasos hasta ahora gunicorn debería estar instalado en el entorno virtual. Para arrancar el servidor ejecute los siguientes comandos desde el directorio raíz:

```
source /env/bin/activate
cd mimir
gunicorn mimir.wsgi
```

En otro terminal desde el directorio raíz, levante Orion

```
cd fiware
docker-compose up
```

```
cd fiware
docker-compose up
```

Puede rellenar Orion con datos de muestra ejecutando el script poblarOrion ubicado en la carpeta scripts.

Mímir viene incluido con usuarios y permisos de prueba.

D. Contenido del CD

- memoria.pdf
- código
  - fiware/
    - docker-compose.yml
  - mimir/
    - manage.py
    - db.sqlite3
    - requirements.txt
    - mimir/
      - settings.py
      - urls.py
      - constants.py
      - accounts/
      - request/
- manual-de-usuario.pdf
- manual-de-administrador.pdf
- namual-de-instalacion.pdf