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 flujo de los datos personales

Autor:
Roberto Bahillo Ortego

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 flujo de los datos personales

Autor:
Roberto Bahillo Ortego

Tutores:
Pablo de la Fuente Redondo
Julián Arroyo Álvarez

Agradecimientos

Me gustaría agradecer en primer lugar a mis tutores por su confianza a la hora de proponer este trabajo y por su paciencia y ayuda para poder llevarlo a cavo.

A la Escuela de Ingeniería Informática por darme la oportunidad de demostrar los conocimientos que me han enseñado y la capacidad para profundizar más en ellos.

A mi familia y amigos por su apoyo a lo largo de este proyecto, aun con todos los malos ratos que han pasado, siempre he podido contar con vosotros.

A David Gracía Matía por nuestra ayuda mutua.

Resumen

El Reglamento General de Protección de Datos es la nueva norma que viene desde Europa acerca de como tratar los datos personales de todos los usuarios de la Unión. Este reglamento sustituye a la Ley Orgánica de Protección de Datos aunque en muchos casos se complementan.

Por su parte, FIWARE es una plataforma para la creación y desarrollo de aplicaciones sobre todo para el entorno de *smart-cities* y para *Internet of Things*, las cuales están muy en alza desde hace ya un tiempo y parece que seguirán siendo tecnologías importantes en los años venideros.

De esta forma este Trabajo Fin de Grado se ha desarrollado teniendo en mente la explicación entre las diferencias que más importantes en el aspecto legal para los informáticos, y el desarrollo de una aplicación usando FIWARE para comprobar como de bien preparado esta para adaptarse al Reglamento General de Protección de Datos que ha entrado en vigor el día 25 de Mayo de 2018.

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 nuevas adiciones que este representa tanto para el usuario como para la empresa y la capacidad que tiene FIWARE para adaptarse al reglamento. Esto se realizara mediante la creación de un prototipo donde se atacaran ciertos aspectos de la privacidad que aparecen en el RGPD. Esto se va a lograr mediante:

- El uso de tecnologías que nos asistan, no solo con el desarrollo del prototipo, si no con su mantenimiento posterior y futuras modificaciones.
- Aplicar los conocimientos adquiridos durante la carrera acerca de programación.
- Realizar un desarrollo correcto que ofrezca unas garantías mínimas de seguridad, protección de datos y estabilidad de la aplicación.

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, de como estaba antes, con la Ley Orgánica de Protección de Datos (en adelante LOPD), a como esta ahora con el RGPD.

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. En mi caso las medidas de seguridad son las que nos ofrece el *framework* sobre el que vamos a trabajar y la barrera que nos ofrece la propia máquina virtual a la hora de acceder a la base de datos.

1.3. Estructura del documento

El documento esta estructurado de la siguiente manera,

El capítulo dos explica la planificación llevada en el proyecto, así como los posibles riesgos que se pueden dar durante el desarrollo de este.

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

El capítulo cuatro explica que es FIWARE y las herramientas que pone a disposición de los usuarios.

El capítulo cinco explica las tecnologías usadas a la hora de desarrollar el proyecto.

El Capitulo seis trata del la explicación y el desarrollo del prototipo para saber lo preparado que esta actualmente FIWARE para el nuevo reglamento.

En el capítulo siete se desarrollan las pruebas realizadas para evaluar la robustez del proyecto.

El capítulo ocho son las conclusiones sobre el proyecto y trabajo futuro.

2. Planificación

Para el desarrollo de este proyecto, y debido a que tenia una gran necesidad de adquisición de conocimientos a la hora de entender tanto FIWARE como el RGPD, se a optado por una planificación iterativa incremental.

2.1. Planificación del trabajo

Para el desarrollo de este trabajo, que se empezó en el mes de Octubre del año 2017, se tenía planeado un desarrollo de tal manera que se terminara sobre los meses de Enero o Febrero del año siguiente. Lamentablemente, y debido a problemas inevitables, el desarrollo de la aplicación se vio aplazada hasta los meses de Abril-Mayo debido principalmente a una prolongación en los tiempos de la búsqueda de información acerca del RGPD y FIWARE.

Durante el desarrollo de este proyecto se han contado con los siguientes recursos para el desarrollo del proyecto:

- **Recursos humanos**: Una persona realizará el proyecto, por lo que tendrá los roles de analista, programador y teste.
- **Otros recurso**: Durante el desarrollo del proyecto se ha contado con recursos adicionales, como un ordenador para el desarrollo del documento y del prototipo y una máquina virtual cedida por la Escuela de Ingeniería Informática que se utilizará para poner en producción el prototipo.

Teniendo en cuentas esto, y la dificultad a la hora de implementar SCRUM como plan de trabajo para este proyecto, se decide realizar un método iterativo e incremental más simple. A medida que se necesitaba realizar distintas tareas, se iban añadiendo a una lista de tareas y se iban realizando. La lista de tareas actúa como una lista de prioridad donde esta, la prioridad, viene dada por la siguiente tabla.

Relación	Prioridad
Documentación	Alta
Prototipo: Funcionalidad	Media
Prototipo: estilo	Baja

Cuadro 1: Prioridades

Una vez una tarea se empieza a desarrollar no es interrumpida hasta que no se acaba. Para aprovechar mejor el tiempo del que se dispone, las tareas que se realizan a la vez que en la Figura 1, se hacen en intervalos de tiempo diferente dentro del mismo día.

2.1.1. Iteraciones

Durante este proyecto se han realizado varias iteraciones, intentando mantener estas lo más independientes posibles. Para esto, se ha seguido los siguientes pasos.

Las iteraciones segunda, tercera y cuarta se sucedieron a la vez.

Id-01	Fase de inicio
Predecesoras	-
Duración	0 horas/persona
Descripción	Se inicia el desarrollo del proyecto.

Cuadro 2: Primera iteración: Fase de inicio.

Id-02	RGPD, Búsqueda de información y compresión
Predecesoras	Id-01
Duración	35 horas/persona
Descripción	Búsqueda de información acerca del RGPD y comprensión de los conocimientos adquiridos.

Cuadro 3: Segunda iteración: RGPD, Búsqueda de información y compresión.

Id-03	FIWARE, Búsqueda de información y compresión
Predecesoras	Id-01
Duración	35 horas/persona
Descripción	Búsqueda de información acerca del FIWARE y comprensión de los conocimientos adquiridos.

Cuadro 4: Tercera iteración: FIWARE, Búsqueda de información y compresión.

Id-04	Desarrollo de la parte del RGPD y FIWARE en el documento
Predecesoras	Id-01
Duración	120 horas/persona
Descripción	Desarrollo en el documento a cerca de las partes relacionadas con el RGPD y FIWARE

Cuadro 5: Cuarta iteración: Desarrollo de la parte del RGPD y FIWARE en el documento.

Id-05	Revisión de la documentación, partes del RGPD y FIWARE
Predecesoras	Id-04
Duración	80 horas/persona
Descripción	Se realiza una revisión y actualización sobre las partes del RGPD y FIWARE en el documento.

Cuadro 6: Quinta iteración: Revisión de la documentación, partes del RGPD y FIWARE.

Id-06	Desarrollo del prototipo
Predecesoras	Id-04
Duración	20 horas/persona
Descripción	Se inicia el desarrollo del prototipo y se plasma la idea sobre la que se va a iterar.

Cuadro 7: Sexta iteración: Desarrollo del prototipo.

Id-07	Implementación del mapa
Predecesoras	Id-06
Duración	10 horas/persona
Descripción	Se implementa el mapa en el proyecto.

Cuadro 8: Séptima iteración: Implementación del mapa.

Id-08	Integración de Orion en el proyecto
Predecesoras	Id-07
Duración	15 horas/persona
Descripción	Se realiza la integración de Orion en el proyecto.

Cuadro 9: Octava iteración: Integración de Orion en el proyecto.

Id-09	Implementación del registro y la identificación de usuarios
Predecesoras	Id-08
Duración	10 horas/persona
Descripción	Se implementa el registro y la identificación de los usuarios.

Cuadro 10: Novena iteración: Implementación del registro e identificación de usuarios.

Id-10	Planificación e implementación de las sesiones
Predecesoras	Id-09
Duración	35 horas/persona
Descripción	Se plantea el dilema acerca del funcionamiento de las sesiones, se resuelve y se implementan en el proyecto.

Cuadro 11: Décima iteración: Planificación e implementación de las sesiones.

Id-11	Implementación de las reservas
Predecesoras	Id-10
Duración	15 horas/persona
Descripción	Se implementan las reservas en el prototipo.

Cuadro 12: Decimoprimera iteración: Implementación de las reservas.

Id-12	Implementación del perfil de usuario
Predecesoras	Id-11
Duración	15 horas/persona
Descripción	Se crean las vistas y el código para mostrar el perfil del usuario.

Cuadro 13: Decimosegunda iteración: Implementación del perfil de usuario.

Id-13	Creación del CSS y un header común
Predecesoras	Id-12
Duración	10 horas/persona
Descripción	Se crea un CSS y un header comun para todas las páginas.

Cuadro 14: Decimotercera iteración: Creación de un CSS y un header común.

Id-14	Habilitar el *host* compartido
Predecesoras	Id-13
Duración	10 horas/persona
Descripción	Se habilita un *host* compartido con ayuda de una máquina virtual cedida por parte de la Escuela de Ingeniería Informática para esto.

Cuadro 15: Decimocuarta iteración: Habilitar el *host* compartido.

Id-15	Añadir las partes de desarrollo e implementación del prototipo
Predecesoras	Id-05
Duración	80 horas/persona
Descripción	Se añaden las partes relacionadas con la implementación y el desarrollo del prototipo.

Cuadro 16: Decimoquinta iteración: Añadir las partes de desarrollo e implementación.

Id-16	Revisión de la implementación y del desarrollo
Predecesoras	Id-15
Duración	40 horas/persona
Descripción	Se revisan las partes de implementación y desarrollo del prototipo debido a un problema con las sesiones.

Cuadro 17: Decimosexta iteración: Revisión de la implementación y el desarrollo.

Id-17	Añadir al documento las partes de tecnologías utilizadas, desarrollo de pruebas y conclusiones y trabajo futuro
Predecesoras	Id-16
Duración	40 horas/persona
Descripción	Se añaden al documento las partes de tecnologías utilizadas, desarrollo de pruebas y conclusiones y trabajo futuro.

Cuadro 18: Decimoséptima iteración: Añadir al documento las partes de tecnologías utilizadas, desarrollo de pruebas y conclusiones y trabajo futuro.

Id-18	Revisión final del proyecto
Predecesoras	Id-17
Duración	20 horas/persona
Descripción	Se revisa todo el documento y el prototipo.

Cuadro 19: Decimoctava iteración: Revisión final del proyecto.

Id-19	Finalización del proyecto
Predecesoras	Id-18 e Id-14
Duración	0 horas/persona
Descripción	El proyecto se da por finalizado.

Cuadro 20: Decimonovena iteración: Finalización del proyecto.

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

1	✓		Fase de inicio	0 días	
2	✓		Busqueda de información y comprensión del RGPD	35 horas	1
3	✓		Bsqueda de información y compresión FIWARE	35 horas	1
4	✓		Desarrollo de las parte del RGPD y FIWARE en el documento	120 horas	1
5	✓		Revisión de la documentación, parte del RGPD y FIRAWE	80 horas	4
6	✓		Desarrollo del prototipo	20 horas	4FC+80 horas
7	✓		Implementación del mapa	10 horas	6
8	✓		Integración de Orion en el proyecto	15 horas	7
9	✓		Implementación del registro y la identificación de usuarios	10 horas	8
10	✓		Planificación e implemetación de las sesiones	35 horas	9
11	✓		Implementación de las reservas	15 horas	10
12	✓		Implementación del perfil de usuario	15 horas	11
13	✓		Creación del CSS y un header común	15 horas	12
14	✓		Habilitar el host compartido	10 horas	13
15	✓		Añadir las partes de desarrollo e implementación del prototipo	80 horas	5
16	✓		Revisión de la implementación y desarrollo	40 horas	15
17	✓		Añadir al documento las partes de tecnologías utilizadas, desarrollo de pruebas y conclusiones y trabajo futuro	40 horas	16
18	✓		Revisión final del proyecto	20 horas	14;17
19	✓		Finalización del proyecto	0 días	18

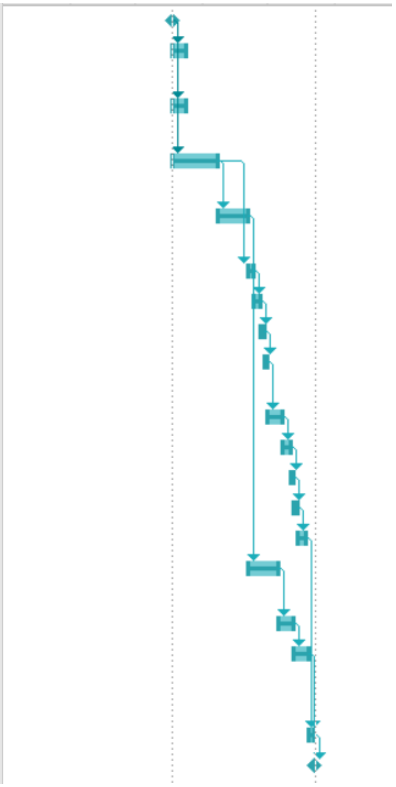

Figura 1: Planificación real del proyecto

Como se puede observar en la Figura 1, las partes de buscar información sobre el RGPD y FIWARE sucedieron a la vez, habiendo tras ellas varias modificaciones y revisiones del proyecto que se han ido llevando a cavo hasta el final de este. Estas se han ido produciendo durante todo su desarrollo, siendo esta la razón por la que se tardara tanto en empezar con el desarrollo del prototipo.

2.2. Plan de Riesgos

Para el desarrollo del proyecto necesitamos analizar también los riesgos a los que nos vamos a poder enfrentar durante el desarrollo de este. Esto nos será útil a la hora de enfrentarnos a dichos riesgos para poder solucionarlos. Para esto se van a analizar los riesgos, dándoles un valor dependiendo de su probabilidad y el impacto general en el proyecto, siguiendo el ejemplo que nos ofrece Pressman[11], de esta forma podremos saber su repercusión en el proyecto.

Tras analizar la repercusión de las diferentes posibilidades de riesgos, lo siguiente es evaluar los distintos riesgos que pueden ocurrir junto con sus planes de protección y contingencia.

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 21: Identificación valor de riesgo frente a Probabilidad/Impacto. Fuente: Torres de Fenicia

Riesgo 01	Ordenador de trabajo estropeado
Impacto	Crítico
Probabilidad	Baja
Plan de Protección	Tener los datos del proyecto guardados con una copia de seguridad y tener a mano otro ordenador donde poder programar.
Plan de Contingencia	Recuperar el último estado almacenado en la copia de seguridad y usarlos en el otro ordenador.

Cuadro 22: Riesgo 01: Ordenador de trabajo estropeado

Riesgo 02	Modificación de la página de inicio de FIWARE
Impacto	Marginal
Probabilidad	Alta
Plan de Protección	Buscar directamente la página de *catalog* de FIWARE.
Plan de Contingencia	Volver a aprender a navegar la página de FIWARE si se necesita algo a parte del *catalog*.

Cuadro 23: Riesgo 02: Modificación de la página de inicio de FIWARE

Riesgo 03	*General Enabler* mal documentado
Impacto	Marginal
Probabilidad	Alta
Plan de Protección	-
Plan de Contingencia	Evitar en la medida de lo posible su uso

Cuadro 24: Riesgo 03: *General Enabler* mal documentado

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[20].

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[25]:

- 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. [27]

- 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[21]. 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[26]:

- 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 25: 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.

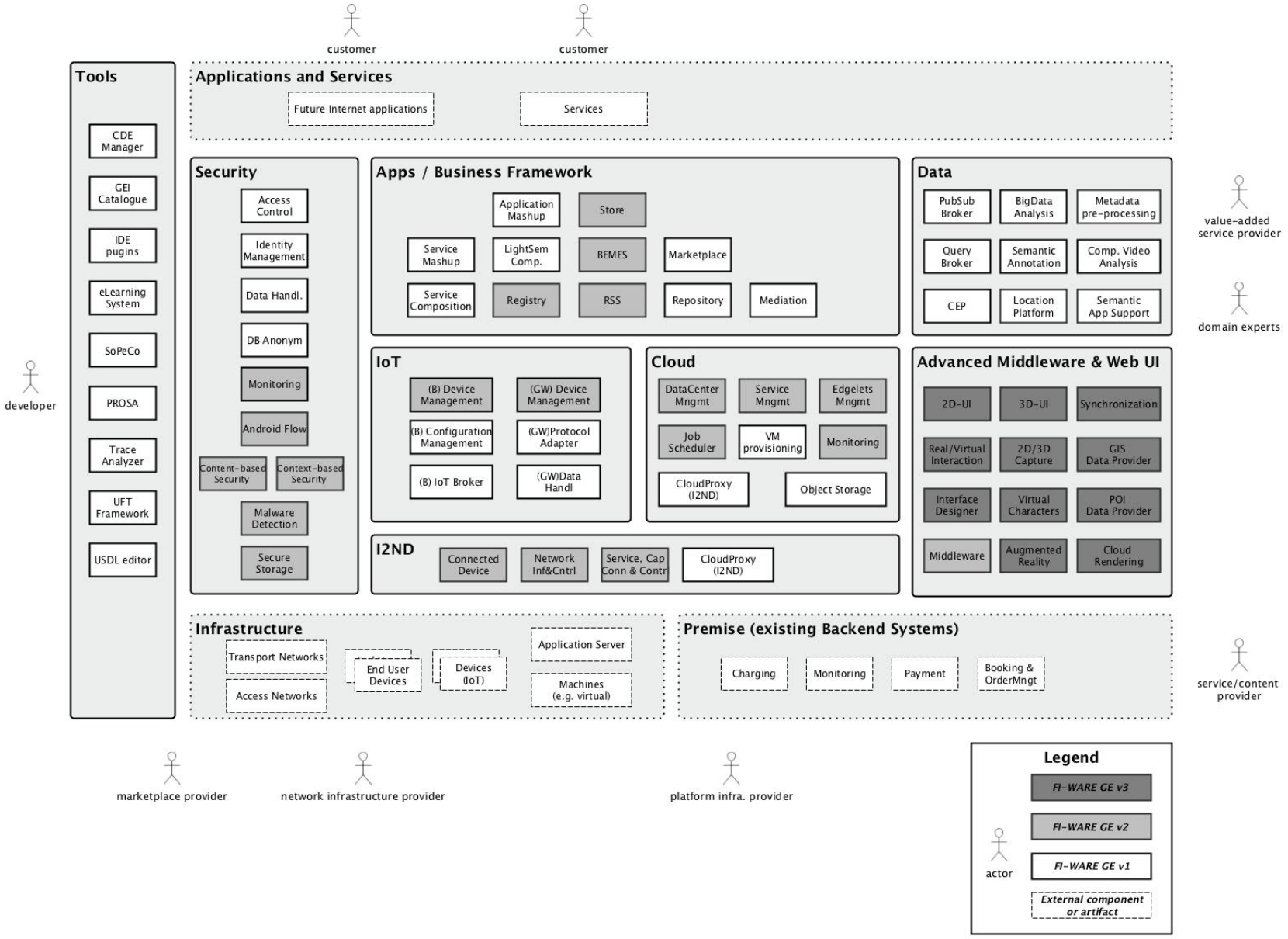

Figura 2: Arquitectura general de FIWARE. Fuente: FIWARE, *Architecture* R4

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.

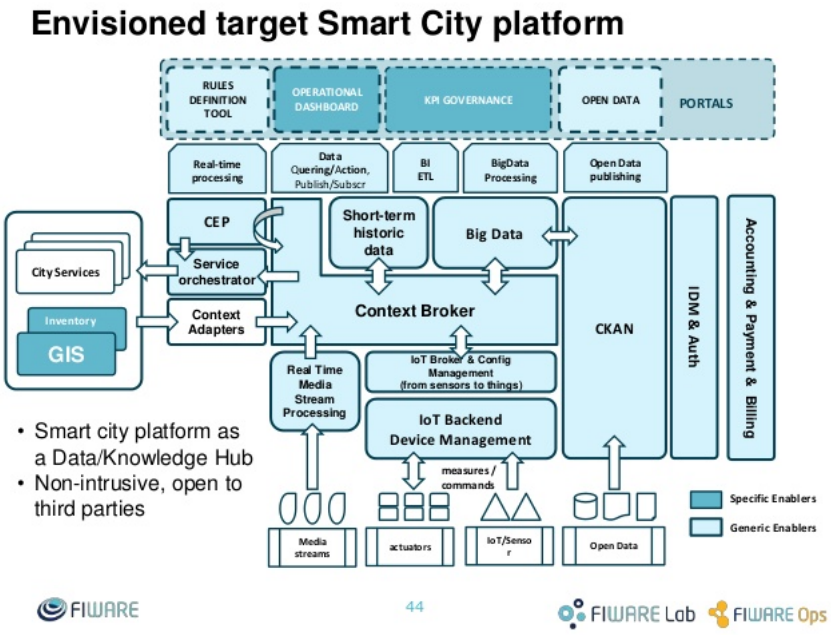

Figura 3: Objetivo previsto de una plataforma de *smart city*. Fuente: FIWARE: *an open standard platform for smart cities* [18]

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. [18]

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

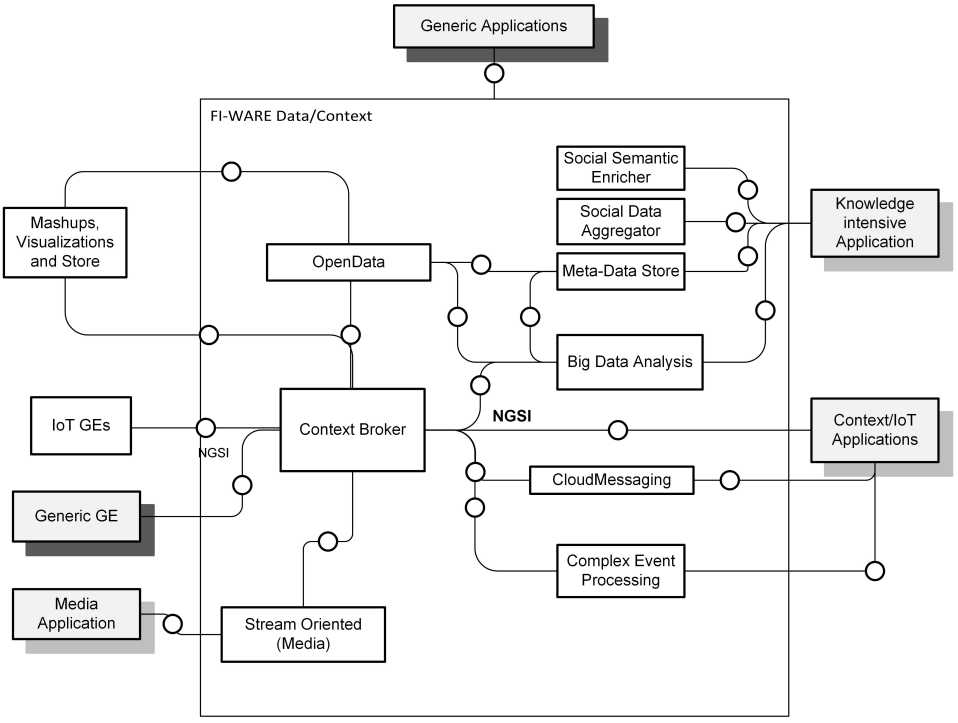

Figura 4: Fuente: FIWARE, *Architecture Context Management*

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

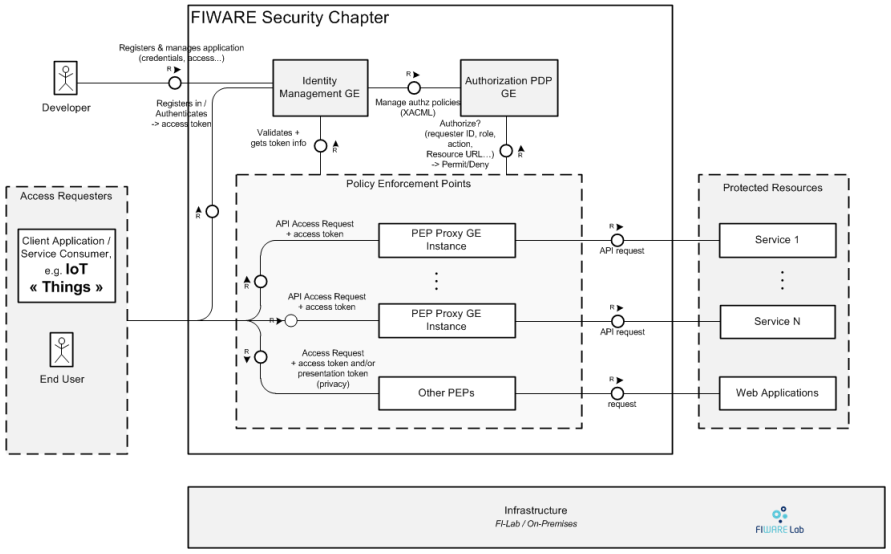

Figura 5: Fuente: FIWARE, *Architecture Security*

todos los requisitos citados, el GE PEP *Proxy* pasará la petición al recurso protegido en cuestión y devolverá la respuesta al usuario.[28]

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.

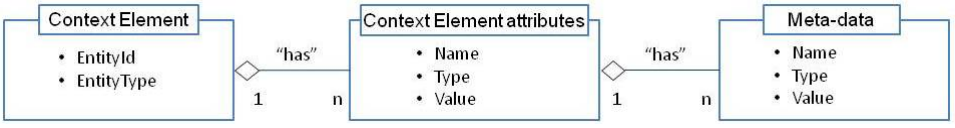

Figura 6: Modelo de la estructura de un elemento de contexto

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.

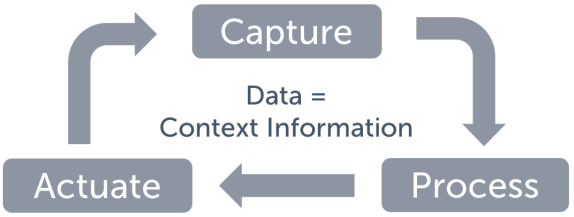

Figura 7: Ciclo de acción de los datos en FIWARE

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**[23] 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. Tecnologías utilizadas

En esta sección se explicarán las tecnologías utilizadas a la hora de desarrollar el proyecto, detallándose también las razones por las que se utilizan.

5.1. Navegadores Web

Los navegadores web son software que permite acceso a la web interpretando código de diferentes tipos de archivos y páginas web para que estos puedan ser visualizados por los usuarios:
El desarrollo web ofrece unas ventajas importantes.

- Cualquier dispositivo con acceso a un navegador web puede interpretar el código que se le presenta y acceder a las aplicaciones que se crean en la web.
- Miles de usuarios usan navegadores web todos los días, por lo que se ha vuelto una herramienta con un gran alcance. Esto facilita el alcance a más usuarios, pero también dificulta la creación de páginas web, las cuales tienen que adaptarse a distintos tamaños de pantallas.
- En la mayoría de las ocasiones, en el lado del cliente no hace falta instalar nada más que el propio navegador para tener acceso a las diferentes páginas.

Esto hace de las herramientas web algo a tener muy en cuenta. Si analizamos la cuota de uso de los diferentes navegadores web, podemos ver, como se muestra en la Figura 8, que los usuarios de Chrome representa más del 58 % del total durante el mes de Mayo de este año[5].

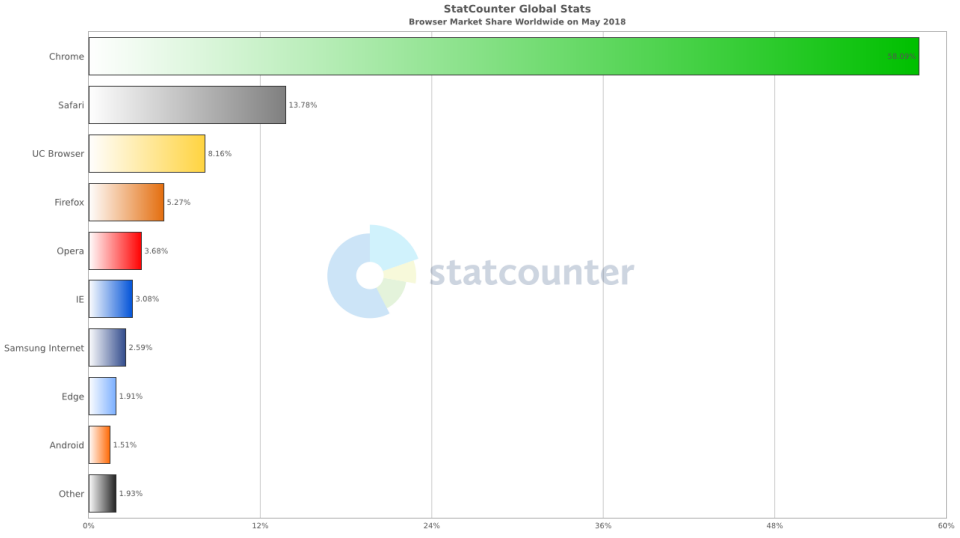

Figura 8: Uso de los distintos navegadores web según StatCounter

Esto nos indica que nuestra aplicación deberemos tener en cuenta que la gran mayoría de los usuarios van a usar este navegador, pero debemos intentar no dejar de lado al resto de los usuarios.

[5]Para más información y mejor visión de la imagen, visitar la página `http://gs.statcounter.com/browser-market-share#monthly-201805-201805-bar`, 30 de jun. de 2018.

5.2. *HyperText Markup Language* (HTML)

HTML es un lenguaje de programación usado para crear páginas web. Es un estándar a cargo del *World Wide Web Consortium* (W3C), una organización dedicada a la estandarización de tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Es un lenguaje sencillo, pero que tiene mucha potencia. En nuestra aplicación lo usaremos sobre todo para la creación de vistas.

5.3. *Hypertext Preprocessor* (PHP)

Hypertext Preprocessor[17] es un lenguaje de código abierto para el desarrollo de aplicaciones web que puede ser incrustado en HTML. Es un lenguaje usado para crear aplicaciones web principalmente. Existe gran variedad de versiones de PHP, habiendo que tener en cuenta que versión se planea utilizar si se quiere desarrollar una aplicación que utilice un framework para PHP, como es este caso, ya que usaremos Laravel. Entre sus ventajas cave destacar:

- Orientado al **desarrollo de aplicaciones web**.
- Lenguaje de programación calificado como **fácil de aprender**.
- El código escrito en PHP es **invisible para el navegador**, y por lo tanto, para el cliente.
- Es **libre**, facilitando el acceso a los usuarios.
- Es un **lenguaje estable** y que **sigue evolucionando** y desarrollándose.
- Existe una **gran variedad de *frameworks*** diferentes para desarrollar en PHP.
- Tiene una **extensa documentación** y resulta fácil encontrar ejemplos.

5.4. Laravel

Laravel[24] es un *framework* de desarrollo para **web artisan**. Este framework trae consigo características que facilitan mucho su uso:

- **Amplia documentación** que resulta fácil de navegar.
- Posee un magnifico **rendimiento**.
- Simplifica mucho la creación y el mantenimiento de las **rutas** y la **autenticación**.
- Facilita la **adición** de nuevas librerías y repositorios externos.
- Posee una **comunidad activa** que crea y mantiene los repositorios de manera constante.
- Facilita el uso del **patrón MVC**, Modelo Vista Controlador.
- Posee un *template* específico, **blade**, simple pero con mucha potencia.
- Es un framework de código abierto bajo la licencia MIT.

- **Eloquent ORM** permite la creación de Modelos para trabajar de manera transparente con tablas de las bases de datos.
- Si se tienen conocimientos básicos de PHP, este *framework*, junto con la documentación, se vuelve una herramienta con mucha potencia y fácil de empezar a manejar.
- Se tiene experiencia previa puesto que se usó durante las practicas en empresa.

Para la implementación del prototipo que vamos a desarrollar usaremos la versión 5.5, lo que hace que como mínimo necesitemos tener la versión de PHP 7.0 instalada.

5.5. PhpStorm

PhpStorm[6] es un IDE para desarrollo, principalmente, de PHP.

- Es un IDE de desarrollo de pago, pero por ser estudiante de la Universidad de Valladolid, JetBrains ofrece una licencia anual de manera gratuita.
- Editor para el manejo de **PHP**, **HTML**, **CSS**, **Javascript** y **SQL**.
- Ofrece la oportunidad de hacer **refactorizaciones**, como el cambio de nombre de un archivo, **de manera automática**.
- Soporta diferentes versiones de PHP, desde la 5.3 hasta la 7.0 y superior.
- Tiene un **entorno de desarrollo** amigable y fácil de comprender y navegar.
- Se tiene experiencia previa puesto que se usó durante las practicas en empresa.

5.6. FIWARE

FIWARE es una plataforma para el desarrollo de aplicaciones, sobre todo orientadas al *Internet of Things* (IoT) y las *smart cities*. Esta plataforma pone a nuestra disposición diversas herramientas que podemos usar. Dado que queremos que nuestra aplicación sea considerada FIWARE, lo mínimo necesario es el uso de Orion. Otra herramienta que podríamos usar es **KeyRock**

KeyRock es un gestor de identidades. Este suele ser usado junto con **Wilma** y **AuthForce**, pero puede usarse solo. A primera vista esta herramienta podría resultar útil, sobre todo para el mantenimiento de sesiones, pero surgen varios problemas, por ejemplo la carencia de una documentación decente para su uso o que Laravel se encarga fácilmente de este asunto, disminuyendo drásticamente la complejidad del prototipo.

6. Desarrollo del supuesto: *Car Renting*

Mediante el uso de FIWARE y aplicando el RGPD se planea realizar una aplicación que sirva para poder alquilar vehículos por tiempo de uso, mostrando la localización de los vehículos, su nivel de batería y autonomía, y permitirle al usuario reservar un vehículo.

Para su desarrollo vamos a estudiar otra aplicación real con el fin de identificar posibles problemas de compatibilidad con el RGPD y que puedan ser solucionados. Dicha aplicación será ZITY, que funciona tanto para móviles Android como para Iphone, y nos servirá como primera toma de contacto para estudiar lo mencionado anteriormente. Tras esto se realizara una versión utilizando alguna de las herramientas que FIWARE pone a disposición.

Lo primero que queremos hacer es saber si esta aplicación cumple con el RGPD, dado que esta empresa recoge datos personales como son el DNI o el numero de cuenta bancaria para poder realizar los pagos. De esta manera, sabremos si podemos basarnos en ella o tenemos que realizar cambios a la hora de crear el prototipo para adaptarnos al RGPD. También nos es interesante averiguar con que fin recaba los datos la aplicación. Para esto vamos a fijarnos en la política de privacidad de la empresa.[6]

> "Los datos personales que sean recabados a través de la Aplicación, el Sitio Web y/o en el marco de la prestación del Servicio se incorporarán a ficheros titularidad de la SOCIEDAD con la finalidad de posibilitar la gestión de los Usuarios del Servicio, la prestación del Servicio y, en general, la gestión, desarrollo y cumplimiento de la relación establecida entre la SOCIEDAD y los Usuarios."
>
> (ZITY , 28 de abr. de 2018)

Este párrafo, extraído de la página de la política de privacidad de ZITY, nos confirma que con el uso de la aplicación y el sitio web recaban datos de los usuarios para analizarlos (hacer un tratamiento de estos) con el fin de gestionar mejor a sus usuarios.

En este documento también se comenta el fin con el que se recogen los datos de una manera más explicita, la legitimación para el tratamiento de estos, el tiempo que los conservan y los derechos del usuario cuando se les facilita estos datos. De esta forma, es adecuado destacar que estos datos que recogen también los usan para hacer perfiles de sus usuarios.

> "Elaboración de perfiles de conducción y/o modelos predictivos, con la finalidad de optimizar el Servicio, personalizar actividades de publicidad y prospección comercial y posibilitar la contratación de coberturas adicionales en el seguro del Servicio."
>
> (ZITY , 28 de abr. de 2018)

Para hacer este análisis, esta empresa utiliza los servicios de otra especializada, en este caso **Ridecell, Inc.**, la cual se localiza en Estados Unidos.

[6] `https://zitycar.es/politica-de-privacidad/`, 28 de abr. de 2018

> "Gestión de la plataforma del Servicio, en el marco de la cual los datos pueden ser transferidos internacionalmente a un encargado del tratamiento ubicado en EE.UU (Ridecell, Inc.), el cual proporciona garantías contractuales apropiadas en base a la normativa europea."
>
> (ZITY , 28 de abr. de 2018)

Tras este breve análisis podemos confirmar que la aplicación ZITY cumple con los estándares que marca el RGPD a la hora del tratamiento de los datos.

Con esto en mente, nuestro objetivo es crear una aplicación similar, pero usando las herramientas que nos proporciona FIWARE.

6.1. Explicación del Supuesto

Como se ha indicado en el punto anterior, se desea crear una aplicación donde los usuarios puedan registrarse e identificarse y donde estos, los usuarios que están identificados, puedan alquilar vehículos en el momento, seleccionado de entre los varios que se muestran en un mapa, para que los puedan usar para desplazarse por la ciudad. Para esto, usaremos diversas tecnologías, exactamente Orion y Laravel.

Como se ha expresado antes, para que una aplicación sea considerada FIWARE, ha de usar, al menos, el *Context Brocker* Orion. Para desarrollar la aplicación nos valdremos del *framework* para PHP Laravel[24]. Como se a indicado con anterioridad, ya estoy familiarizado a trabajar con Laravel, y posee gran cantidad de documentación proporcionada por los propios creadores del framework a mayores de repositorios en github de creadores independientes que nos pueden ayudar a la hora de desarrollar el prototipo.

Cuando se empezó con el prototipo, se decidió hacer una integración completa con la Rest API de Orion para usar esta como base de datos de la aplicación. Cuando se necesitó crear las sesiones para que los usuarios pudieran reservar vehículos se planteó la necesidad de crear una base de datos externa para guardar los datos necesarios para la sesión. De esta manera se explican 3 supuestos distintos dependiendo del uso para guardar los datos que se haga de Orion.

Independientemente del supuesto que sea, nos propondremos unos objetivos a cumplir del RGPD, estos de muestran en el Cuadro 26 y su continuación el Cuadro 27, y en los Cuadros 28 y 29 se muestra cómo vamos a resolverlos en nuestra aplicación.

Otro de las cosas que el RGPD tiene muy en cuenta es la protección de datos desde el diseño, en este caso, es una de las premisas principales para el desarrollo de este proyecto, es tener en cuenta la protección de los datos de los usuarios según el RGPD a la hora de diseñar una aplicación FIWARE, por lo que este punto le cumplimos por la propia definición de este trabajo.

Se ha decidido escoger estos derechos en particular puesto que se han creído los más relevantes a la hora de realizar la aplicación.

También, independientemente del nivel de integración con Orion escogido, la aplicación necesitará cumplir una serie mínima de requisitos explicados a continuación.

6.2. Funcionalidad requerida para el prototipo

Los requisitos detallados en las siguientes secciones nos ayudaran a entender que necesidades tiene la aplicación que vamos a desarrollar.

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.

Cuadro 26: Definición de los puntos del RGPD que tratamos en nuestra aplicación

Derecho del RGPD	Significado
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 que tenemos sobre su uso de los vehículos.
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.

Cuadro 27: Definición de los puntos del RGPD que tratamos en nuestra aplicación (Continuación)

Derecho del RGPD	Cómo resolverlo
Derecho de Acceso.	En el prototipo presentado, el tratamiento de datos que se realizara sera para saber cuando se usan los vehículos y de donde se recogen y a donde van a parar.
Derecho de Rectificación.	Si un usuario cancela la cuenta bancaria (o tarjeta) que utiliza para efectuar los pagos del alquiler de los vehículos esta debe poderse modificar.
Derecho de Cancelación	En el prototipo solo se recogen los mínimos datos necesarios para el funcionamiento de esta, a excepción de la cuenta bancaria, que será necesaria para el desarrollo completo de la aplicación, aunque ahora no se hace nada con ella.
Derecho de Oposición	En este caso, los datos que se recogen no se van a utilizar con fines publicitarios ni para la toma de decisiones que le afecten ya que estos datos se emplean principalmente para la facturación o para tramitar infracciones de tráfico.

Cuadro 28: Cómo resolvemos los puntos del RGPD en nuestra aplicación

Derecho del RGPD	Cómo resolverlo
Derecho al Olvido	En este caso, cuando el usuario decida darse de baja de la aplicación, los datos que tenemos almacenados en la base de datos con respecto a el serán eliminados o anominizados, cumpliendo un tiempo mínimo para estos datos por problemas de carácter judicial. En el prototipo, estos datos serán borrados de manera inmediata.
Derecho a la Portabilidad	En el prototipo que se presente este punto no se ha desarrollado puesto que sería necesario encontrar un formato coherente y estructurado de todos esto datos, aunque si se deja para futuras iteraciones del mismo.
Limite en el tiempo de conservación de datos	Los datos que recogemos de los usuarios serán utilizados hasta que éstos se den de baja, y los que se guarden serán con fines estadísticos habiendo sido anonimizados.

Cuadro 29: Cómo resolvemos los puntos del RGPD en nuestra aplicación (Continuación)

6.2.1. Requisitos funcionales

Los Requisitos Funcionales (RF) van a definir el funcionamiento del sistema, cómo comportarse. A continuación se definen los distintos requisitos funcionales que requiere la aplicación.

- **RF-01**: La aplicación permitirá a los usuarios registrarse.
- **RF-02**: La aplicación permitirá a los usuarios identificarse.
- **RF-03**: La aplicación permitirá a los usuarios ver un mapa de la zona con los vehículos disponibles.
- **RF-04**: La aplicación permitirá a los usuarios ver los detalles de los diversos vehículos.
- **RF-05**: La aplicación permitirá a los usuarios identificados reservar vehículos.
- **RF-06**: La aplicación permitirá a los usuarios identificados modificar su correo electrónico.
- **RF-07**: La aplicación permitirá a los usuarios identificados modificar su contraseña.
- **RF-08**: La aplicación permitirá a los usuarios identificados modificar su cuenta bancaria.
- **RF-09**: La aplicación permitirá a los usuarios identificados cambiar su número de teléfono.
- **RF-10**: La aplicación permitirá a los usuarios identificados cerrar su sesión.
- **RF-11**: La aplicación permitirá a los usuarios identificados darse de baja de esta.
- **RF-12**:La aplicación borrará los datos de los usuarios que se den de baja.

6.2.2. Requisitos No Funcionales

Los Requisitos No Funcionales, RNF, son requisitos que determinan cómo debe funcionar el sistema, sus características. Estas son cosas como el tiempo de respuesta, facilidad de aprendizaje para usarla, rendimiento...

Existen tres tipos de Requisitos No Funcionales.

- Requisitos del producto: Especifican el comportamiento, cosas referidas al rendimiento.
- Requisitos de la organización: Especifican políticas de procedimiento de la organización. Estas suelen venir dadas como estándares, como por ejemplo para los datos de una base de datos.
- Requisitos externos. Especifican factores externos al sistema, referido sobre todo a factores legales y éticos.

A continuación se muestras los Requisitos No Funcionales de la aplicación.

- **RNF-01**: La aplicación deberá cumplir con los derechos escogidos del RGPD definidos en las tablas 26 y 27.
- **RNF-02**: La aplicación debe ser fácil de usar, llevando a un usuario con un nivel medio de conocimiento menos de una hora en poder manejarse por el sistema.

- **RNF-03**: La aplicación podrá ser ejecutada por cualquier navegador que use HTML 4.0 o superior.
- **RNF-04**: La aplicación deberá conectarse a Orion que cumplirá una función de base de datos.
- **RNF-05**: La aplicación no consumirá demasiados recursos en el lado del usuario.
- **RNF-06**: La aplicación no consumirá demasiados recursos en el lado del servidor.
- **RNF-07**: La aplicación cambiará entre ventanas en un tiempo aceptable.

6.2.3. Requisitos de Información

Los Requisitos de Información (RI) son los que se establecen a partir de los datos que queremos almacenar.

- **RI-01**: La aplicación deberá almacenar información de los usuarios.
  - Nombre.
  - Apellidos.
  - DNI.
  - E-Mail.
  - Contraseña.
  - Número de Teléfono.
  - Cuenta Bancaria.
- **RI-02**: La aplicación deberá almacenar información de los vehículos.
  - Matrícula.
  - Marca.
  - Modelo.
  - número de plazas.
  - Batería.
  - Autonomía.
  - Disponible.
  - Localización.
- **RI-03**: La aplicación deberá almacenar información de los viajes realizados por los usuarios.
  - id-Usuario.
  - id-Vehículo
  - Fecha.
  - Kilómetros recorridos.
  - Tiempo usado.
  - Punto de Origen.
  - Punto de Fin.

6.3. Diferentes modelos para el supuesto

Como se ha indicado antes, a la hora de desarrollar el supuesto, surgieron varias ideas dependiendo del nivel de integración de Orion como base de datos en el prototipo. De esta manera, los tres modelos se clasifican como integración completa, media o mínima con Orion.

Todos van a tener que manejar los mismo datos, aunque de diferente manera. Estos datos serán los siguientes.

- **Person**: Representa a los usuarios registrados en la aplicación. Su identificador sera el correo electrónico.
- **Trip**: Representa los viajes que hacen los usuarios con los vehículos. Un usuario puede realizar varios viajes y un vehículo puede haber realizado varios viajes, pero cada viaje es de un solo vehículo y usuario. Dado a las limitaciones de usar Orion y una base de datos Mongodb, el identificador del viaje estará relacionado con el identificador del usuario para facilitar las búsquedas con respecto a los usuarios.
- **Car**: Representa a los vehículos que actualmente pueden usarse o están siendo usados. Su identificador es la matrícula del vehículo en cuestión.

6.3.1. Integración completa con Orion: Una sola empresa (Supuesto inicial)

Se planea hacer una aplicación web desde la cual los usuarios puedan registrarse e identificarse para que estos, los usuarios identificados, puedan alquilar un vehículo para moverse por la ciudad y, al finalizar el trayecto con el vehículo, cobrarles en consecuencia con el tiempo de uso. Los datos que se almacenen de estos trayectos serán posteriormente tratados para establecer perfiles generales de uso de los usuarios de la aplicación estimando fechas y horas de uso pico de los vehículos para evitar escasez de esto y que todos los usuarios puedan tener acceso a vehículos cerca de la zona donde lo van a usar. Este tratamiento de datos será completamente anonimizado puesto que solo nos interesa el hecho de si ha sido usado, en que franja horaria y en que fecha y donde ha sido recogido el vehículo y donde ha sido dejado, con el fin de poder establecer ciertos patrones de uso de la aplicación a la par que posibles zonas donde se recogen muchos vehículos y zonas donde estos van a parar.

Los datos solo se guardaran mientras el usuario este registrado en la aplicación y un tiempo prudencial desde que este se dé de baja como registro por posibles problemas judiciales.

Este supuesto tiene varias ventajas a la hora de cumplir con el RGPD y manteniendo una base de datos muy sencilla como se muestra en la Figura 9.

6.3.2. Integración media con Orion

En este supuesto lo que queremos hacer es una aplicación para una empresa que tiene los datos de los vehículos, usuarios y viajes guardados usando Orion, mientras que nosotros ofreceremos una página web para estos usuarios ya registrados para que puedan reservar los vehículos. En este supuesto se usara una segunda base de datos que se encargará de los datos necesarios para guardar las sesiones, por lo necesitamos se capaces de tomar solo los datos que necesitamos de los usuarios para establecer la sesión, siendo estos **id** y **password**. Por su parte, la base de datos sería como se muestra en la Figura 10

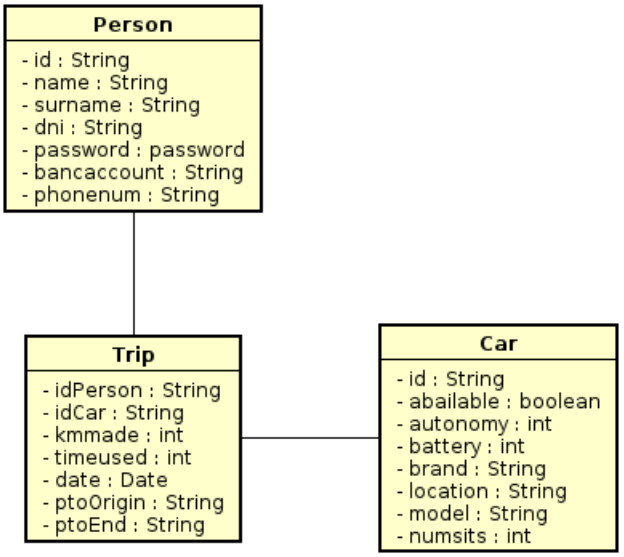

Figura 9: Modelo de datos para la integración completa

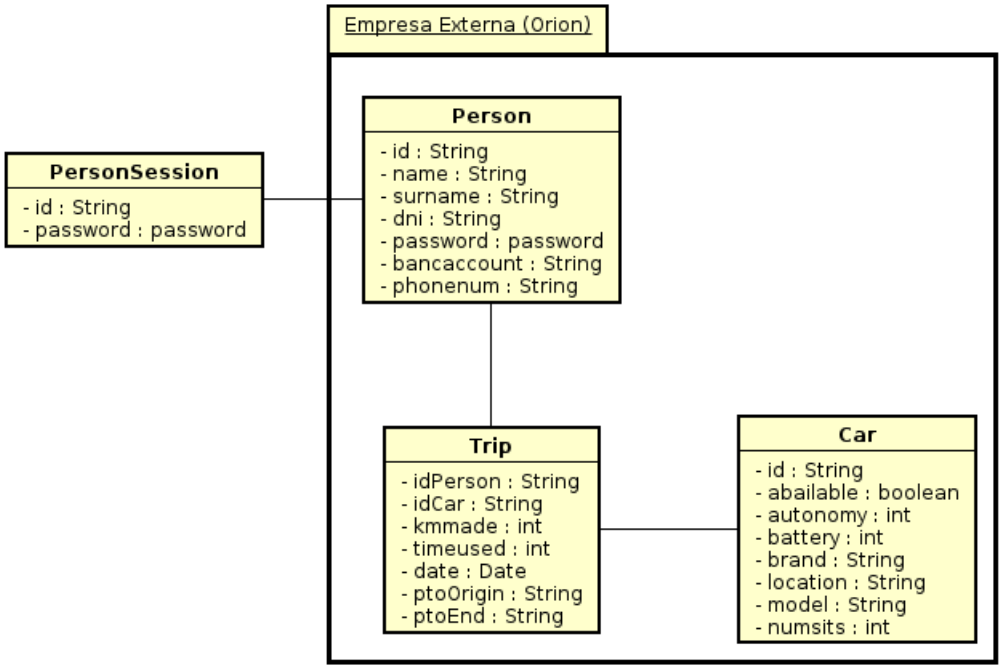

Figura 10: Modelo de datos para la integración media

En este caso la única diferencia con el modelo anterior es la existencia de una nueva tabla, fuera del sistema Orion, que solo guardará los datos necesarios para iniciar la sesión.

En comparación con el caso anterior, la base de datos gana complejidad, y se necesitan hacer una operación de tratamiento para tomar solo los datos necesarios de Orion. Para esto Orion nos proporciona las herramientas adecuadas a la hora de realizar la llamada. Esta aproximación nos permitiría fácilmente crear las sesiones.

6.3.3. Integración mínima con Orion: Dos empresas separadas

En este supuesto seriamos una empresa que ha llegado a un acuerdo con otra para usar sus vehículos. Para esto necesitan guardar la información de los viajes que realizamos para poder realizar la facturación, nosotros por el mismo motivo nosotros tenemos que guardar la información de los viajes realizados por nuestros usuarios. El tratamiento de los datos que se realiza en este punto, como se puede observar en la Figura 11 es la eliminación de quien ha realizado el viaje, lo cual viene dado por **idUser**.

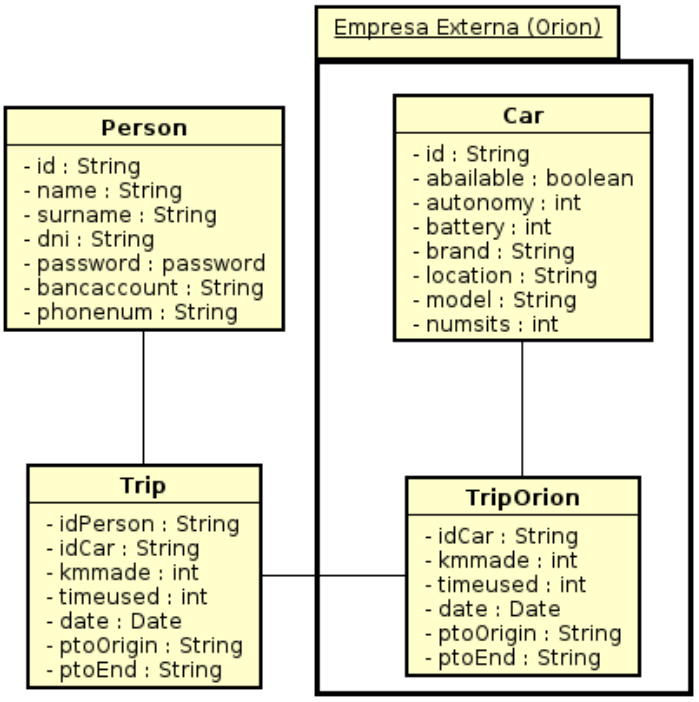

Figura 11: Modelo de datos para la integración mínima

En este caso, la base de datos es la más compleja de las tres, pero a cambio ofrece un gran control sobre los datos que se transmiten, por lo que nos permite cumplir el requisito de mínimo paso de información necesario. Esta aproximación nos permite también crear sesiones de manera sencilla, pues los datos de la sesión están guardados en nuestra base de datos.

6.4. Desarrollo del prototipo

Tras esta explicación de los diferentes supuestos que hay para el desarrollo de la aplicación decidimos seleccionar el supuesto llamado **Integración completa con Orion**, puesto que es la

que, ofreciendo gran capacidad para la protección de los datos según los estándares del RGPD, es la menos compleja a la hora de construirla.

6.4.1. Arquitectura básica

A la hora de crear la arquitectura del modelo nos basamos en el patrón Modelo Vista Controlador (MVC). Este está modificado para adaptarse al uso de Orion, el cual será nuestro Modelo.

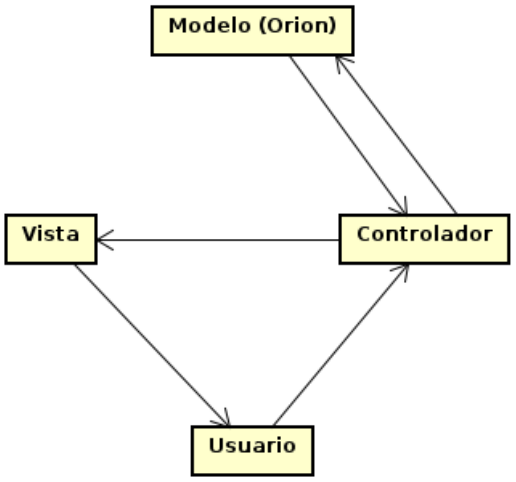

Figura 12: Patrón Modelo Vista Controlador empleado

Como se muestra en la Figura 12, tanto las vistas como los controladores se manejan desde la propia aplicación, mientras que el modelo es Orion, una Rest API que es externa a la propia aplicación.

Como en la aplicación se va a mostrar donde se encuentran los vehículo, necesitamos tener un mapa para que se muestre, para ello necesitamos tener acceso a la API de Google para los mapas, también necesitaremos una página donde los usuarios puedan registrarse e identificarse. Para esta iteración del desarrollo se planea permitir a los usuarios identificados reservar el coche que hayan seleccionado en el mapa, por ello la opción de reserva estará situada en los detalles del vehículo. En la Figura 13 se puede observar una primera iteración de este modelo.

El id de **Person** será su email. De esta manera nos aseguramos que sea un identificador único. De manera similar, el id de ***Car*** será la matrícula de este. Para ***Trip***, el id que usaremos en Orion sera el siguiente

```
<idPerson>-Trip<numero>
```

Donde el numero vendrá dado por la cantidad de viajes que haya realizado la persona en cuestión. Se hace de esta manera porque si la persona se da de baja, resultará mucho más fácil buscar estos datos, siguiendo la estructura, para poder eliminarlos. Un ejemplo practico de esto seria: **ejemplo@ejemplo.com-Trip0**, donde **ejemplo@ejemplo.com** sería el id del usuario y el número al final indica el viaje, siendo el primero el número 0.

Como en la realidad se esta usando una base de datos tipo Mongo a través de Orion, es necesario tener claro cuales van a ser los identificadores de los diferentes datos, sobre todo a la

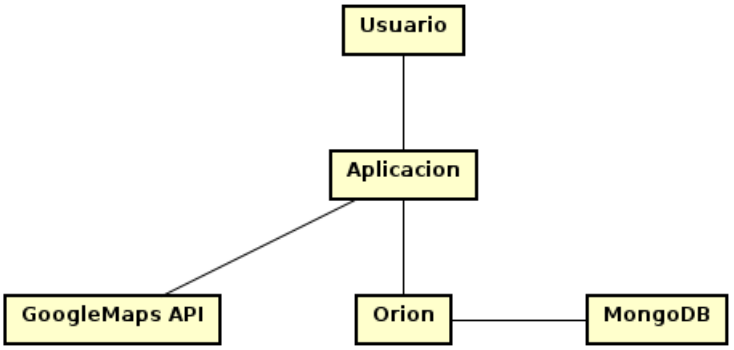

Figura 13: Primera iteración del desarrollo de la aplicación

hora de buscar los datos con las llamadas desde el programa, por eso es importante explicar como se planean construir.

Si ampliamos la parte de la Aplicación, podemos ver con mejor detalle como se conectan estos datos unos con otros. Como se puede observar en la Figura 14, realmente la aplicación está constituida por tres partes importantes, el encargado de los mapas que se comunicará directamente con *Google Maps* API, el encargado de las reservas que se comunicará directamente con Orion para acceder a los datos del vehículo y otro de Registro/Identificación que se comunicará también con Orion para tomar y almacenar datos de los usuarios.

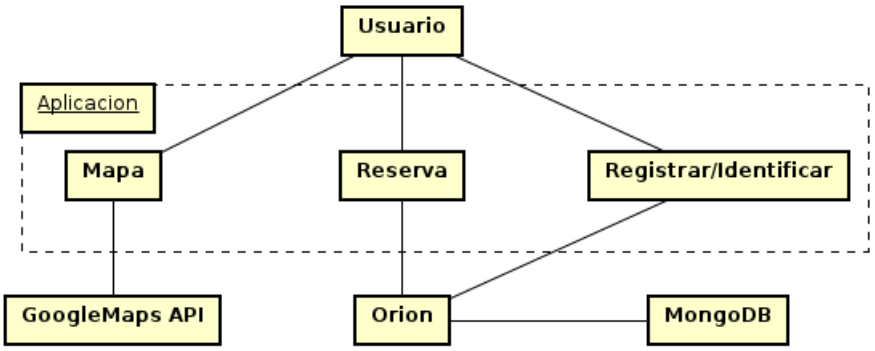

Figura 14: Segunda iteración del desarrollo de la aplicación

7. Desarrollo de pruebas

7.1. Introducción

A la hora de desarrollar la aplicación se desarrollan también las pruebas que esta va a tener que superar de manera exitosa, esto es que la salida esperada y la obtenida sea la misma, para considerar que es una aplicación robusta.

Para esto se hará un listado con los distintos Casos de Prueba (CP) describiendo cada uno de ellos, las entradas que recibe y las salidas esperada y obtenida.

7.2. Pruebas de funcionamiento general

CP-01	Registrar a un nuevo usuario.
Descripción	Registrar un usuario que no esta registrado en la base de datos.
Entrada	Nombre: Roberto, Apellido1: Bahillo, Apellido2: Ortego, DNI: 71957429C, E-Mail: roberto.bahillo@alumnos.uva.es, Contraseña: 1234, Confirmar contraseña:1234, Cuenta Bancaria:es41896413894652, Número de Teléfono:111111111.
Salida esperada	Registrado un nuevo usuario con los datos de entrada.
Salida obtenida	Salida esperada.

Cuadro 30: CP-01: Registrar un nuevo usuario.

CP-02	Registrar a un usuario previamente registrado.
Descripción	Registrar un usuario que ya esta registrado en la base de datos.
Entrada	Nombre: Roberto, Apellido1: Bahillo, Apellido2: Ortego, DNI: 71957429C, E-Mail: roberto.bahillo@alumnos.uva.es, Contraseña: 1234, Confirmar contraseña:1234, Cuenta Bancaria:es41896413894652, Número de Teléfono:111111111, acepta los Términos y Condiciones de Uso.
Salida esperada	Aviso de que el usuarios ya existe.
Salida obtenida	Salida esperada.

Cuadro 31: CP-02: Registrar un usuario previamente registrado.

Para el CP-05 que aparece en la Tabla 34, se tiene como prerrequisito que el usuario ya exista.

CP-03	Registrar a un usuario faltando datos.
Descripción	Registrar un usuario faltando por rellenar una o varias casillas del formulario.
Entrada	Nombre: Roberto, Apellido1: , Apellido2: Ortego, DNI: , E-Mail: roberto.bahillo@alumnos.uva.es, Contraseña: 1234, Confirmar contraseña:1234, Cuenta Bancaria:es41896413894652, Número de Teléfono:111111111, aceptar los Términos y Condiciones de Uso.
Salida esperada	Se muestra un mensaje de error indicando las casillas que faltan por rellenar.
Salida obtenida	Salida esperada.

Cuadro 32: CP-03: Registrar un usuario faltando datos.

CP-04	Registrar a un usuario sin aceptar los Términos y Condiciones de Uso.
Descripción	Registrar un usuario que ya esta registrado en la base de datos.
Entrada	Nombre: Roberto, Apellido1: Bahillo, Apellido2: Ortego, DNI: 71957429C, E-Mail: roberto.bahillo@alumnos.uva.es, Contraseña: 1234, Confirmar contraseña:1234, Cuenta Bancaria:es41896413894652, Número de Teléfono:111111111, no aceptar los Términos y Condiciones de Uso.
Salida esperada	Se muestra el mensaje de error “The terms must be accepted”.
Salida obtenida	Salida esperada.

Cuadro 33: CP-04: Registrar un usuario previamente registrado.

CP-05	Identificarse como usuario.
Descripción	Identificarse en la aplicación.
Entrada	E-Mail:roberto.bahillo@alumnos.uva.es, Contraseña: 1234.
Salida esperada	Te identificas en la aplicación.
Salida obtenida	Salida esperada.

Cuadro 34: CP-05: Identificarse como usuario.

CP-06	Mostrar mapa.
Descripción	Mostrar el mapa con los vehículos disponibles.
Entrada	
Salida esperada	La vista del mapa con marcadores representando los vehículos disponibles, estando centrado este en la posición del usuario.
Salida obtenida	Salida esperada.
Salida alternativa	Dado que necesita la geolocalización, esto no será posible en todos los casos dado que el usuario puede tenerla bloqueada, en cuyo caso el mapa no mostrará nada. Si se decide usar una localización preestablecida para entrar el mapa, como es el caso de la escuela, algunas veces el mapa se centrará en las coordenadas 0,0 (null island).

Cuadro 35: CP-06: Mostrar mapa.

CP-07	Detalles del vehículo.
Descripción	Al hacer doble click sobre un marcador, aparecerá una ventana emergente mostrando los detalles del vehículo con una foto de este.
Entrada	
Salida esperada	Ventana emergente con los detalles del vehículo.
Salida obtenida	Salida esperada.
Salida alternativa	Dependiendo del navegador usado y de si el usuario permite las ventanas emergentes, esta salida puede modificarse como una nueva pestaña o no aparecer.

Cuadro 36: CP-07: Detalles el vehículo.

El caso de prueba 08 tiene como prerrequisito que el usuario no esté identificado en la aplicación.

CP-08	Reservar vehículo sin estar identificado.
Descripción	El usuario intentara reservar un vehículo sin haber iniciado sesión en la aplicación.
Entrada	
Salida esperada	En la ventana de detalles del vehículo no aparecerá el botón de reserva si no un enlace a identificarse.
Salida obtenida	Salida esperada.

Cuadro 37: CP-08: Reservar vehículo sin estar identificado.

El caso de prueba 09 tiene de prerrequisito que el usuario tenga una sesión iniciada en la aplicación

CP-09	Reservar vehículo estando identificado.
Descripción	El usuario intenta reservar un vehículo.
Entrada	
Salida esperada	En la ventana de detalles del vehículo aparecerá un botón que permitirá reservar el vehículo, al pulsarlo el estado de este pasa a no estar disponible.
Salida obtenida	Salida esperada.

Cuadro 38: CP-09: Reservar vehículo estando identificado.

CP-10	Introducir los detalles del viaje correctamente.
Descripción	Introducir los detalles del viaje, Kilómetros recorridos, tiempo empleado y las coordenadas.
Entrada	Kilómetros hecho: 15, Tiempo usado(min): 4, Punto Final (indica la linea en el archivo coordenadas.txt, empieza en 0): 0.
Salida esperada	Se crea una nueva entidad tipo Trip con los datos indicados, las coordenadas se recogen un archivo externo, más los del vehículo usado, fecha actual y punto de origen que recoge automáticamente el sistema.
Salida obtenida	Salida esperada.

Cuadro 39: CP-10: Introducir los detalles del viaje correctamente.

CP-11	Introducir los detalles del viaje con datos faltantes.
Descripción	Introducir los detalles del viaje, Kilómetros recorridos, tiempo empleado y las coordenadas.
Entrada	Kilómetros hecho: 15, Tiempo usado(min): , Punto Final (indica la linea en el archivo coordenadas.txt, empieza en 0): 0.
Salida esperada	Se muestra un mensaje de error indicando los datos que faltan.
Salida obtenida	Salida esperada.

Cuadro 40: CP-11: Introducir los detalles del viaje correctamente.

CP-12	Introducir los detalles del viaje con un número en Punto Final mayor que el número de líneas.
Descripción	Introducir los detalles del viaje, Kilómetros recorridos, tiempo empleado y las coordenadas.
Entrada	Kilómetros hecho: 15, Tiempo usado(min): 4, Punto Final (indica la linea en el archivo coordenadas.txt, empieza en 0): 1000.
Salida esperada	Se crea una nueva entidad tipo Trip con los datos indicados, las coordenadas se recogen un archivo externo (escoge automáticamente la última coordenada del fichero), más los del vehículo usado, fecha actual y punto de origen que recoge automáticamente el sistema.
Salida obtenida	Salida esperada.

Cuadro 41: CP-12: Introducir los detalles del viaje correctamente.

Los casos de pruebas del 13 al 22 tienen de prerrequisito que el usuario tenga una sesión iniciada.

CP-13	Modificar el E-Mail con un nuevo E-Mail.
Descripción	Modificar el E-Mail del usuario y los datos pertinentes para la tabla viajes.
Entrada	Nuevo E-Mail: roberbahillo@yo.es.
Salida esperada	Actualización del E-Mail para el usuario y los viajes que este ha realizado.
Salida obtenida	Salida esperada

Cuadro 42: CP-13: Modificar E-Mail

CP-14	Modificar el E-Mail con el mismo E-Mail.
Descripción	Modificar el E-Mail del usuario y los datos pertinentes para la tabla viajes.
Entrada	Nuevo E-Mail: roberbahillo@yo.es.
Salida esperada	Actualización del E-Mail para el usuario y los viajes que este ha realizado.
Salida obtenida	Salida esperada

Cuadro 43: CP-14: Modificar el E-Mail con el mismo E-Mail

CP-15	Modificar contraseña.
Descripción	Modificar la contraseña.
Entrada	Nueva Contraseña: 4567, Confirmar Contraseña:4567.
Salida esperada	Se actualiza la contraseña.
Salida obtenida	Salida esperada.

Cuadro 44: CP-15: Modificar contraseña.

CP-16	Modificar contraseña con la misma contraseña anterior.
Descripción	Modificar la contraseña.
Entrada	Nueva Contraseña: 4567, Confirmar Contraseña:4567.
Salida esperada	Se actualiza la contraseña.
Salida obtenida	Salida esperada.

Cuadro 45: CP-16: Modificar contraseña con la misma contraseña anterior.

CP-17	Modificar la cuenta bancaria.
Descripción	Modificar la cuenta bancaria.
Entrada	Nueva Cuenta Bancaria: es41896413894987.
Salida esperada	Se actualiza la cuenta bancaria.
Salida obtenida	Salida esperada.

Cuadro 46: CP-17: Modificar la cuenta bancaria.

CP-18	Modificar la cuenta bancaria con la misma cuenta bancaria anterior.
Descripción	Modificar la cuenta bancaria.
Entrada	Nueva Cuenta Bancaria: es41896413894987.
Salida esperada	Se actualiza la cuenta bancaria.
Salida obtenida	Salida esperada.

Cuadro 47: CP-18: Modificar la cuenta bancaria con la misma cuenta bancaria anterior.

CP-19	Modificar el número de teléfono.
Descripción	Modificar el número de teléfono.
Entrada	Nueva número de Teléfono: 222222222.
Salida esperada	Se actualiza el número de teléfono.
Salida obtenida	Salida esperada.

Cuadro 48: CP-19: Modificar el número de teléfono.

CP-20	Modificar el número de teléfono con el mismo número de teléfono.
Descripción	Modificar el número de teléfono.
Entrada	Nueva número de Teléfono: 222222222.
Salida esperada	Se actualiza el número de teléfono.
Salida obtenida	Salida esperada.

Cuadro 49: CP-20: Modificar el número de teléfono con el mismo número de teléfono.

CP-21	Cerrar sesión.
Descripción	Se cerrará la sesión y se borraran los datos de la misma.
Entrada	
Salida esperada	Se cierra la sesión.
Salida obtenida	Salida esperada.

Cuadro 50: CP-21: Cerrar sesión.

CP-22	Darse de baja.
Descripción	Se borrarán todos los datos pertenecientes al usuario y que estén relacionados con este.
Entrada	
Salida esperada	Se borran los datos del usuario y sus viajes.
Salida obtenida	Salida esperada.

Cuadro 51: CP-22: Darse de baja.

7.3. Pruebas de seguridad de la aplicación

Debido a cómo funciona Laravel, no se puede acceder a una llamada POST sin el uso de **{{csrf_field()}}**. Desde su versión 5.x, Laravel tiene habilito por defecto el *middleware VerifyCsrfToken* destinado a la protección contra ataques CSRF[7]. Dada la necesidad de este token, los usuarios no pueden forzar una llamada tipo POST que le brinde acceso a datos que no debería tener.

CP-23	Funcionamiento POST de los formularios.
Descripción	En ningún formulario de tipo POST debe aparecer los datos que se pasan, para este ejemplo se prueba a la hora de identificar al usuario.
Entrada	E-Mail:roberbahillo@yo.es, Contraseña: 1234.
Salida esperada	Se comprueba que los datos que se envían a través de la llamada POST no se muestran en la URL.
Salida obtenida	Salida esperada.

Cuadro 52: CP-23: Funcionamiento POST de los formularios.

[7]Cross-site request forgery[31]

CP-24	Comprobación del tiempo máximo de la sesión.
Descripción	Verificar que el tiempo máximo de la sesión es de 5 minuto.
Entrada	Iniciar sesión con unas credenciales correctas.
Salida esperada	Tras haber pasado los 5 minutos, los datos de la sesión son eliminados.
Salida obtenida	Salida esperada.

Cuadro 53: CP-24: Comprobación del tiempo máximo de la sesión.

CP-25	Usuario y/o contraseña no validos.
Descripción	Los datos del usuario y/o la contraseña no son validos para iniciar una sesión.
Entrada	E-Mail:correoincorrecto@incorrecto.com, Contraseña: 1234.
Salida esperada	Se devuelve el mensaje de error "E-Mail o Contraseña incorrectos".
Salida obtenida	Salida esperada.

Cuadro 54: CP-25: Usuario y/o contraseña no validos.

CP-26	Acceso a rutas invalidas.
Descripción	Se accede a una ruta invalida, por ejemplo escribir en la url virtual.lab.inf.uva.es:20052/CarRenting/verify. En este caso se está haciendo una llamada GET, y aunque su llamada POST este definida en **web.php**, Laravel lo trata como si de dos llamadas distintas se tratara.
Entrada	
Salida esperada	Se captura el error y se muestra una página que muestre el número de error adecuado (en este caso debería ser un error 404 página no encontrada).
Salida obtenida	Una página de error donde se muestra la línea que ha lanzado la excepción y el estado de la máquina, esta se ve más detallada en la Figura37.

Cuadro 55: CP-26: Funcionamiento POST de los formularios.

8. Conclusiones y Trabajo Futuro

8.1. Conclusiones

La entrada en vigor del RGPD plantea la revisión del tratamiento de datos personales de las aplicaciones aunque estas ya cumplieran lo establecido en la LOPD. Por otro lado, en muchas aplicaciones de *smartcities* suelen aparecer datos personales, cuyo tratamiento debe estar actualmente conforme a lo establecido en el RGPD.

Dado que FIWARE es una plataforma abierta para, entre otras el desarrollo de aplicaciones para *smartcities* en este trabajo se ha pretendido comprobar el cumplimiento o no del RGPD en el tratamiento de datos personales que plantean los *Generic Enablers* dedicados al tratamiento de datos, en concreto Orion Context Broker.

Con la manera en la que hemos trabajado con Orion, destaca una vulnerabilidad a la hora de buscar y traer datos independientemente de la base de datos usada. En este caso, Orion es accedido vía web, realizando peticiones a través de una url. Esto limita mucho las operaciones de búsqueda, haciendo que realmente resulte mucho más sencillo cambiar de base de datos, deshaciéndose de Orion, y realizar la integración con el proyecto.

Orion por si solo tampoco ofrece ninguna garantía de protección para los datos añadida, éstas vienen únicamente dadas por la máquina, puesto que si puedes acceder a la maquina de Orion, accediendo a /v2/entities, puedes acceder a todos los datos guardados.

Aun con todo, si que es cierto que FIWARE proporciona muchas herramientas que si podrían ser útiles para el desarrollo de otras aplicaciones ya que Orion es una base sobre la que asentar todo lo demás.

8.1.1. Objetivos alcanzados

Tras este desarrollo, se han podido comprobar los objetivos alcanzados.

- Se ha comprobado la versatilidad de FIWARE a la hora de crear una aplicación.
- Se ha creado una aplicación que ha cumplido con los la parte del RGPD que se ha decidido.
- La aplicación desarrollada se ha mantenido lo más simple posible para que esta sea fácil de mantener y seguir desarrollando y mejorando.
- Se ha demostrado que trabajar con FIWARE es posible y que Orion es muy versátil.
- Se ha comprobado lo preparado que esta Orion por si solo para manejar la privacidad y la seguridad de los usuarios siguiendo lo que nos dicta el Reglamento General de Protección de Datos.

8.2. Trabajo Futuro

- Una de las primeras cosas que viene es revisar con más detalle la documentación de Orion para poder sacarle partido a las cosas que ofrece, como por ejemplo a las subscripciones (subscriptions), las cuales nos podrían ayudar a la hora de actualizar los datos.

- Profundizar más en el registro de las personas, la verificación de la identidad de estas (mediante una foto del DNI) y comprobación automatizada del estado de la licencia de conducir de estas.
- Validación de correos electrónicos.
- Habilitar una pasarela de pago.
- Mediante la ayuda de FIWARE, surge la oportunidad de añadir detectores para medir la contaminación atmosférica en los vehículos, permitiendo, además del servicio ofrecido, generar un mapa de la ciudad con los niveles de contaminación atmosférica.
- Crear una aplicación para Sistemas Operativos móviles, principalmente Android y iOS, ya que son estos los más usados.
- Añadir en la vista del perfil del usuario un botón que le permita acceder a los viajes que ha realizado.
- Evitar que se muestre la página de error propia de Laravel cuando se intenta acceder a rutas no existentes y capturarlo mostrando el estado del acceso en PHP, en este caso, error 404.
- Colocar un marcador centrado en la posición del usuario.
- Crear una vista de supervisor para los datos como usuarios, viajes y vehículos, y hacer que se puedan ordenar.
- Si se sigue ampliando el controlador **OrionController**, dividirlo en 3 ó 4 controladores diferentes: uno especifico para los usuarios, otro para los vehículos y otro para los viajes.
- Solucionar el problema con los detalles de la geolocalización de la aplicación.

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] Zürcher Hochschule der Angewandten Wissenschaften y Stephan Neuhaus. «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] Jet Brains. *PhpStorm: Lightning-Smart IDE for PHP Programing by JetBrains*. 20 de jun. de 2018. URL: https://www.jetbrains.com/phpstorm/.

[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. «Firawe-Orion». En: (2 de jun. de 2018). URL: https://fiware-orion.readthedocs.io/en/master/index.html.

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

[14] fiware. «fiware/orion - Docker Hub». En: (2 de jun. de 2018). URL: https://hub.docker.com/r/fiware/orion/.

[15] Google. *Administrar Recursos*. 4 de jun. de 2018. URL: https://console.developers.google.com/cloud-resource-manager.

[16] Google. *Obtener una clave o autenticación | Google Maps Geocoding API | Google Developers*. 4 de jun. de 2018. URL: https://developers.google.com/maps/documentation/geocoding/get-api-key?hl=es-419.

[17] The PHP Group. «PHP: Documentation». En: (30 de jun. de 2018). URL: http://php.net/docs.php.

[18] 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.

[19] Mark Sagi Kazar. «Guzzle, PHP HTTP client — Guzzle Documentation». En: (7 de jun. de 2018). URL: http://docs.guzzlephp.org/en/latest/index.html.

[20] 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/.

[21] 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.

[22] Steve Marks. «CodeIgniter Google Maps V3 API Library | BIOSTALL». En: (3 de jun. de 2018). URL: http://biostall.com/codeigniter-google-maps-v3-api-library/.

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

[24] Taylor Orwel. «Laravel - The PHP Framework For Web Artisan». En: (1 de jun. de 2018). URL: https://laravel.com/.

[25] 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/.

[26] 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.

[27] 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/.

[28] 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.

[29] StatCounter Global Stats. En: (30 de jun. de 2018). URL: http://gs.statcounter.com/browser-market-share#monthly-201805-201805-bar.

[30] Farhan Wazir. «GitHub-farhanwazir/laravelgooglempas: Laravel Google Maps Package». En: (3 de jun. de 2018). URL: https://github.com/farhanwazir/laravelgooglemaps.

[31] Wikipedia. En: (5 de jul. de 2018). URL: https://es.m.wikipedia.org/wiki/Cross-site_request_forgery.

[32] ZITY. «Politica de privacidad - ZITY». En: (28 de abr. de 2018). URL: https://zitycar.es/politica-de-privacidad/.

A. Implementación del proyecto CarRenting

En esta parte de la memoria se explicará de manera más detallada los pasos seguidos para la implementación de este prototipo, así como problemas a la hora de la creación del prototipo. Como vamos a trabajar en PHP utilizaré el entorno de trabajo de phpStrorm[6] con el cual también estoy familiarizado.

A.1. Usando Laravel

Como se comento en el caso de uso, para el desarrollo del prototipo se ha decidido usar el *framework* para web artisan de Laravel. A continuación se planea profundizar en la cuestión de como se ha desarrollado el código paso a paso para el prototipo.

Para realizar el proyecto lo primero que hacemos, y como se indica en la página de documentación de Laravel, tras ser instalado, es crear un nuevo proyecto con la siguiente linea de código en un termina.

```
composer create-project --prefer-dist laravel/laravel
CarRenting
```

Con esto hemos creado un composer usando Laravel llamado **CarRenting**.

Para poder acceder en a la pagina que acabamos de crear en el terminal escribimos lo siguiente.

```
php artisan serve
```

Ahora podemos acceder a **localhost:8000** para comprobar que hemos creado correctamente el proyecto como se muestra en la Figura 15.

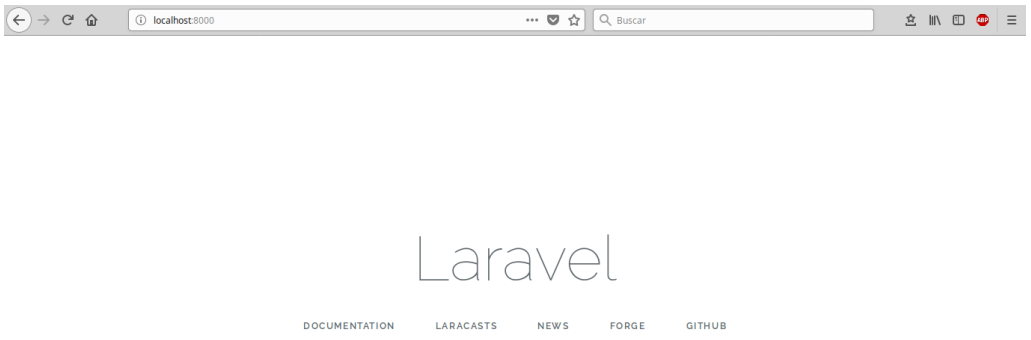

Figura 15: Página de inicio del proyecto recién creado

A.2. Integración de Google Maps en el proyecto

Para la aplicación de *Car Renting* es necesario usar un mapa.En este, los usuarios podrán seleccionar el vehículo que deseen de acuerdo a necesidades como la cercanía a este y el nivel de

batería entre otras, usando para esto Google Maps. Con el fin facilitar su uso, esto se realizara a través de un repositorio en GitHub [30]. También nos ayudaremos del código para el *framework* CodeIgniter [22], y aunque tanto el código que muestra de ejemplo como el manual son para el CodeIgniter y no para Laravel, podemos transformarlo fácilmente.

Para poder usar el servicio de **Google Maps** es necesario obtener una **API key**[15], para ello visitamos la página que se nos indica en el archivo **googlemaps.php** en la carpeta **config** de nuestro proyecto.

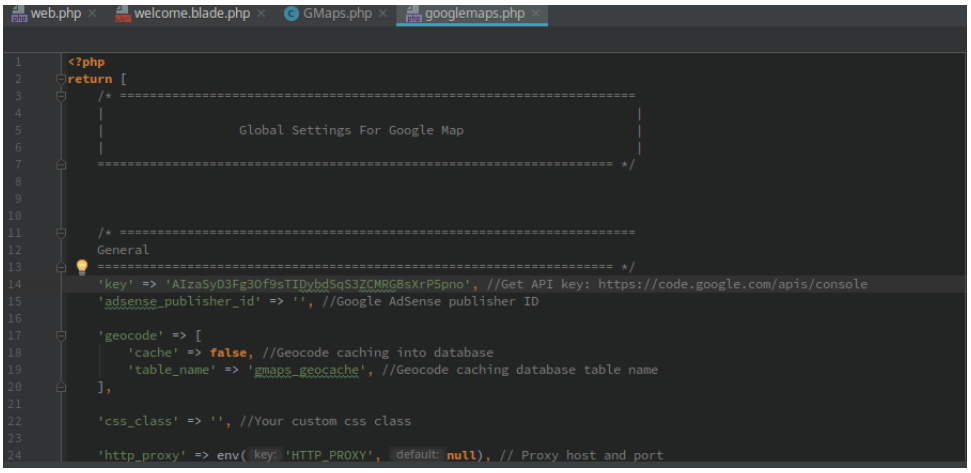

Figura 16: googlemaps.php

En las primeras lineas de código, como se muestra en la Figura 16, encontramos lo siguiente.

```
'key' => '...'
```

Esta clave es necesaria para poder emplear, en este caso, **googlemaps**. Para obtener esta clave para la API necesitamos primero crearnos un proyecto en la página que nos indica, y después habilitar las APIs que queremos usar. En nuestro caso hemos habilitado las siguientes:

- ***Geocoding API***: Convierte entre direcciones y coordenadas geográficas.
- ***Geolocalization API***: Localización de dispositivos a través de redes móviles y nodos wifi.
- ***Maps JavaScript API***: Maps para tu sitio web.

Tras haber habilitado estas APIs pedimos obtener la clave [16], seleccionamos nuestro proyecto y la copiamos en nuestro archivo **googlempas.php**. Si en algún momento tenemos que añadir más APIs habrá que actualizar esta clave.

Ahora para mostrar el mapa tenemos que añadir una variable llamada **$config** en **web.php**, que es el encargado de enrutar las diferentes llamadas hechas desde el proyecto, e inicializar el mapa pasándolo en una variable llamada **$map** a la vista. El archivo **web.php** esta localizado en la carpeta **routes**.

Como se puede apreciar en la Figura 17, **$config** es, en esta caso, un *array* de datos, nosotros vamos a usar los de ***center***, centramos el mapa en el edificio de la Escuela en el Campus (si

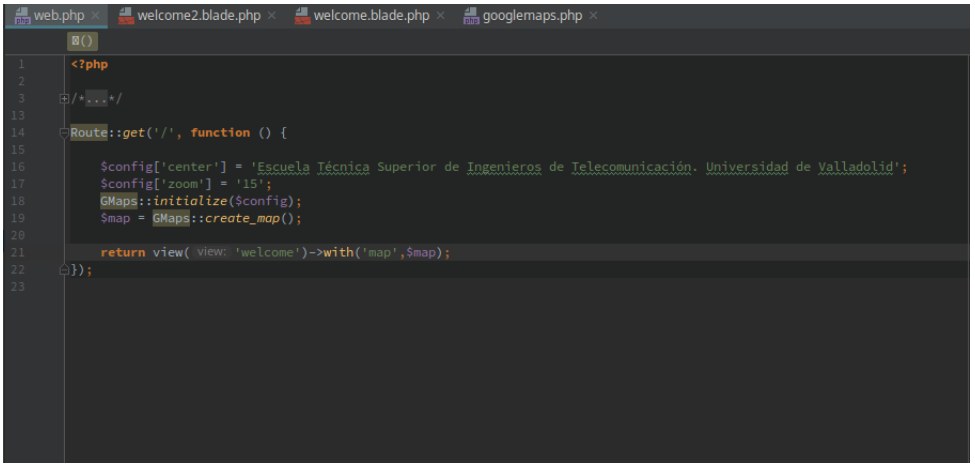

Figura 17: web.php

escribimos **'auto'** aparecerá automáticamente centrado en la localización[8] del usuario si este le da permiso para localizarle), y **zoom** que se lo hemos puesto a 15. Existen otras variables como las encargadas de alterar las dimensiones del mapa en la página web o si permitimos la posibilidad de hacer scroll entre otras. Tras haber configurado el mapa le creamos como dijimos antes con la variable **$map** y se los pasamos a la vista **map** que obviamente tenemos que modificar para que esta pueda interpretar el mapa.

En el archivo **map.blade.php** (en Laravel, los archivos **.blade.php** son *templates*, en este caso usados para las vistas) situado en la carpeta **resources/views**, eliminamos la etiqueta <**style**> y todo el contenido de <**body**>, y añadimos, como se observa en la Figura 18, en <**body**>,

```
{!! \$map['html']!!}
```

mientras que al final de <**head**>,

```
{!! \$map['js']!!}
```

De esta manera, y como se muestra en la Figura 19 ha aparecido centrado en el Campus Miguel Delibes, exactamente en el edificio de la Escuela de Ingeniería Informática y de Telecomunicaciones, donde le indicamos en **web.php**.

Tras esto vamos a añadirle marcadores al mapa, en nuestro caso los marcadores representarán los vehículos. Para ello usamos en **web.php** la variable **$maker** con los valores que se muestran en la Figura 20.

Como se observa en **'position'** esta puede ser tanto una dirección real o coordenadas. Ahora veremos como ha cambiado el mapa tras añadir estos nuevos puntos. Como se puede preciar en la Figura 21 ahora tenemos dos marcadores en rojo que representara dos vehículos estacionados, para mostrar el comentario hay que clickar en el marcador y este en su momento mostrará datos del coche importantes como su nivel de batería, autonomía estimada...

Al crear los marcadores los he añadido con coordenadas en vez de una dirección puesto que es con lo que el detector de los vehículos trabajará.

[8]Se ha observado que la geolocalización, la localización automática, no funciona correctamente en la aplicación, no se sabe si es problema de el código o de los dispositivos usados. Esta opción estará desactivada para el desarrollo de las pruebas y la demostración

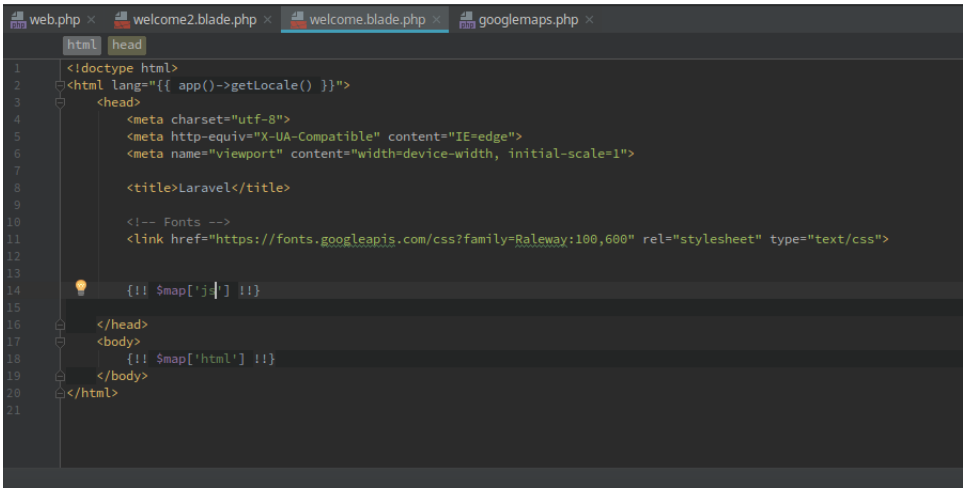

Figura 18: map.php

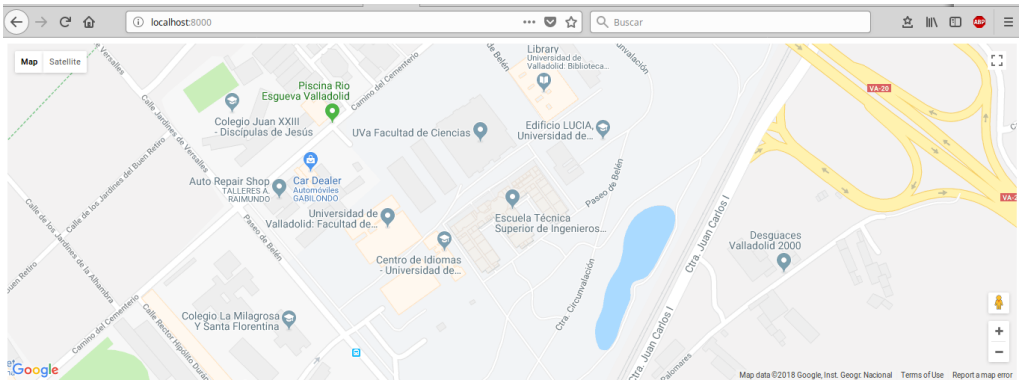

Figura 19: mapa.php

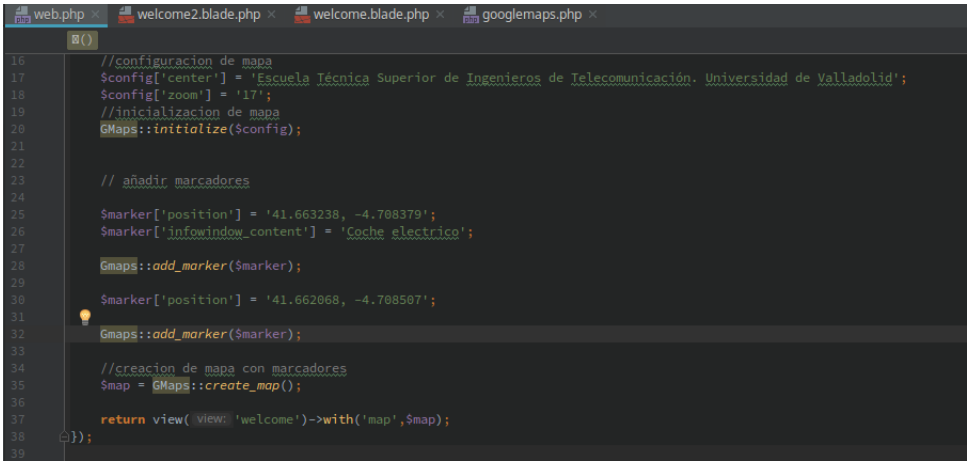

Figura 20: web.php con marcadores

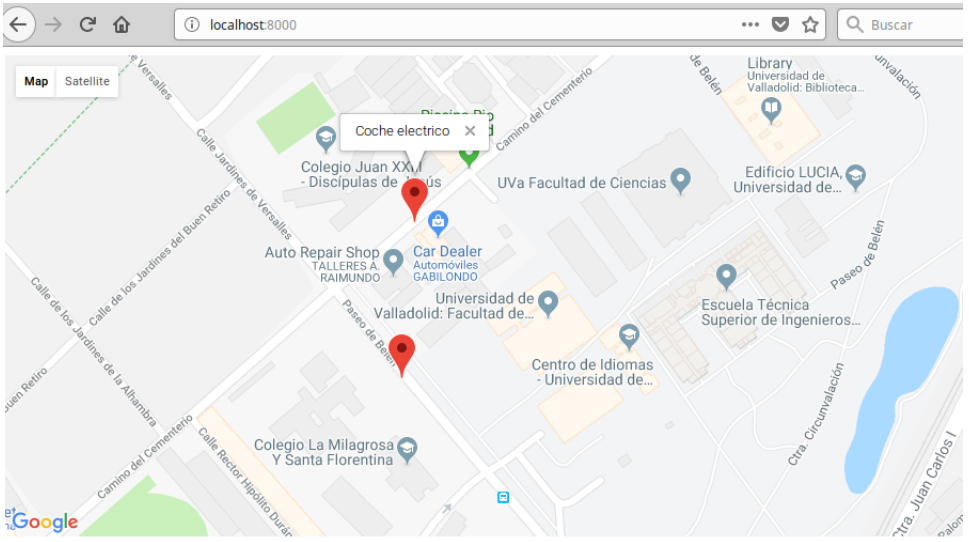

Figura 21: mapa con marcadores

A.3. Implementación de FIWARE-Orion

Dado que vamos a trabajar con FIWARE es un requisito imprescindible usar Orion *Context Broker*[12]. Debido a que no tenemos acceso a un CentOS ni un RedHat vamos a usar un **docker** para tener acceso a Orion[14], para ello instalamos docker-compose en nuestra máquina y siguiendo los pasos que se indican en la referencia creamos una carpeta a la que hemos llamado **FIWARE** y hacemos que por defecto el *Context Brocker* este escuchando en el puerto 1026.

Para activar Orion usamos la siguiente linea de código

```
sudo docker-compose up
```

Comprobamos que lo hemos hecho bien mediante la siguiente linea de código

```
curl localhost:1026/version
```

Tras esto podemos empezar a hacer las transferencias de datos entre nuestra aplicación en PHP y Orion mediante llamadas Rest API apoyándonos en JSONs.

A la hora de guardar los datos necesarios para el proyecto, y como se ha explicado en la parte de desarrollo del supuesto, se usaran 3 entidades para las bases de datos, estas son las entidades de de User y Trip y Vehicle.

Para poder comunicarnos desde el proyecto con Orion hemos necesitado ayuda de un nuevo repositorio de Laravel, llamado Guzzle[19]. Usando su cliente y haciendo una petición get a **localhost:1026/v2/entities/**, que es donde se encuentran los datos, podemos recibir la página web. Tras una pequeña transformación de esta como se muestra en la Figura 22, cogemos el **body** de este, el cual es en este punto un string con todos los datos, y lo transformamos en un JSON. Tras esto hacemos revisamos todos los datos que están guardados cogiendo solo los que pertenecen al tipo **Car**, y los devolvemos.

Mientras tanto, GMapsController le hemos modificado también para que acepte los datos como nos los devuelve **OrionController**. Hemos creado una **private function** en este controlador, **GMapsController**, que crea una nueva instancia de **OrionController**, puesto que va a ser algo que usemos mucho de diversas formas.

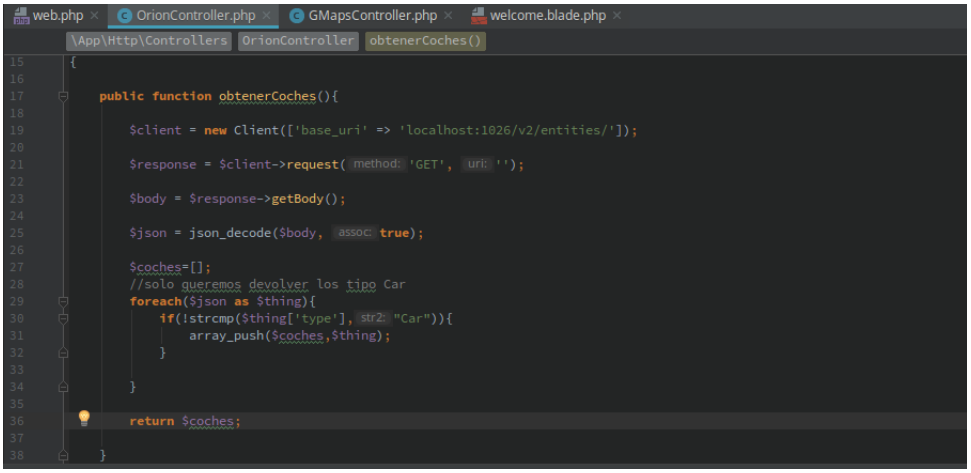

Figura 22: Orion Controller

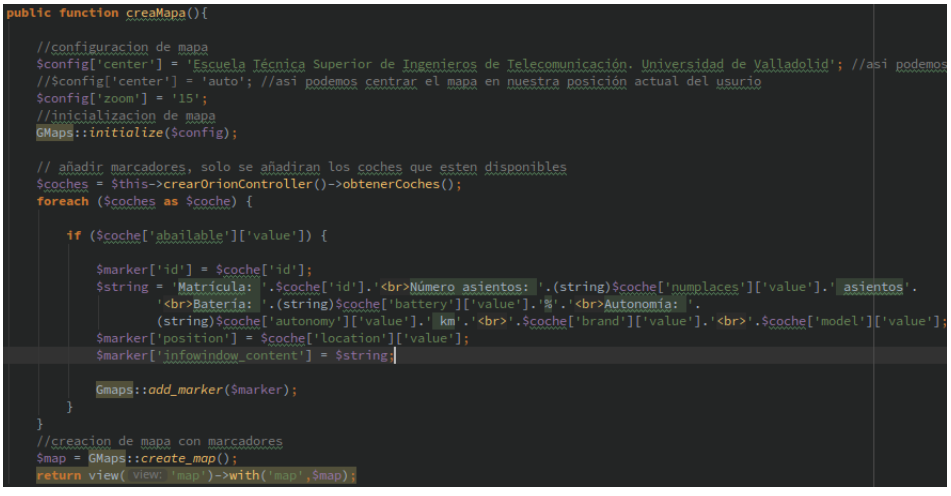

Figura 23: GMaps Controller

Como podemos ver en la Figura 23, ahora se crea realizamos la llamada **$this->crearOrionController()->obtenerCoches()** para obtener un JSON con todos los coches que están disponibles. Primero vamos sacando los coches uno por uno con el **foreach**, y vamos sacando los datos de estos para añadírselos a la etiqueta que tendrá el marcador, como el nivel de batería o el número de asiento, sacamos la localización del vehículo y creamos el marcador. Este proceso se repite para todos los coches que haya en la lista.

A.3.1. Registrar usuarios

Ahora crearemos un dos vistas, una para que los usuarios puedan registrarse y otra para que puedan identificarse. Dejaremos las ambas vistas simples puesto que lo que queremos es, en este caso, el paso de datos a través de Orion.

Para que el usuario se registre vamos a crear una vista en **resources/views** llamada **register.blade.php**.

```
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Identificarse</title>

    <!-- Fonts -->
    <link href="https://fonts.googleapis.com/css?family=Raleway:100,600" rel="stylesheet" type="text/css">
    <div>
        @if ($errors->any())
            <div class=alert-danger">
                <ul>
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </div>
        @endif
    </div>

</head>
```

Figura 24: register.blade.php

Como se puede apreciar en la Figura 24, el código que esta dentro del <**head**>, se encarga de indicar los errores aparecidos a la hora de introducir los datos, por ejemplo que falta el nombre de usuario, que el E-Mail tiene una estructura correcta o que la contraseña no coincide con su verificación. Esto último se consigue habiendo llamado al al segundo password **password_confirmation** en la vista y en el controlador haciendo un comprobación. Este formulario realiza una llamada a **verify**, que realmente no es más que llamar a la función **añadirUsuario(Request)** de **OrionController**.

Como se muestra en la Figura 25, se validan los datos del **$request**, que es una entidad de la clase **Request**. Esta clase nos permite recibir datos de los formularios de los **blades** y realizar una validación de estos mediante el comando **validate**. Entre otras cosas se pueden comprobar que sean tipo *string*, alfanuméricos, que tengan un tamaño en concreto que tengan sea requerido (**required**), o con la contraseña usando confirmed saber si la contraseña y la confirmación coinciden. De esta forma nos ahorramos el código a la hora de hacer las comprobaciones, y si algo esta mal, devolverá una lista con los errores que haya, la cual aparece en la cabecera de **register.blade.php** automáticamente.

De manera similar realizamos una comprobación para saber si e usuario ya existe. En este caso se decide hacer un arreglo, puesto que se van a tener pocos usuarios en la base de datos. Como se puede observar en las primeras lineas, lo que hacemos es crear una entidad del cliente usando la función **crearCliente()**, con la cual creamos una instancia para realizar una llamada a Orion

```php
/**
 * @param Request $request Realiza una petición y comprueba que los datos enviados sean correctos
 */
public function añadirUsuario(Request $request){

    $request->validate([
        "name" => 'string|required',
        "firstsurname" => 'string|required',
        "secondsurname" => 'string|required',
        "dni"=>'alpha_num|required|size:9',
        "mail"=>'email|required',
        "password"=>'required|confirmed',
        "bancaccount"=>'alpha_num|required',
        "phonenum"=>'integer|required',
    ]);
```

Figura 25: Request y Validate

CarRenting › app › Http › Controllers › OrionController.php

web.php | identify.blade.php | register.blade.php | welcome.blade.php | OrionController.php | GMapsController.php

\App\Http\Controllers › OrionController › añadirUsuario()

```php
            "bancaccount"=>'alpha_num|required',
            "phonenum"=>'integer|required',
        ]);

        $cadena = $request->input( key: 'dni');

        $client = $this->crearCliente();
        $response = $client->request( method: 'GET', uri: '?type=Person');
        $body = $response->getBody();
        $personas = json_decode($body, assoc: true);
        foreach ($personas as $persona){
            if($cadena == $persona['id']) {
                echo 'La persona ya está registrada';
                return view( view: 'register');
            }
        }
        $usuario['id'] = $cadena;
        $usuario['type'] = "Person";
        $usuario['name']['value'] = $request->input( key: 'name');
        $usuario['name']['type'] = "String";
        $usuario['firstsurname']['value'] = $request->input( key: 'firstsurname');
        $usuario['firstsurname']['type'] = "String";
        $usuario['secondsurname']['value'] = $request->input( key: 'secondsurname');
        $usuario['secondsurname']['type'] = "String";
        $usuario['mail']['value'] = $request->input( key: 'mail');
        $usuario['mail']['type'] = "String";
        $usuario['password']['value'] = $request->input( key: 'password');
        $usuario['password']['type'] = "String";
        $usuario['bancaccount']['value'] = $request->input( key: 'bancaccount');
        $usuario['bancaccount']['type'] = "String";
        $usuario['phonenum']['value'] = $request->input( key: 'phonenum');
        $usuario['phonenum']['type'] = "String";

        $conec = $client->request( method: 'POST', uri: '',['json'=>$usuario]);

        return $this->crearGMaps()->creaMapa();
    }
```

Figura 26: Comprobación y añadir entrada en Orion

mediante una uri base que en este caso es **localhost:1026/v2/entities/** que es donde se guardan todas las entidades de la base de datos. Esto se ha actualizado para la función **obtenerCoches()** de este controlador. Esto simplifica las llamadas, puesto que todas van a ser realizadas a esa misma uri básica con algunos añadidos detrás, como por ejemplo el que se realiza después, con el cual pedimos que nos devuelva todas las instancias de **Person** que tenemos en la base de datos. Tras esto comprobamos que no exista ningún usuario con esa id, si existe se realiza un echo del problema y la vista vuelve a **register.blade.php**, en caso contrario, se guardan todos los datos con la estructura que pide Orion en un JSON y utilizando el $client anterior realizamos esta vez una llamada tipo **POST** y le pasamos el JSON de los datos, tras esto se vuelve a crear el mapa para mostrarlo. Se ha hecho notar la falta de la afirmación de que el usuario ha sido registrado con éxito en la base de datos. Esto se planea arreglar mediante la adición de un mensaje emergente dando esta información si es posible, o cargando una página entre medias que lo muestre.

A.3.2. Identificar Usuario

Ahora que tenemos usuarios creados podemos centrarnos en permitirles que se identifiquen, para ello tenemos que tener una manera de guardar el estado en el que se encuentra la sesión. Afortunadamente, Laravel tiene diversas herramientas con las que podemos trabajar para lograrlo, una de ellas es la utilización de los **Helpers** de **Session** y **Authentication**.

Para empezar necesitamos verificar que el usuario existe, para eso necesitamos hacer una llamada **try/catch** con Client de Guzzel y recoger el error que nos genera como se muestra en la Figura 27.

```
/**
 * @param $mail
 * @return int el entero indica si la entrada existe (1) o no (0)
 */
private function existeUsuario($mail){

    $feedback = [];
    $client = $this->crearCliente();

    try {
        $client->get($mail);
    } catch (RequestException $e) {
        array_push($feedback,Psr7\str($e->getRequest()));
        if ($e->hasResponse()) {
            array_push($feedback,Psr7\str($e->getRequest()));;
        }
    }
    if(count($feedback) == 0){$existe = 1;}
    else{$existe= 0;}
    return $existe;
}
```

Figura 27: existeUsuario

En esta función se genera un *feedback* para capturar los errores. Se ha decidido no hacer nada con él porque solo nos es necesario para saber si se ha generado un error a la hora de buscar por el usuario, aunque este dato podría guardarse en un archivo .log en la carpeta de **storage** del proyecto. Esto no se hace porque se aumentaría de manera innecesaria el tamaño del proyecto, ya que se llama a esta función para comprobar que el usuario no existe a la hora de registrar un nuevo usuario y se espera que devuelva una excepción al hacerlo.

La vista encargada de identificar al usuario es muy similar a la utilizada en el registro, solo se han eliminado las entradas que no nos sirven, como las de nombre, apellidos y demás.

A.3.3. Sesiones

Una vez llegados a este punto, nos surge la duda de si Orion podrá servir para crear sesiones en Laravel. Es en este punto es donde se planean los tres posibles escenarios distintos que atacan la creación de las sesiones (Integración completa, media y mínima). Tras unas pruebas nos damos cuenta que las sesiones en Laravel son más sencillas de lo que en un principio aparentaban y que Orion es suficiente para poder manejarlas, puesto que los datos sobre las sesiones se van a almacenar de manera local en una de las carpetas del proyecto, exactamente en **storage/session**.

Para poder manejar las sesiones correctamente, se decide crear un nuevo controlador llamado SessionController. De esta misma forma, y aprovechando que tenemos acceso a las sesiones se amplia sobre la vista **map.blade.php** para trabajar con ellas. También se unen las vistas de **register.blade.php** y **identify.blade.php** en una sola manteniéndolas separadas mediante tablas en html.

Para crear una sesión no se necesita gran cosa, en este caso nuestras sesiones van a venir dadas por un par de claves, el E-Mail y el nombre y apellidos del usuario como se puede apreciar en la Figura 28.

```php
/**
 * @param $data Datos que serviran de key para poder identificar a los usuarios
 */
public function crearSesion($data){
    session()->put('email', $data[0]);
    session()->put('name', $data[1]);
    return;
}
```

Figura 28: Iniciar una sesión

De la misma manera vamos a querer poder cerrarla por lo que también se añade otra función como se aprecia en la Figura 29. En este caso solo se borran todas las claves.

```php
/**
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function cerrarSesion(){
    session()->flush();
    return redirect( to: '/');
}
```

Figura 29: Cerrar una sesión

En la configuración de Laravel en **config/session** la sesión se cerrar automáticamente tras 5 minutos de no haber hecho nada con ella.

En la vista **map.blade.php** hemos añadido más código para trabajar con las sesiones. Como se puede apreciar en la Figura 30, dependiendo de si el usuario tiene establecida o no una sesión, en la página aparecerán enlaces a Identificarse y Registrarse o el que le permitirá acceder a su perfil.

A.3.4. Reserva de Vehículos

Ahora crearemos las vistas para, primero, ver los detalles del vehículo que queremos reservar. Esta mostrara los datos importantes del vehículo. Ahora, y al ser solamente un prototipo, estos

Figura 30: map.blade.php

vehículos tienen pocos datos, así que lo que hará a mayores con respecto al mensaje del marcador es mostrar una foto del vehículo en cuestión.

Figura 31: detalles.blade.php

Como se muestra en la Figura 31, los datos que usa los recibe del Controlador **ReservaController**. A este controlador se accede a través de la ruta:

```
Route::get('/map/detalles/{id}', 'ReservaController@mirarCoche');
```

En esta llamada se pasa el id del vehículo en cuestión en la propia url y como se muestra en la Figura 32, este id se usa para localizar el vehículo en Orion y traer sus datos, antes de acceder a la vista, el identificador del vehículo se guarda en la sesión como una nueva clave de este la cual se borrará al crear el viaje.

Figura 32: ReservaController.php detallesVehiculo

En la vista de detalles del vehículo si tienes una sesión iniciada, aparecerá el botón para realizar la reserva, si no, un enlace para que el usuario pueda identificarse. Para acceder a esta vista se ha creado, modificado los marcadores, para que estos, al hacerles doble click, abran dicha vista (**detallesVehiculo.blade.php**) en una ventana emergente. Esto puede causar problemas dependiendo del navegador y de si el usuario permite abrir ventanas emergentes o no.

```
$marker[‘ondblclick’] =
    ‘window.open(\‘/map/detalles/’.$marker[‘id’].
    ‘\’,\‘_blank\’,\‘width=700,height=300,toolbar=no\’)’;
```

Con esto le pasamos la url concreta al marcador, ya que cuando se le haga doble click, llamará a la url que le indique su identificador, que es la matrícula del vehículo, pudiendo de esta manera pasar el dato que necesitamos para identificar el vehículo.

Si estamos identificados, en la página de los detalles del vehículo, podremos reservar el vehículo para usarlo. Este botón realiza una llamada a **reservaVehiculo** en **ReservaController** donde cambia el estado del vehículo a ocupado, se guarda la matrícula del vehículo en la sesión y en este caso nos devuelve la vista encargada de los datos del viaje a realizar. Esto en la realidad seria rellenado con los datos que tomemos del GPS del vehículo en cuestión.

Figura 33: usoVehiculo.blade.php

Como se observa en la Figura 33, solo se piden 3 datos, los kilómetros recorridos, el tiempo que se tarda en minutos y unas coordenadas que toma de un archivo llamado **coordenadas.txt**

localizado en **storage/coordenadas**. Le introduces la linea que quieras que lea y coloca al vehículo en dichas coordenadas. Esto lo hace a través de **CoordenadasController**, donde se lee el archivo y se le elimina el último caracter si no es el final del archivo, que es el de salto de linea.

La llamada de **usoVehiculo.blade.php**, es a **realizarTrip** en **OrionController**. Como se observa en la Figura 34, es aquí donde se realiza la llamada a **CoordenadasController**, tras esto, se van añadiendo los datos como las nuevas coordenadas del vehículo, se cambia el estado y la localización del vehículo y se crea el viaje. Después de añadir a la base de datos el viaje, se elimina la clave de matrícula de la sesión y se cierra la ventan emergente a través del script que se retorna.

```php
public function realizarTrip(Request $request){
    $request->validate([
        "kmMade" => 'required',
        "timeUsed" => 'required',
        "endPoint" => 'required',
    ]);
    $coorController = new CoordenadasController();
    $coordenadas = $coorController->obtenerCoordenadas($request->input( key: 'endPoint'));

    $Trip = [];
    $idVehiculo = session()->get( key: 'matricula');
    $uri = $idVehiculo . '?options=values&attrs=location';
    $client = $this->crearCliente();
    $body = $client->get($uri)->getBody();
    $location = json_decode($body, assoc: true);

    $this->cambiaEstadoCoche($idVehiculo);
    $lugar['location']['value'] = $coordenadas;
    $lugar['location']['type'] = "String";

    $baseTrip = session()->get( key: 'email') . '-Trip';
    $numero = $this->numeroTrip($baseTrip);
```

Figura 34: realizarTrip.php

A.3.5. Perfil del usuario

Esto se encuentra en la vista **perfil.blade.php**. Aquí se muestran los datos del usuario y se le permite cerrar sesión, darse de baja de la aplicación, cambiar ciertos datos como la contraseña o la cuenta bancaria y volver a la página del mapa. Cada una de estas tiene una llamada a una función especifica en **OrionController**, siendo las más complejas las que se usan para darse de baja de la aplicación y para cambiar el correo electrónico, donde tenemos que recorrernos toda la base de datos para borrarlos o cambiarlos. De manera similar, los datos que usa los toma de la sesión que está abierta y de las entradas que recibe.

A.3.6. Portada, términos, cabecera y estilos

Tras la creación de la página encargada de mostrar los datos del usuario, se decide crear una página de portada bastante simple, en este caso se decide, imitando la página original de **welcome.blade.php**, hacer una página sencilla donde se pueda identificar la tecnología empleada y el nombre de la aplicación. Otra cosa que se hizo fue extraer los estilos de esta y añadirlos a **style.blade.php**, archivo que hemos creado y que se va a encargar del estilo de toda la aplicación. También se crea el archivo **header.blade.php** que será el responsable de la cabecera en todas las páginas.

Para poder incluir tanto los estilos como la cabecera a las distintas vistas se utiliza el comando:

```
@include('template')
```

donde **template** indica el archivo .blade.php que vamos a incluir.

Para la cabecera, como se ha mostrado con anterioridad, tiene la condición de, si la sesión tiene una clave identificada como **name**, lo que correspondería con un usuario identificado, mostrar los enlaces a mi perfil y en nombre del usuario, en caso contrario, mostrar los enlaces para registrarse e identificarse. Como se muestra en la Figura 35, ahora en vez de estar en una tabla están divididas por las etiquetas ¡div¿. Cada una tiene una clase definida en el archivo **style.blade.php**.

```
<div class="top">
    <h1></h1>
<div class="top-left links"><a href="/">Página Principal</a></div>
<div class="top-right links">
    <a href="/map">Mapa</a>
    Bienvenido
    @if (!session()->has('name'))
        Usuario
        <a href="/map/identify">Identificarse</a>
        <a href="/map/register">Registrarse</a>
    @else
        {{session('name')}} <a href="/perfil">Mi Perfil</a>
    @endif
</div>
</div>
```

Figura 35: header.blade.php

A.3.7. Habilitar la aplicación en un *Host* compartido

Los técnicos de la escuela nos han dado acceso a una máquina virtual localizada en virtual.lab.inf.uva.es, con los siguientes puertos abiertos.

- 20051 Puesto para SSH
- 20052 Puerto para Web (http)

El usuario que tenemos para acceder es tfg.

Lo primero que hacemos es cambiar la contraseña de cambialaclaveya a robbahi. Tras esto subimos nuestra **aplicación** y las carpetas de **fiware**, donde se encuentra el archivo **docker-compose.yml**, y **script**, donde esta el script para plagar de vehículos el mapa. Para el caso del proyecto hacemos:

```
scp -P 20051 -r CarRenting/ tfg@virtual.lab.inf.uva.es:/home/tfg
```

De manera similar subimos las carpetas de **fiware** y la carpeta **scrip**.

Lo primero que hacemos es mover la carpeta del proyecto de **/home/tfg** a **/var/www**. Creamos una carpeta en **/var/www/html** llamada CarRenting, y todos los archivos que se encuentran en la carpeta de **/var/www/CarRenting/public** les copiamos en la que acabamos de crear, incluido el archivo **.htaccess**. A continuación tenemos que modificar el archivo **index.php** de la carpeta **CarRenting** localizada en html y modificar las siguientes lineas:

```
require __DIR__.'/../vendor/autoload.php';
\$app = require_once __DIR__.'/../bootstrap/app.php';
```

por

```
require __DIR__.'/../../CarRenting/vendor/autoload.php';
\$app = require_once __DIR__.'/../../CarRenting/bootstrap/app.php';
```

Tras esto tenemos que habilitar el uso de .htaccess en apache. Para esto hacemos lo siguiente:

```
sudo a2enmod rewrite
```

abrimos el archivo **apache2.conf**.

```
sudo vi /etc/apache2/apache2.conf
```

en el archivo buscamos las siguientes lineas.

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

y sustituimos **None** por **All**.

Ahora solo tenemos que modificar todos los enlaces de las vistas localizados en **/var/www/CarRenting/r** añadiendo delante de cada llamada **/CarRenting**. Por ejemplo, la llamada al mapa que se localiza en el archivo **header.blade.php** ya no seria :

```
<a href="/map">Mapa</a>
```

si no

```
<a href="/CarRenting/map">Mapa</a>
```

Una vez hecho esto y comprobado que los enlaces funcionan, se activa Orion desde **/home/tfg/fiware** usando el comando

```
docker-compose up
```

Tras esto se cargan todos los datos en Orion usando el script que también nos hemos traído y recargamos el mapa para comprobar que ha cargado correctamente los vehículos. Ahora podemos terminar de comprobar todos los enlaces creándonos un usuario nuevo, reservando un coche y demás.

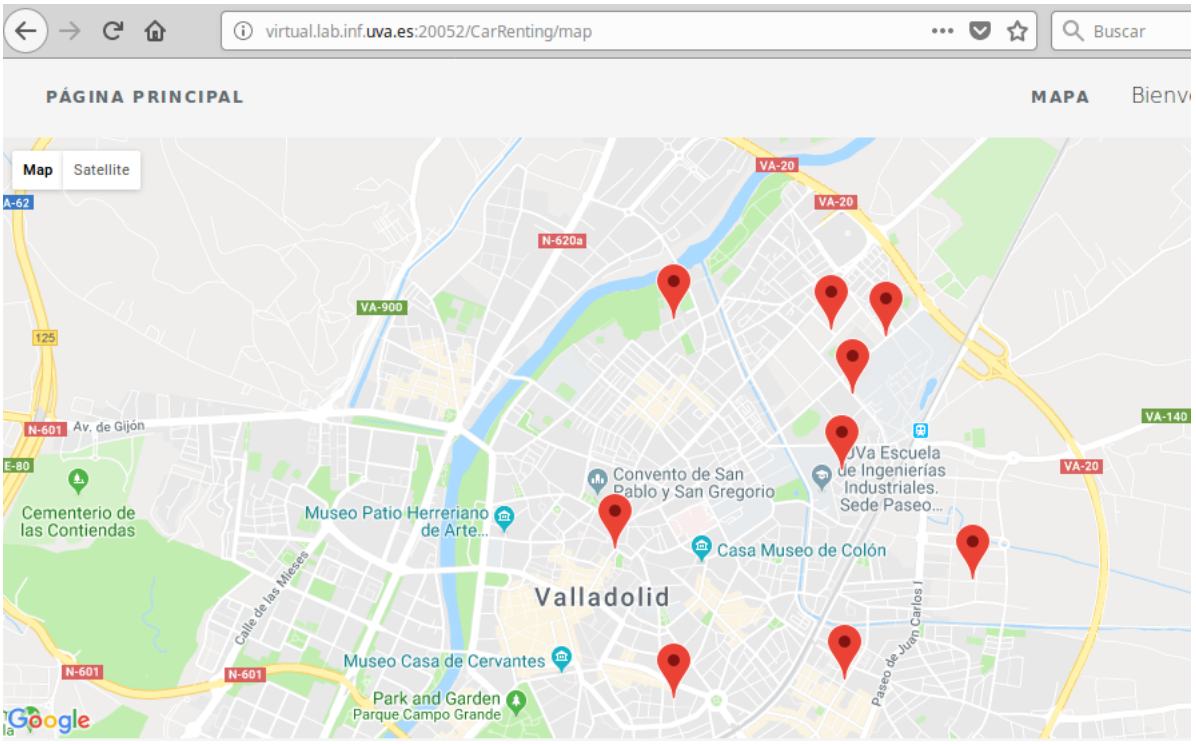

Figura 36: Vista del mapa en virtual.lab.inf.uva.es con todos los marcadores

B. Acceder a una página de manera incorrecta

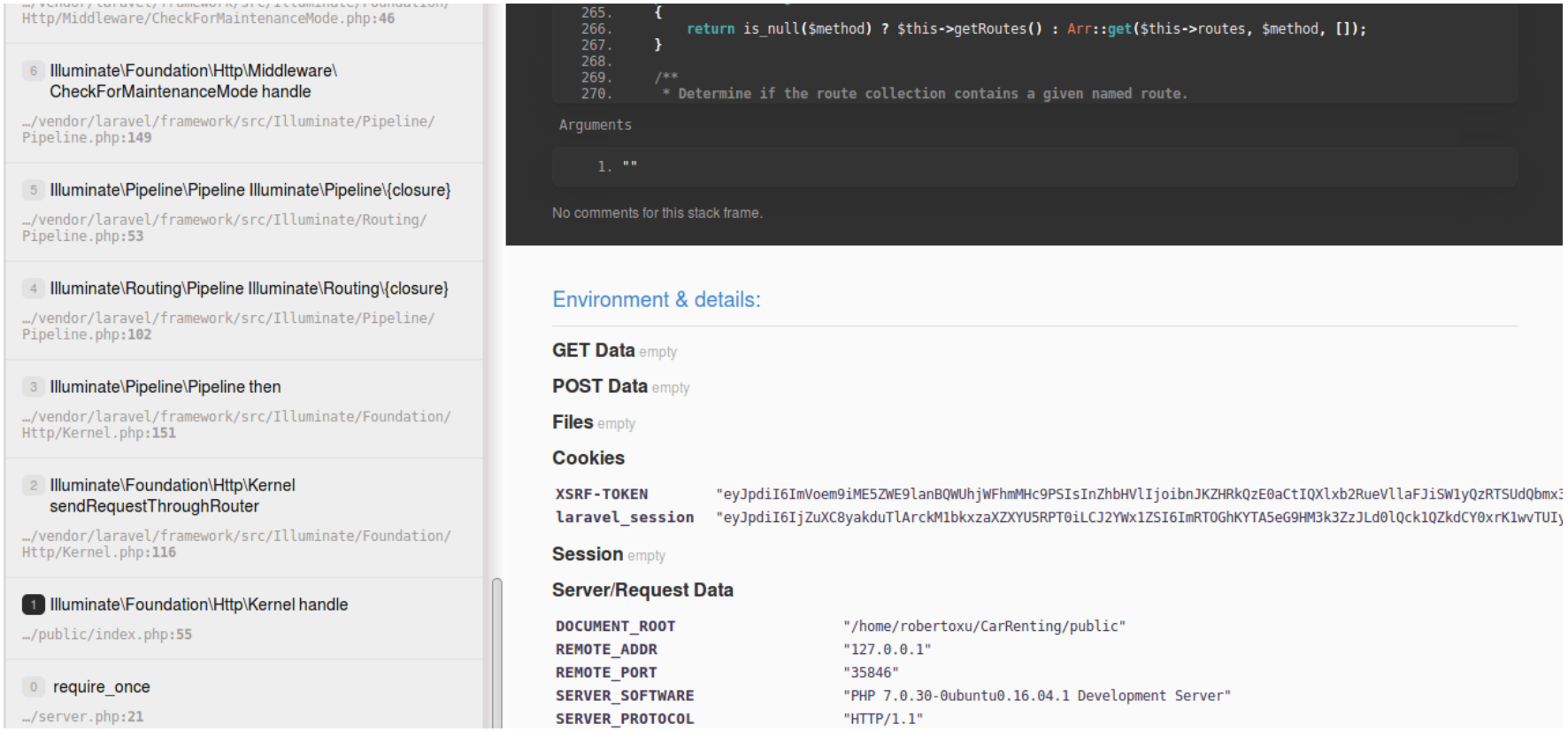

Figura 37: Vista de acceso a una página incorrecta

Se muestran datos importantes como el XSRF-TOKEN y datos importantes de la máquina que no tendrían que revelarse

Contenido del CD adjunto

- memoria.pdf
- CarRenting/ (Código de la aplicación)
  - app/
    - Console/
    - Exceptions/
    - Extension/
    - Http/
      - Controllers (Aquí están los archivos *Controller.php)
      - Middleware
      - kernel.php
    - Model/
    - Providers/
    - User.php
  - artisan
  - bootstrap/
  - composer.json
  - composer.lock
  - config (Archivos de configuración)
  - database/
  - package.json
  - phpinput.xml
  - public/
  - readme.md
  - resources/
    - assets/
    - lang/
    - views/ (Archivos .blade.php que usamos en las vistas)
  - server.php
  - storage/ (Carpeta que se usa para guardar datos propios del servidor)
  - test/
  - vendor/
  - webpack.mix.js
- fiware/
  - docker-composer.yml

- scripts/
    - orion.sh