Universidad de Valladolid

ISmartParking: Privacidad en aplicaciones de parking inclusivo desarrolladas sobre la plataforma Fiware

Curso 2019-2020

Escuela de Ingeniería Informática

Grado en Ingeniería Informática, Mención en Tecnologías De La Información

Autor:
Adrián Bailador Panero
Tutores:
Mercedes Martínez González
Pablo de la Fuente Redondo

Agradecimientos

A mis tutores, Mercedes y Pablo, por su confianza a la hora de proponer este proyecto, su paciencia y ayuda para poder llevarlo a cabo.
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 mis padres, Jose Luis y Rosa, 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 mi hermano, Héctor, que me ha apoyado y tocado las narices al mismo tiempo.
A mi novia, Raquel, que ha tenido que aguantar mis cabreos y mis días no tan buenos, pero siempre ha estado ahí para ayudarme.
A mis abuelos, Tere y Hortensio, que este donde este, él me sigue ayudando. Te echo de menos.

Resumen

Uno de los problemas habituales de la vida en las ciudades es el de encontrar, especialmente a determinadas horas, un lugar disponible para aparcar cuando nos desplazamos a un lugar dentro de la misma. Aunque en muchas ciudades existen lugares para aparcar tanto externos (*onstreet*) como en aparcamientos ubicados en locales, subterráneos, etc. (*offstreet*), el problema es localizar un lugar dónde aparcar el vehículo con la restricción añadida, en la mayoría de los casos, de la proximidad al lugar de destino.

Existen estudios que analizan el tiempo medio de espera en la localización de dicho lugar donde aparcar y la incidencia que tiene dicho tiempo en el conductor. Por ello, uno de los campos de interés de las aplicaciones para *Smartcities* es el desarrollo de sistemas de localización de espacios vacíos en los lugares de aparcamientos, que no garantiza la disponibilidad de los mismos en el momento de acceso, y, en algunos casos, sistemas de reserva previa de sitio para minimizar el tiempo de acceso al lugar de aparcamiento deseado.

El objetivo de este proyecto es realizar un prototipo de gestión de sitios de aparcamiento, con algunas de las características poco comunes, utilizando la plataforma Fiware, que intenta satisfacer algunos aspectos de los requeridos en el Reglamento General de Protección de Datos (RGPD).

Índice

Índice de figuras

Índice de tablas

1. Introducción

Smartcity es un término ligado a la utilización de las Tecnologías de la Información y las Comunicaciones (TIC, ICT en inglés) para resolver problemas de las ciudades, como economía local, transporte o calidad de vida. Dichas tecnologías permiten a las administraciones públicas o empresas desarrollar aplicaciones que faciliten las operaciones a realizar en dichas áreas.

Una de las áreas de especial interés en todas las ciudades es el del transporte, a veces se habla genéricamente de movilidad. Bajo dicho concepto se engloban muchos retos como gestión del transporte público, gestión de las tasas o gestión de los aparcamientos, ya sean públicos o privados. Los servicios relacionados con la gestión de aparcamientos proporcionan nuevas formas de optimizar el uso de los espacios de aparcamiento lo que facilita al usuario, entre otras cosas, la localización de un espacio adecuado y próximo de acuerdo con sus características.

Es evidente que las aplicaciones de este tipo utilizan, en algún momento, datos privados de los usuarios. Desde hace tiempo existe un fuerte interés por parte de los usuarios de las Tecnologías de la Información sobre el uso de sus datos personales. La entrada en vigor del Reglamento General de Protección de Datos (RGPD) en mayo de 2018 plantea retos a resolver en el desarrollo de aplicaciones para *Smartcities*. Si se consideran aspectos relacionados con la privacidad (datos personales) en dicho contexto, hay que tener en cuenta que los ataques a la privacidad pueden venir de la infraestructura (gestión de la información) o ataques externos. En este trabajo nos vamos a centrar en el caso de un sistema de gestión de sitios de aparcamiento que facilitará al usuario la posibilidad de seleccionar y reservar un sitio para aparcar su vehículo en un aparcamiento, teniendo en cuenta las características de su vehículo y las personales del conductor, como, por ejemplo, personas con vehículos adaptados por su discapacidad, mujeres embarazadas con problemas de movilidad y necesidades de espacios específicos, etc.

Como se ha indicado, la aplicación de la regulación RGPD a partir de mayo de 2018 en lo que respecta al tratamiento de datos personales implica consideraciones a tener en cuenta en todo el proceso de desarrollo de aplicaciones que traten en algún momento del mismo con datos personales. De hecho, en el documento de RGPD se habla de *"privacidad desde el diseño y por defecto"*, lo que da una idea de la importancia que debe tener este aspecto en el desarrollo de aplicaciones.

El objetivo de este TFG es realizar una prueba de concepto sobre la incidencia que el cumplimiento de lo especificado en el Reglamento General de Protección de Datos (en adelante RGPD) tiene en el desarrollo de aplicaciones para cuestiones relacionadas con las *Smartcities*. En concreto, se ha estudiado un prototipo de gestión de sitios de aparcamiento, con algunas de las características antes citadas, utilizando la plataforma FIWARE que intenta satisfacer aspectos de los requeridos en el RGPD.

El reglamento de la U.E. (2016/679) relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de las mismos, proporciona una serie de definiciones que nos facilitan el uso adecuado de los términos sin confusión. En lo que respecta al trabajo presentado, intentaremos ajustarnos al uso de los conceptos tal y como están definidos en dicho reglamento. Entre dichas definiciones, transcribimos algunas que nos parecen más relevantes para el trabajo:

a) *datos personales:* toda información sobre una persona física identificada o identificable («el interesado»); se considerará persona física identificable toda persona cuya identidad pueda determinarse, directa o indirectamente, en particular mediante un identificador, como por ejemplo un nombre, un número de identificación, datos de localización, un identificador en línea o uno o varios elementos propios de la identidad física, fisiológica, genética, psíquica, económica, cultural o social de dicha persona.

b) *tratamiento:* cualquier operación o conjunto de operaciones realizadas sobre datos personales o conjuntos de datos personales, ya sea por procedimientos automatizados o no, como la recogida, registro, organización, estructuración, conservación, adaptación o modificación, extracción, consulta, utilización, comunicación por transmisión, difusión o cualquier otra forma de habilitación de acceso, cotejo o interconexión, limitación, supresión o destrucción.

c) *tercero:* persona física o jurídica, autoridad pública, servicio u organismo distinto del interesado, del responsable del tratamiento, del encargado del tratamiento y de las personas autorizadas para tratar los datos personales bajo la autoridad directa del responsable o del encargado.

d) *elaboración de perfiles:* toda forma de tratamiento automatizado de datos personales consistente en utilizar datos personales para evaluar determinados aspectos personales de una persona física, en particular para analizar o predecir aspectos relativos al rendimiento profesional, situación económica, salud, preferencias personales, intereses, fiabilidad, comportamiento, ubicación o movimientos de dicha persona física.

1.1. Objetivos

A partir del objetivo general descrito previamente se han planteado los siguientes objetivos concretos:

- Diseñar y desarrollar una aplicación en la que el usuario pueda tanto reservar plazas en un parking, como modificar o cancelar dicha reserva.
- Proteger los datos personales del usuario de la aplicación acorde con el RGPD.
- Utilizar tecnologías que agilicen el desarrollo de un prototipo y den como resultado una aplicación escalable y mantenible fácilmente.
- Seleccionar aquellos elementos de FIWARE que pueden ser útiles para nuestro supuesto y utilizarlos en el desarrollo.
- Aplicar los conocimientos adquiridos a lo largo de la carrera.

1.2. Estructura del documento

A continuación, se proporciona un pequeño esquema de los temas principales que van a ser tratados en los siguientes capítulos, así como una breve descripción de cada uno de ellos.

- Introducción: Breve introducción del objetivo del proyecto.
- Contexto: Caso práctico al problema.
- Planificación: Desglose del tiempo y esfuerzo que se estiman necesarios para llevar a cabo cada una de las fases de las que se compone el proyecto. Incluye la gestión de riesgos.

- FIWARE: Describe la estructura de FIWARE como un marco de trabajo a utilizar para desarrollar aplicaciones para Smartcities, explicando con más detalle las posibilidades del Generic Enabler (GE) Orion que se ha utilizado en este TFG.
- Modelos de Privacidad: Describe algunos modelos de privacidad, así como algunas propuestas de arquitecturas para la realización de aplicaciones con tratamiento de aspectos de privacidad.
- Análisis: Detalle de los aspectos más relevantes de la arquitectura y diseño.
- Arquitectura y diseño: Detalle de los aspectos más relevantes de la arquitectura y diseño.
- Desarrollo del prototipo: Detalle de los aspectos correspondientes a la fase de la construcción.
- Herramientas y Tecnologías utilizadas: Se explican las tecnologías utilizadas en la solución propuesta.
- Conclusiones y líneas futuras: Se comentan las conclusiones obtenidas a lo largo del desarrollo del proyecto y las posibles líneas futuras de trabajo.
- Referencias: Fuentes consultadas a lo largo de la realización de proyecto.

2. Contexto

2.1. Caso práctico al problema

En nuestro supuesto práctico hemos considerado un ejemplo de sistema de reserva previa con algunas características menos habituales, como el tamaño de la plaza o las necesidades específicas para las plazas en el caso de algunos usuarios (problemas de movilidad, embarazadas, sillas de bebés, etc.). El proceso de registro y reserva, así como el de pago de la ocupación del lugar reservado, precisa del manejo de datos personales y, por tanto, sometidos a regulaciones concretas como las establecidas en el RGPD, lo que implica que el desarrollo de cualquier aplicación debe tener en cuenta no sólo el manejo de dichos datos personales sino las implicaciones que las pautas de privacidad tiene sobre el procesamiento de los mismos.

En este apartado describiremos al grupo de personas a las que estará destinada la aplicación a desarrollar en este TFG. Para ello, tendremos que elaborar una descripción precisa de nuestro usuario. La idea que se persigue es que hay que diseñar la solución para todos los usuarios. Es importante destacar que habrá dos tipos de roles:

- Usuarios: Usuarios finales que interactuaran con el sistema. Éste será el personaje primario.
- Administrador: Persona encargada de llevar un control de la aplicación.

3. Planificación

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

3.1. Planificación y seguimiento del trabajo

Para el desarrollo de este trabajo, que se empezó en el mes de enero del año 2019, 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 el mes de marzo de 2020 debido principalmente a que trabajo en una empresa y 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 probador.
- **Otros recursos:** 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 Ingeniería Informática que se utilizará para poner en producción el prototipo.

La documentación de las tareas de desarrollo se va compaginando con la documentación de la memoria, realizando las tareas de desarrollo por las mañanas y las de documentacion por las tardes y a la vez compaginandolo con el trabajo, intentado ir añadiendo las novedades regularmente.

Se decide realizar un método iterativo e incremental más simple que SCRUM. 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	Baja

Tabla 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 tabla [1], se hacen en intervalos de tiempo diferente dentro del mismo día.

3.2. Iteraciones

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

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

1. **Iteración 01:** Ámbito y límites del proyecto.
    - **Duración:** 11/Enero/2019 - 29/Enero/2019.
    - **Descripción:** Definición del propósito del proyecto con las limitaciones, restricciones y suposiciones adoptadas.
2. **Iteración 02:** RGPD, Búsqueda de información y comprensión.
    - **Duración:** 30/Enero/2019 - 19/Febrero/2019.
    - **Descripción:** Búsqueda de información acerca del RGPD y comprensión de los conocimientos adquiridos.
3. **Iteración 03:** FIWARE.
    - **Duración:** 30/Enero/2019 - 19/Febrero/2019.
    - **Descripción:** El objetivo es buscar información acerca de FIWARE y comprensión de los conocimientos adquiridos.
4. **Iteración 04:** Desarrollo de la parte del RGPD y FIWARE en el documento.
    - **Duración:** 22/Febrero/2019 - 11/Marzo/2019.
    - **Descripción:** Desarrollo en el documento a cerca de las partes relacionadas con el RGPD y FIWARE.
5. **Iteración 05:** Revisión de la documentación, partes del RGPD y FIWARE.
    - **Duración:** 14/Marzo/2019 - 10/Enero/2020.
    - **Descripción:** Se realiza una revisión y actualización sobre las partes del RGPD y FIWARE en el documento.
6. **Iteración 06:** Especificación de requisitos.
    - **Duración:** 2/Abril/2019 - 19/Abril/2019.
    - **Descripción:** Definición de los objetivos principales del proyecto tras haber entendido la necesidad del problema.
7. **Iteración 07:** Casos de uso principales.
    - **Duración:** 21/Abril/2019 - 10/Mayo/2019.
    - **Descripción:** Enunciado de las funcionalidades que se obtienen a partir de los requisitos especificados.

8. **Iteración 08:** Estimación de los riesgos.
    - **Duración:** 14/Mayo/2019 - 31/Mayo/2019.
    - **Descripción:** Identificación de los riesgos que pueden surgir a lo largo de la vida del proyecto.
9. **Iteración 09:** Realización plan de fases.
    - **Duración:** 4/Junio/2019 - 21/Junio/2019.
    - **Descripción:** Estructuración de las distintas fases que sigue el método de trabajo seleccionado para el desarrollo del proyecto.
10. **Iteración 10:** Redacción plan de desarrollo.
    - **Duración:** 25/Junio/2019 - 12/Julio/2019.
    - **Descripción:** Redacción del plan de desarrollo.
11. **Iteración 11:** Elicitación de requisitos.
    - **Duración:** 16/Julio/2019 - 2/Agosto/2019.
    - **Descripción:** Acuerdo de los requisitos funcionales como no funcionales anteriormente mencionados con el cliente.
12. **Iteración 12:** Especificación de requisitos.
    - **Duración:** 6/Agosto/2019 - 23/Agosto/2019.
    - **Descripción:** Exposición de los requisitos ya acordados por el cliente como por el responsable del proyecto.
13. **Iteración 13:** Diagrama Casos de Uso.
    - **Duración:** 27/Agosto/2019 - 16/Septiembre/2019.
    - **Descripción:**Diagrama UML con los requisitos obtenidos anteriormente y estableciendo los actores que realizan cada caso de uso.
14. **Iteración 14:** Realización Casos de uso.
    - **Duración:** 18/Septiembre/2019 - 4/Octubre/2019.
    - **Descripción:** Exposición detallada de cada caso de uso mencionado el nombre, descripción, actor, precondiciones, secuencia, excepciones y postcondiciones de cada caso de uso.
15. **Iteración 15:** Modelado de dominio.
    - **Duración:** 7/Octubre/2019 - 5/Noviembre/2019.
    - **Descripción:** Representación del vocabulario y conceptos clave del dominio del problema en UML.

16. **Iteración 16:** Elaboración de diagramas de secuencia, E-R.

    - **Duración:** 6/Noviembre/2019 - 20/Noviembre/2019.
    - **Descripción:** Elaboración de diagramas de secuencia, E-R: Elaboración de cada uno de los distintos tipos de diagramas UML que representan la lógica y modelo de datos del negocio del proyecto.

17. **Iteración 17:** Construcción Base de datos.

    - **Duración:** 21/Noviembre/2019 - 28/Noviembre/2019.
    - **Descripción:** de la base de datos a partir del modelo Entidad – Relación obtenido anteriormente.

18. **Iteración 18:** Diseño Plataforma web.

    - **Duración:** 29/Noviembre/2019 - 6/Diciembre/2019.
    - **Descripción:** construcción de plataforma web que permitirá ser la interfaz entre el administrador y los datos de la aplicación del proyecto.

19. **Iteración 19:** Construcción Funcionalidad.

    - **Duración:** 9/Diciembre/2019 - 30/Diciembre/2019.
    - **Descripción:** Construcción de aplicación Web que guiará a los usuarios a reservar una plaza en el parking de su ciudad.

20. **Iteración 20:** Revisión arquitectura.

    - **Duración:** 13/Enero/2020 - 20/Enero/2020.
    - **Descripción:** revisión de la arquitectura software del proyecto para comprobar que sigue el diseño anteriormente elaborado.

21. **Iteración 21:** Documentación funcionalidad.

    - **Duración:** 21/Enero/2020 - 23/Enero/2020.
    - **Descripción:** Creación de manual de funcionamiento de la aplicación para su posterior uso.

22. **Iteración 22:** Realización de pruebas.

    - **Duración:** 24/Enero/2020 - 27/Enero/2020.
    - **Descripción:** procesos que permiten verificar y revelar la calidad de la aplicación centrándose en fallos de implementación, calidad (estabilidad, escalabilidad, eficiencia y seguridad) o usabilidad.

23. **Iteración 23:** Realización de manuales de usuario.

    - **Duración:** 28/Enero/2020 - 3/Febrero/2020.
    - **Descripción:** realización de los manuales de usuario y administrador de la aplicación.

24. **Iteración 24:** Realización del manual de instalación.

    - **Duración:** 5/Febrero/2020 - 10/Febrero/2020.
    - **Descripción:** realización de los manuales de instalación.

En el diagrama de gantt siguiente tenemos todas las actividades que se han realizado.

Nombre	Inicio	Finalización	Duración
Ámbito y límites del proyecto	11/1/2019	29/1/2019	13 días
RGPD, Búsqueda de información y comprensión.	30/1/2019	19/2/2019	15 días
FIWARE	30/1/2019	19/2/2019	15 días
Desarrollo de la parte del RGPD y FIWARE en el documento	22/2/2019	11/3/2019	12 días
Revisión de la documentación, partes del RGPD y FIWARE	14/3/2019	10/1/2020	217 días
Especificación de requisitos	2/4/2019	19/4/2019	14 días
Casos de uso principales	22/4/2019	10/5/2019	15 días
Estimación de los riesgos	14/5/2019	31/5/2019	14 días
Realización plan de fases	4/6/2019	21/6/2019	14 días
Redacción plan de desarrollo	25/6/2019	12/7/2019	14 días
Elicitación de requisitos	16/7/2019	2/8/2019	14 días
Especificación de requisitos	6/8/2019	23/8/2019	14 días
Diagrama Casos de Uso	27/8/2019	16/9/2019	15 días
Realización Casos de uso	17/9/2019	10/10/2019	18 días
Modelado de dominio	11/10/2019	11/11/2019	22 días
Elaboración de diagramas de secuencia, E-R	12/11/2019	26/11/2019	11 días
Construcción Base de datos	27/11/2019	11/12/2019	11 días
Diseño Plataforma web	12/12/2019	19/12/2019	6 días
Construcción Funcionalidad	20/12/2019	20/1/2020	22 días
Revisión arquitectura	21/1/2020	28/1/2020	6 días
Documentación funcionalidad	29/1/2020	31/1/2020	3 días
Realización de pruebas	3/2/2020	4/2/2020	2 días
Realización de manuales de usuario	5/2/2020	11/2/2020	5 días
Realización del manual de instalación	12/2/2020	17/2/2020	4 días

Figura 1: Diagrama de Gantt Inicial.

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

El tiempo ha sido considerablemente mayor de lo planificado. Entre las causas que se deben a la llegada de este punto han sido el cumplimiento de los riesgos R02 (ausencia de familiaridad de las herramientas y tecnologías a usar en el desarrollo del proyecto), R03 (incumplimiento del calendario de planificación), R04 (baja del responsable del proyecto por causas de salud, familiares, etc.), y R08 (problemas asociados al carácter distribuido del calendario). Estos riesgos, junto con la situación laboral del responsable del proyecto han determinado que la duración de este sea mayor de lo planificado. También el cumplimiento del riesgo R07 (Problemas en el trabajo) el ataque de un virus, hizo que perdiera más de un mes de trabajo en el proyecto.
Los riesgos mencionados anteriormente vienen descritos en el punto 3.3.

A continuación, se muestra el diagrama de gantt final.

Nombre	Inicio	Finalización	Duración
Ámbito y límites del proyecto	11/1/2019	29/1/2019	13 días
RGPD, Búsqueda de información y comprensión.	30/1/2019	1/3/2019	23 días
FIWARE	30/1/2019	1/3/2019	23 días
Desarrollo de la parte del RGPD y FIWARE en el documento	4/3/2019	29/3/2019	20 días
Revisión de la documentación, partes del RGPD y FIWARE	1/4/2019	28/2/2020	240 días
Especificación de requisitos	2/4/2019	19/4/2019	14 días
Casos de uso principales	22/4/2019	10/5/2019	15 días
Estimación de los riesgos	14/5/2019	31/5/2019	14 días
Realización plan de fases	4/6/2019	21/6/2019	14 días
Redacción plan de desarrollo	25/6/2019	12/7/2019	14 días
Elicitación de requisitos	16/7/2019	2/8/2019	14 días
Especificación de requisitos	6/8/2019	23/8/2019	14 días
Diagrama Casos de Uso	26/8/2019	18/9/2019	18 días
Realización Casos de uso	19/9/2019	22/10/2019	24 días
Modelado de dominio	23/10/2019	2/12/2019	29 días
Elaboración de diagramas de secuencia, E-R	3/12/2019	17/12/2019	11 días
Construcción Base de datos	18/12/2019	1/1/2020	11 días
Diseño Plataforma web	2/1/2020	9/1/2020	6 días
Construcción Funcionalidad	10/1/2020	10/2/2020	22 días
Revisión arquitectura	11/2/2020	18/2/2020	6 días
Documentación funcionalidad	19/2/2020	21/2/2020	3 días
Realización de pruebas	24/2/2020	25/2/2020	2 días
Realización de manuales de usuario	26/2/2020	3/3/2020	5 días
Realización del manual de instalación	4/3/2020	9/3/2020	4 días

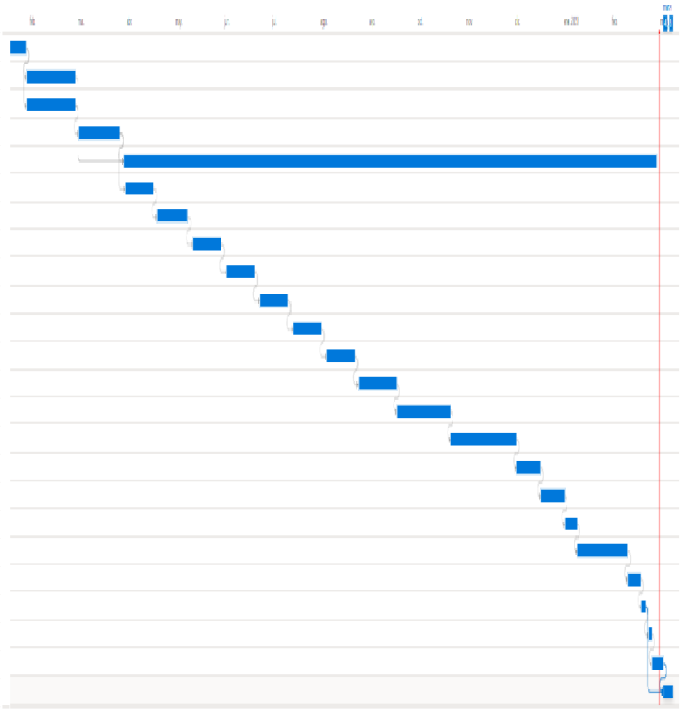

Figura 2: Diagrama de Gantt Final.

3.3. 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 [14], 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

Tabla 2: Identificación valor de riesgo frente a Probabilidad/Impacto.

3.3.1. Riesgos

1. **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.
2. **Riesgo 02:** La falta de experiencia del desarrollador con FIWARE.
    - **Impacto:** Marginal
    - **Probabilidad:** Alta
    - **Plan de Protección:** Dedicar una gran cantidad de tiempo en el aprendizaje de forma autónoma de dichas tecnologías.
    - **Plan de Contingencia:** Volver a aprender a navegar la página de FIWARE.
3. **Riesgo 03:** Los posibles errores de organización del tiempo disponible por parte del desarrollador.
    - **Impacto:** Marginal
    - **Probabilidad:** Alta
    - **Plan de Protección:** Elaborar una buena planificación que sea realista y detallada, y dejando un margen de flexibilidad temporal para posibles imprevistos.
    - **Plan de Contingencia:** Tener un margen de flexibilidad temporal bastante amplio.
4. **Riesgo 04:** Enfermedad del trabajador.
    - **Impacto:** Crítico
    - **Probabilidad:** Baja
    - **Plan de Protección:** Al consistir el personal de una única persona, está deberá evitar situaciones de riesgo.
    - **Plan de Contingencia:** Compensar las horas perdidas una vez recuperado.
5. **Riesgo 05:** Fallos ortográficos en la memoria.
    - **Impacto:** Medio
    - **Probabilidad:** Medio
    - **Plan de Protección:** Revisiones del documento con un corrector ortográfico.
    - **Plan de Contingencia:** Revisión de la memoria por una tercera persona.

6. **Riesgo 06:** Un mal diseño de la aplicación a construir.
    - **Impacto:** Alto
    - **Probabilidad:** Medio
    - **Plan de Protección:** Puede dificultar el alcanzar el objetivo principal del proyecto.
    - **Plan de Contingencia:** Realizar reuniones frecuentes con los directores del proyecto para comentar las decisiones de diseño que se está tomando.
7. **Riesgo 07:** Problemas en el trabajo.
    - **Impacto:** Alto
    - **Probabilidad:** Medio
    - **Plan de Protección:** Diferentes problemas que surgen diariamente en el entorno laboral que puueden dificultar el cumplimiento de objetivos del proyecto.
    - **Plan de Contingencia:** Llevar un seguimiento exhausistivo de los plazos.
8. **Riesgo 08:** Problemas asociados a los periodos vacacionales en el trabajo.
    - **Impacto:** Medio
    - **Probabilidad:** Medio
    - **Plan de Protección:** Los periodos vacacionales puedan afectar al desarrollo del proyecto.
    - **Plan de Contingencia:** Actualizar la planificación del proyecto (carga de trabajo).

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 *Smartcities*.

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 *Smartcities*, *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 de 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 contabilidad.
- **Interface to Networks and Devices (I2ND) Architecture:** Los GEs ubicados en esta categoría proporciona 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 HTML5 y un acercamiento a interfaces de usuario basadas en web. Para esto estos GEs añaden nuevas capacidades de interacción como por ejemplos 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 *Smartcities* de última generación.

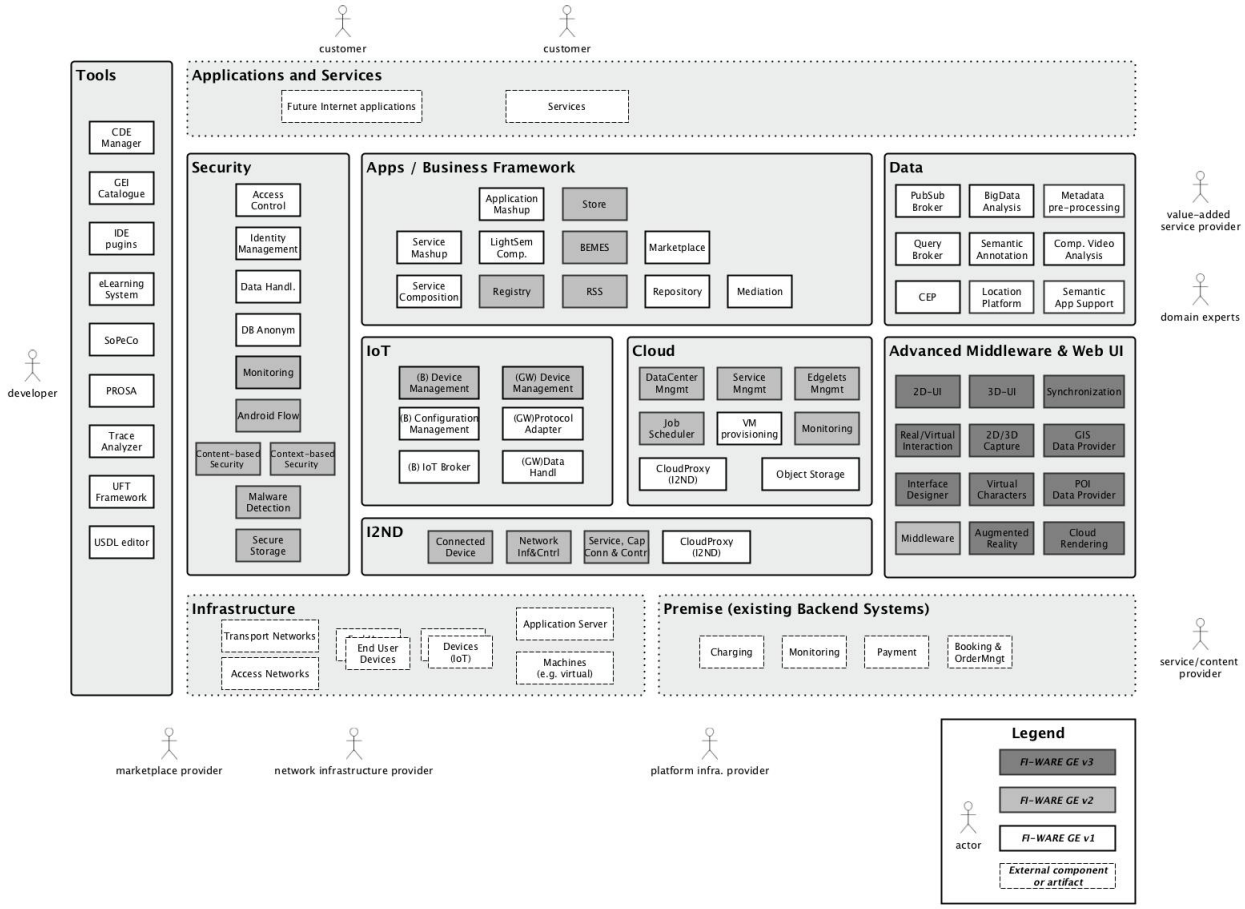

Figura 3: Arquitectura general de FIWARE.[15]

Como podemos ver en la Figura [3] 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.

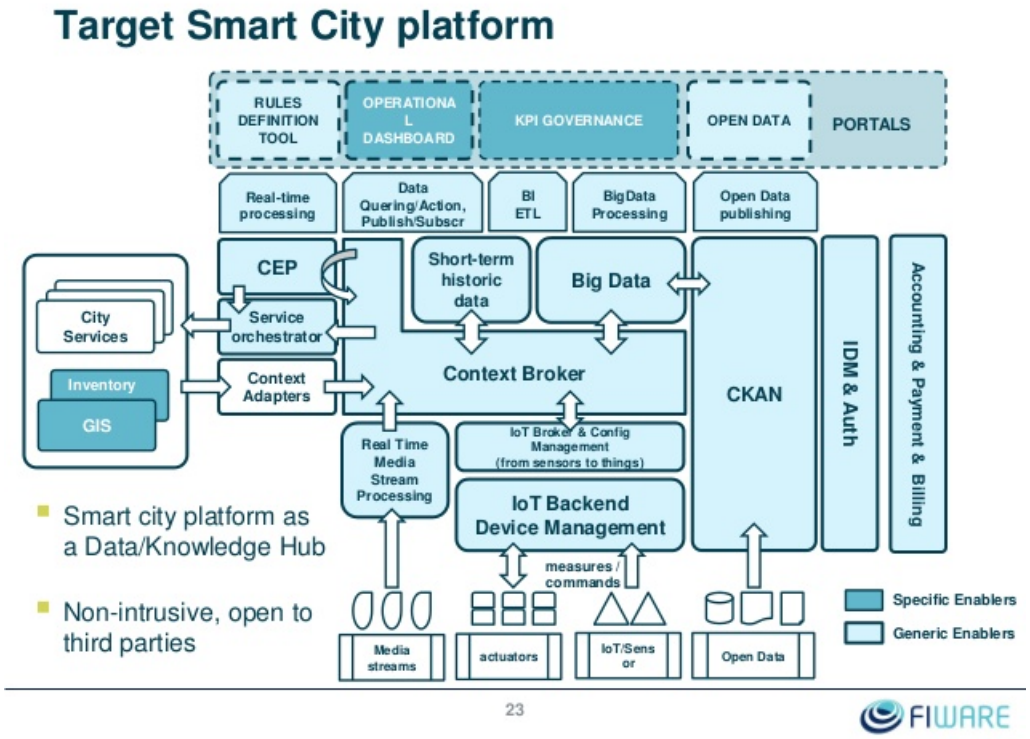

Figura 4: Arquitectura FIWARE para smartcities[19]

La Figura [4] 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 imagen), 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*[16]. Para nuestro estudio es necesario saber más 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 el que se recogen los GEs encargados de tratar con los datos.

En la Figura 5 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 conectando distintos GEs tanto del mismo grupo como es el GE *OpenData* como de

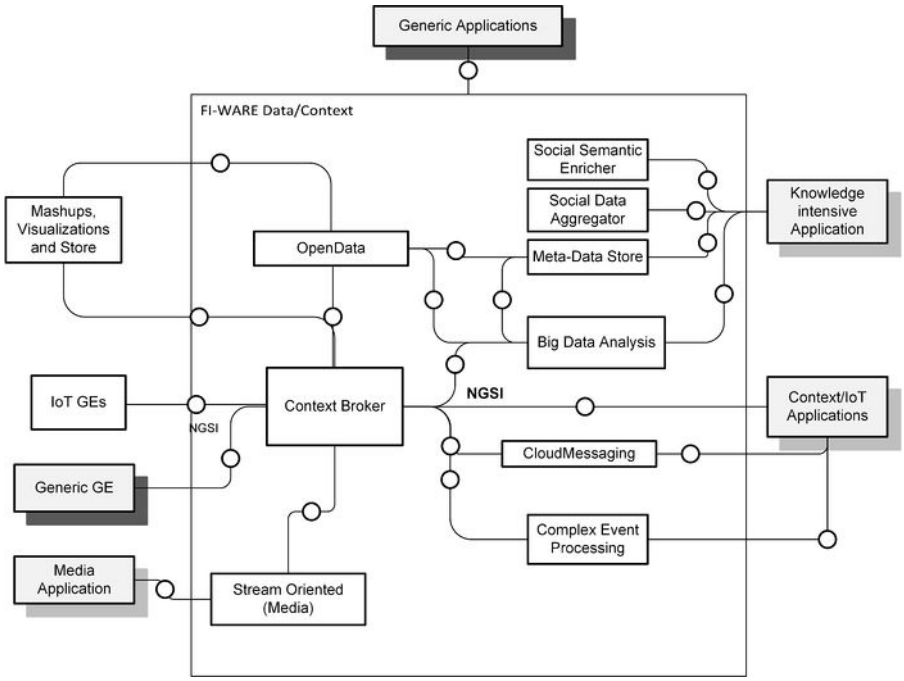

Figura 5: Fuente: FIWARE, Architecture Context Management

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* pasara los datos de múltiples sensores al *Context Broker*, el cual, a su vez, se los pasará al GE *Big Data Analysis* para 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 basa 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.

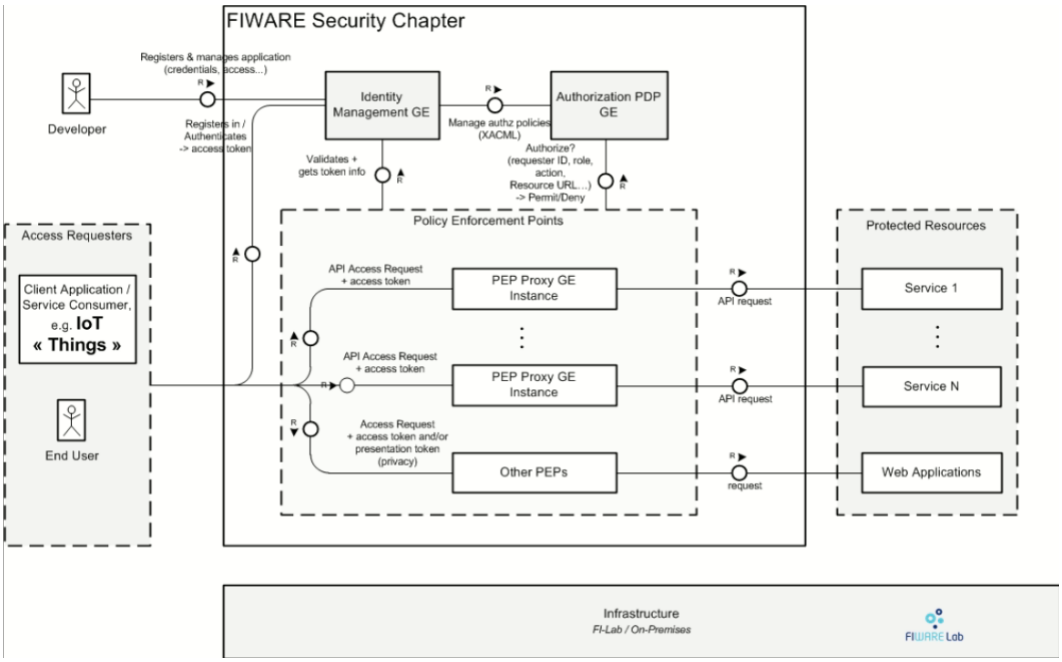

Figura 6: Fuente: FIWARE, Architecture Security

En la Figura [6] se describen las interacciones de los GEs de seguridad, tanto entre ellos como con elementos ajenos.

Un usuario, ya sea una persona física como otro sistema, que quiera acceder a alguno de los recursos protegidos por los GEs de seguridad debería 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 debería identicarse contra el GE *Identity Management* el cual devolverá un token de acceso, tras esto debería 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 todos los requisitos citados, el GE *PEP Proxy* pasará la petición al recurso protegido en cuestión y devolverá la respuesta al usuario.

4.5. 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 durante un tiempo se trabajó en un GE llamado *Privacy* encargado de incluir privacidad en la autenticación.

- ***KeyRock***[33]: Implementación del GE Identity Management[2] para el control de acceso. Permite gestionar cómo los usuarios y organizaciones acceden a los recursos del sistema. La última actualización en GitHub es del 03/10/2018.

- ***AuthZForce***[11]: Implementación del GE *AuthorizationPDP*[10] 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 tiempo de desarrollo y su última actualización en GitHub fue en abril de 2018.

- ***Wilma***[3] 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*. Útima actualización el 27/06/2018.

- ***Privacy***. Durante un periodo de tiempo, un grupo dirigido por Stephan Neuhaus[24] estuvo trabajando en un GE llamado *Privacy*. El objetivo era facilitar la autenticación preservando aspectos de privacidad. Lamentablemente este proyecto no ha seguido evolucionando desde 2015. Esto supone que cualqquier aplicación que use FIWARE debe implementar, hasta el momento actual, sus propios mecanismos de privacidad.

El objetivo principal de este trabajo es desarrollar un prototipo para un supuesto práctico, utilizando elementos de FIWARE y considerando aspectos de la privacidad de algunos datos tratados, de acuerdo con el nuevo marco normativo fijado a partir del RGPD[30] en la Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales[21]. Por ello, vamos a utilizar las posibilidades de los GEs de *Security* de FIWARE, dado que el trabajo sobre el GE *Privacy* parece que no tiene continuidad, al menos hasta este momento.

En el siguiente apartado vamos a explicar Orion [7] que es el principal y único componente obligatorio de cualquier plataforma o solución desarrollada con FIWARE, el cual aporta una función fundamental en cualquier solución inteligente: administrar la información de contexto, consultarla y actualizarla.

4.6. Orion (administra todo el ciclo de vida de la información en el conjunto de Fiware)

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.

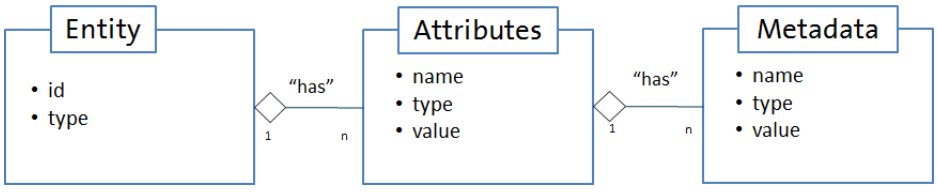

Figura 7: Modelo

Para Orion y, por tanto, para 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. Dicha información es tratada por el sistema de la misma manera. En la Figura [7] podemos ver como se estructura un elemento del contexto.

Los elementos del contexto, *entity* 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. Un elemento de contexto puede ser un sensor, con id sensorXXX que puede tener el tipo *TemperatureSensor*, uno de cuyos atributos puede ser temperatura del parking, de tipo temperatura y valor YY. Esta información de contexto se procesa y, tras eso, sirve para informar a los actores externos, permitiendoles 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.

4.7. Modelos de datos de FIWARE

A continuación, se describen los diferentes tipos de datos que usan los *Generic Enablers* predefinidos. En la página de modelos de datos de FIWARE éstos se encuentran organizados por categorías, que procederemos a analizar a continuación, para tener claro cuáles debemos considerar más relevantes cuando queremos tener en cuenta aspectos de privacidad en una aplicación.

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:** 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”.
   - **location:** Localización del lugar donde se ha producido la alerta. Representada por geometría GeoSON Obligatoria si no está presente “address”.
   - **address:** Dirección física de la alerta. Obligatoria si no está 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 (un estándar abierto y modelo cooperativo para el seguimiento de problemas civiles desarrollados 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 sería 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:

- **provider:** El proveedor del dispositivo. Según **schema.org** 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 momentos 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 él 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 *Smartcities*. 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 el se llegue a información sensible.
- **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 due na 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/TouristInformationCente”

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

    - **SmartPointOfInteraction:** Esta entidad define un lugar con tecnologíaa 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 *SmartSpots* 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 necesita 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 usa 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:** Due no 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 requiere 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 climatol lógicas 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 “address”, “PostalAddress” y/o “Text”. Estos tipos de datos definidos en schema.org/address, schema.org/PostalAddress y schema.org/Text respectivamente, contienen campos sensibles para la privacidad que debemos tener en cuenta:

    - **address**
    - **Text**
    - **PostalAddress** En el caso de PostalAddress los campos con los que debemos tener cuidadoson los referentes a las direcciones (definidos como tipo Address) ya que en ellos se pueden incluir los campos sensibles de Address mencionados anteriormente.Los campos de tipo PostalAddress son:
        - **addressCountry:** El país.
        - **addressLocality:** La localidad.
        - **addressRegion:** La región.
        - **postOfficeBoxNumber:** El número de apartado de correos para las direcciones de apartados postales.
        - **postalCode:** El código postal.
        - **streetAddress:** La dirección de la calle.

    Teniendo en cuenta todos los anteriores modelos de datos de FIWARE, hemos decidido utilizar la categoría Parking, y dentro de ella hemos utilizado atributos de OffStreetParking, OnStreetParking y ParkingGroup. Gracias a esas tres categorías hemos creado los modelos de datos para Parking y tipo de plaza.

5. Modelos de Privacidad

En Europa, la privacidad está considerada como un derecho fundamental. En 1981 la convención 108 del Consejo de Europa estableció "*la protección de los individuos con relación al proceso automático de datos personales*". En 1995, la directiva de protección de datos 95/46/EC consideró "*la protección de los individuos con respecto al proceso de datos personales y el libre movimiento de tales datos*". Por último, y antes de la presentación en 2016 del RGPD, en 2009 se presentó la directiva 009/13/EC que trata del "*proceso de datos personales y la protección de la privacidad en el sector de las comunicaciones electrónicas*". De esta introducción, queda clara la importancia que en la Unión Europea se ha dado al tema de la privacidad respecto al tratamiento de datos personales.

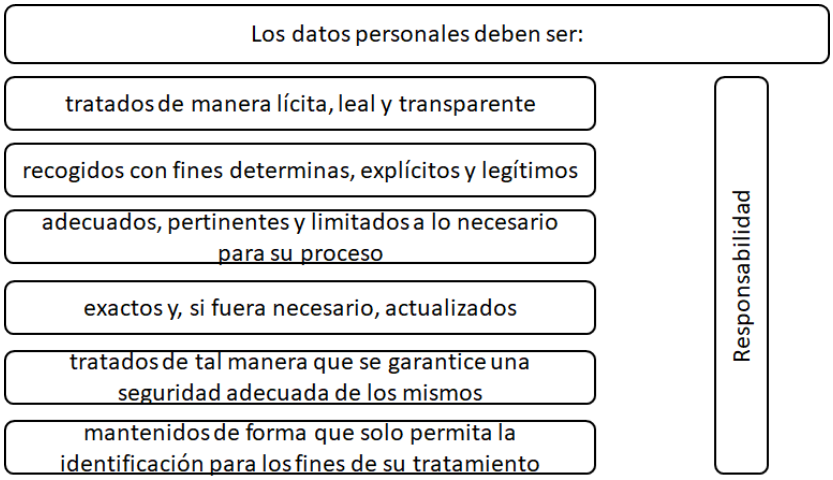

Figura 8: Principios relativos al tratamiento de datos personales (RGPD)

Por otro lado, el desarrollo de aplicaciones en los contextos de IoT (*Internet of Things*) o de *Smartcities* plantea retos relacionados con la seguridad y la privacidad de los datos. Antes de la aparición del RGPD varias empresas y organizaciones llevaban un tiempo trabajando sobre estándares relacionados con la privacidad y, especialmente, con el desarrollo de modelos arquitectónicos que consideraran la privacidad como fundamental en el desarrollo de determinadas aplicaciones. Cuando se consideran dichos modelos aparecen las funciones relacionadas con auditoría y control de la privacidad que se pueden combinar con los servicios y procesos de seguridad, ya considerados habitualmente en el marco de seguridad de la información. En este capítulo se hace un breve repaso de algunos modelos y propuestas relacionados con la privacidad de los datos.

5.1. Modelo de referencia de gestión de privacidad (PMRM)

La gestión adecuada de la privacidad es un tema complicado ya que debe funcionar en sistemas complejos de integración de información como *health IT*, servicios financieros, aplicaciones para móviles, internet de las cosas, Big Data, etc. Conseguir dicho objetivo es un reto complicado en dichos sistemas, en los que el uso de Información Personal en el sistema está gobernada por un conjunto de leyes, regulaciones, contratos, políticas operativas y tecnologías[12].

El grupo OASIS publicó en 2013 la primera versión del Modelo y metodología de referencia de administración de privacidad (PMRM, versión 1, cs01). En 2016, presentó al Comité

las modificaciones que conforman la revisión segunda de dicho estándar. Según ellos, “Los objetivos fundamentales de PMRM son permitir el análisis de casos de uso complejos para comprender y diseñar los servicios adecuados de gestión operativa de la privacidad y su funcionalidad subyacente, para implementar dicha funcionalidad en mecanismos y para alcanzar su cumplimento sobre los límites de dominios, sistemas, propiedad y políticas. Un derivado de PMRM, Arquitectura de gestión de la privacidad (PMA), se puede utilizar como una herramienta para informar del desarrollo de políticas aplicables a dominios múltiples produciendo Controles, Servicios y Funciones de privacidad, implementando mecanismos y potencialmente una Arquitectura de privacidad”[12].

Su forma de abordar la cuestión de la privacidad plantea:

- Analizar el impacto de nuevos casos de uso de privacidad para su empresa.
- Diseñar servicios operativos de gestión de la privacidad.
- Mejorar los servicios que deben ser compatibles con el RGPD.
- Determinar el uso y los requisitos de los servicios de seguridad desde un punto de vista de privacidad.
- Ser la entrada para desarrollar una arquitectura de solución de privacidad.

Para desarrollar una arquitectura de privacidad, tiene sentido investigar si las funciones de auditoría y control para la privacidad se pueden combinar con los servicios y procesos de seguridad que ya existen.

Este modelo es particularmente relevante para evaluar los casos de uso en los cuales la información personal (PI) fluye a través de los límites regulatorios, políticos, jurisdiccionales y del sistema.

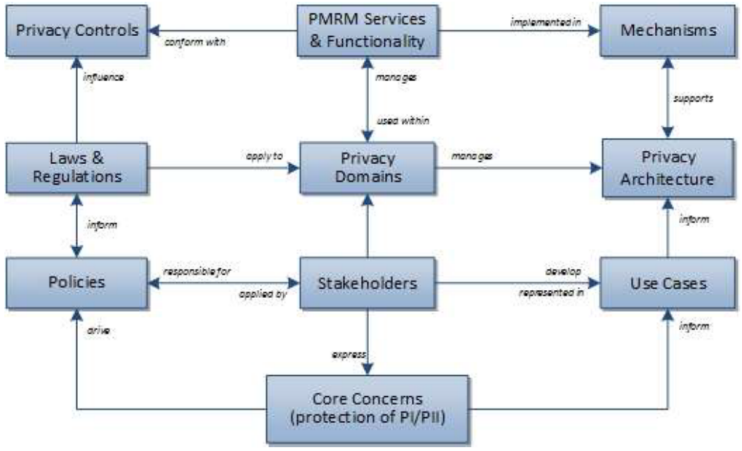

Figura 9: Modelo de referencia de gestión de privacidad (PMRM) de OASIS

De forma resumida, la PMRM consta de:

- Un modelo conceptual de gestión de la privacidad, incluyendo definiciones de los términos relevantes.
- Una metodología.

- Un conjunto de Servicios y Funciones operativas junto con las interrelaciones de estos tres elementos.

En la figura [9] se ven los diferentes bloques del modelo así como las relaciones entre ellos.

5.1.1. Modelo de gestión de privacidad

Un modelo de administración de privacidad describe cómo se pueden clasificar los diversos procesos necesarios para la administración de la privacidad.

En el documento de especificación de la metodología[12] se describe también una propuesta de las etapas del proceso para pasar de los requisitos generales a las construcción de una implementación (Figura [10]).

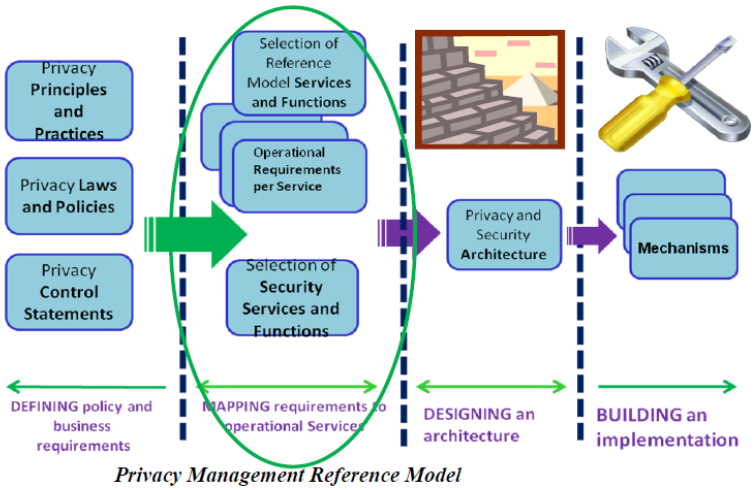

Figura 10: Pasos del Modelo de referencia de gestión de privacidad

Sobre la base de los procesos fundamentales, en cada arquitectura de solución se debe describir una vista de dichos procesos, como se aprecia en la figura [11]. Con el uso de una topología de visión general del proceso, es más fácil mapear la superposición entre la privacidad, la seguridad y los procesos y departamentos de TI y riesgos generales.

5.2. Otros modelos

Aunque PMRM es un modelo de referencia, en el que cabe perfectamente la inclusión de lo regulado por el RGPD, existen otras propuestas que contemplan los temas de privacidad de los datos como por ejemplo Synchronicity.

Synchronicity[1] es un proyecto financiado por la UE del 1 de enero de 2017 al 31 de diciembre de 2019. Uno de sus objetivos es crear una propuesta de servicios para un mercado de una única ciudad digital. El resultado esperado de la propuesta es establecer una arquitectura de referencia para aplicaciones enmarcadas en IOT y samartcities. La arquitectura que proponen se basa en el conocimiento adquirido por propuestas europeas como OASC[2], FIWARE[3], FIRE[4], etc. Previo a su propuesta arquitectónica se realizó un estudio de las dife-

[1] https://cordis.europa.eu/project/rcn/206511/factsheet/es
[2] https://oascities.org/
[3] https://www.fiware.org
[4] https://fire-in.eu/

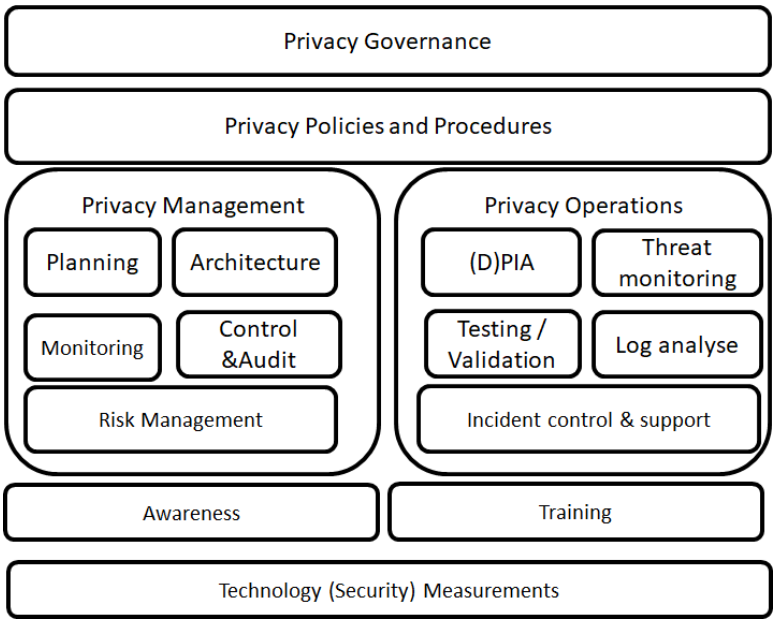

Figura 11: Modelo de gestión de privacidad

rentes propuestas arquitectónicas para smartcities desarrolladas en proyectos financiados, total o parcialmente, por la U.E.

Como consecuencia de dicho estudio, se obtuvo que dichas propuestas planteaban una arquitectura por capas, entre las que aparecía habitualmente una capa de gestión de datos y otra relacionada con la gestión de la seguridad y privacidad en los sistemas subyacentes.

En la figura [12] se presenta un esquema general de la arquitectura propuesta por Syncronicity. Se acompaña de un esquema más detallado de dicha arquitectura en que se aprecian varios aspectos:

- Un modelo arquitectónico de capas, como los tratados en las iniciativas analizadas.
- Unos bloques más trasversales en los que destaca la parte de seguridad y privacidad.

En la figura [13] se detallan las diferentes capas que considera la propuesta. En ella se aprecia la distinción entre datos privados y datos abiertos y los bloques mas trasversales de Seguridad y Privacidad, así como de Monitorización y Seguimiento. En el caso del bloque de Seguridad, Privacidad y Gobernanza, las propuestas de los elementos a considerar: gestión de identidad, protección de datos y privacidad, etc. son tomados de la propuesta de FIWARE para dichos aspectos. Hay que recordar que FIWARE no tiene definida, hasta este momento, una propuesta para el tratamiento de la privacidad y recomienda que cada aplicación desarrolle su propia solución en función de los datos a tratar y las restricciones sobre los mismos.

En los últimos años también han aparecido diversos trabajos sobre perfiles de UML para privacidad [9, 6]. Dichas propuestas están pensadas para facilitar el modelado en UML de los problemas concretos de la privacidad de los datos, pero, de nuevo, no resuelven todos los aspectos a considerar en el desarrollo de aplicaciones.

En 2018 el NIST (*National Institute of Standards and Technology*) puso en marcha el Privacy Framorks[5] para proporcionar información y materiales a las empresas y administraciones de Estados Unidos. En el contexto de arquitecturas para seguridad y privacidad está accesible el material proporcionado por *Open Reference Architecture for Security and Privacy*

[5]https://www.nist.gov/privacy-framework

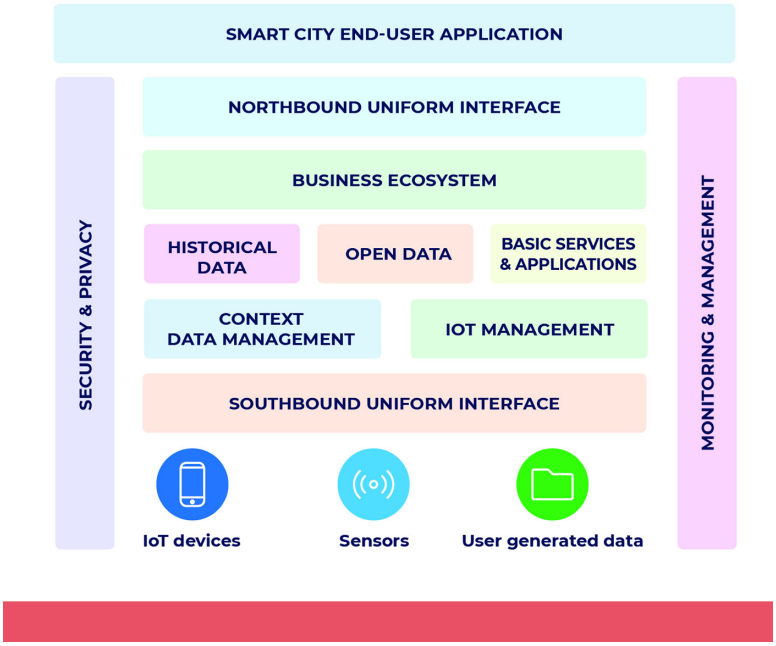

Figura 12: Versión general de la arquitectura de Synchronicity

[6] en el que respecto a privacidad hay bastante material, entre otros enlaces a algunos modelos de privacidad ya indicados anteriormente como el PMRP y otros como el *IMMA Privacy reference architecture*, publicación del Dutch Ministry of Infrastructure and the Environment de marzo de 2016. En general, el material proporcionado por ambos enlaces puede ser un punto de partida para abordar modelos arquitectónicos de la privacidad.

[6] https://security-and-privacy-reference-architecture.readthedocs.io/en/latest/

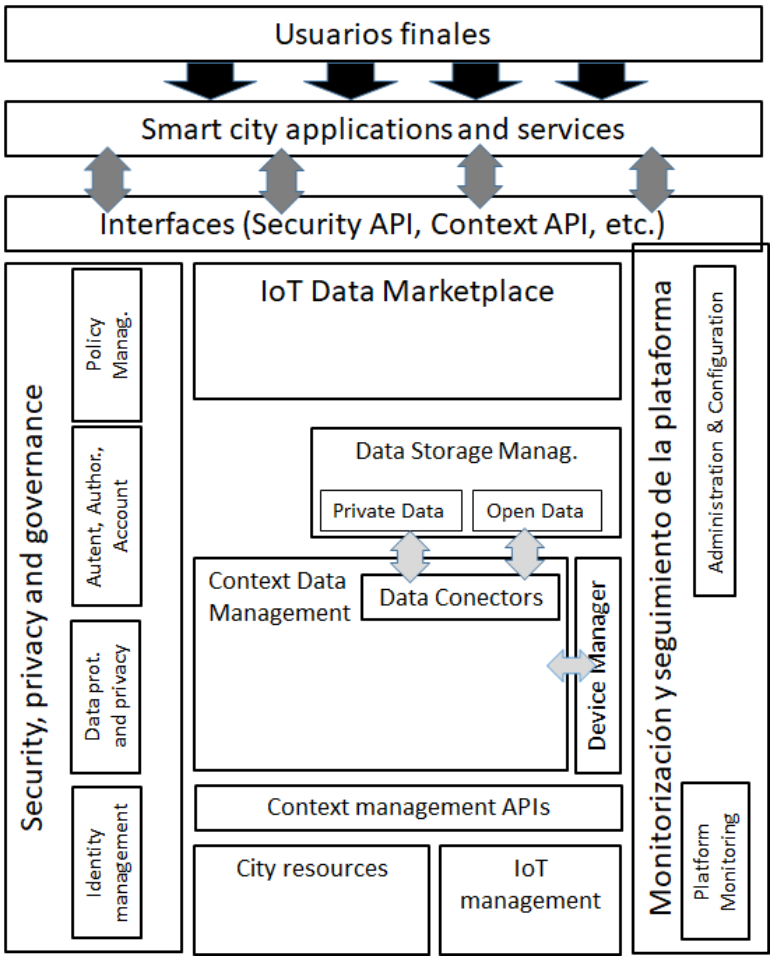

Figura 13: Versión detallada de la arquitectura de Synchronicity

6. Análisis

6.1. Introducción

En este apartado, nos concentramos en el análisis de los requisitos que ha de cumplir la aplicación a desarrollar y la especificación de dicha aplicación. El resultado de este capítulo ayudará a conseguir los objetivos planteados facilitando el diseño y la implementación de la aplicación con los requisitos identificados y las funcionalidades explicadas detalladamente.

Uno de los problemas habituales de la vida en las ciudades es el de encontrar, especialmente a determinadas horas, un lugar disponible para aparcar cuando nos desplazamos a un lugar dentro de la misma. Aunque en muchas ciudades existen lugares para aparcar tanto externos (*onstreet*) como en aparcamientos ubicados en locales, subterráneos, etc. (*offstreet*), el problema es localizar un lugar dónde aparcar el vehículo con la restricción añadida, en la mayoría de los casos, de la proximidad al lugar de destino.

Existen estudios que analizan el tiempo medio de espera en la localización de dicho lugar donde aparcar y la incidencia que tiene dicho tiempo en el conductor. Por ello, uno de los campos de interés de las aplicaciones para *smartcities* es el desarrollo de sistemas de localización de espacios vacíos en los lugares de aparcamientos, que no garantiza la disponibilidad de los mismos en el momento de acceso, y, en algunos casos, sistemas de reserva previa de sitio para minimizar el tiempo de acceso al lugar de aparcamiento deseado.

6.2. Funcionalidad requerida para la aplicación

Los requisitos que se van detallar en las siguientes secciones van a ayudar a entender cuáles son las necesidades exactas de los usuarios del sistema que se desarrollará.

6.2.1. Requisitos funcionales

Como requisito funcional entendemos requisitos que especifican la funcionalidad que el sistema debe proporcionar a los usuarios. Son declaraciones de servicios que el sistema tiene que ofrecer, la forma en que el sistema debe responder a determinadas entradas y cómo el sistema debería comportarse en situaciones particulares.

Los requisitos funcionales que forman nuestro sistema son los siguientes:

- Respecto al **usuario** de la aplicación:

    - **RF-01: Registrar usuario:** El usuario debe poder registrarse en el sistema introduciendo los datos que le sean demandados.
    - **RF-02: Identificar Usuario:** El usuario debe poder acceder al sistema introduciendo el mail de usuario y la contraseña.
    - **RF-03: Cerrar sesión:** El usuario debe poder cerrar sesión y salir del sistema.
    - **RF-04: Modificar datos del Usuario:** El usuario, una vez identificado en el sistema, podrá consultar y modificar ciertos datos.
    - **RF-05: Dar de baja el Usuario:** El usuario, una vez identificado en el sistema, podrá darse de baja en el sistema.
    - **RF-06: Visualizar reservas del usuario:** El usuario, una vez identificado en el sistema, debe poder visualizar las reservas realizadas.

- **RF-07: Realizar reservas:** El usuario, una vez registrado, debe poder realizar reservas.
- **RF-08: Cancelar reservas:** El usuario debe poder cancelar sus reservas.
- **RF-09: Consultar disponibilidad de plazas:** El usuario debe poder consultar la disponibilidad de las plazas estando registrado en el sistema.

Ahora explicaremos cada Requisito Funcional más detalladamente.

- **RF-01: Registrar usuario:**
  1. La aplicación debe permitir al usuario introducir sus datos en el formulario de registro.
  2. El sistema se encargará de validar los datos.
  3. El sistema mostrará un mensaje de error si alguno de los datos es incorrecto o no cumple las condiciones especificadas del formulario de registro.
  4. En el caso de que la validación sea correcta, el sistema se encargará de guardar los datos del usuario en base de datos.
  5. La aplicación mostrará un mensaje de bienvenida al usuario y le redirigirá a la pantalla principal con su sesión ya iniciada.
- **RF-02: Identificar Usuario:**
  1. Para iniciar sesión el usuario deberá identificarse con su email de usuario y contraseña correspondiente
  2. El sistema se encargará de validar y permitir o denegar el acceso a la aplicación.
  3. El sistema mostrará un mensaje de error en el caso de que la validación no sea correcta.
  4. En el caso de que la validación sea correcta, se mostrará un mensaje de bienvenida al usuario y le redirigirá a la pantalla principal.
- **RF-03: Cierre de sesión:**
  1. Cualquier usuario de la aplicación debe poder finalizar sesión en la aplicación.
  2. El usuario será redirigido a la pantalla principal, pero en este caso sin estar logado.
- **RF-04: Modificar Datos del Usuario:**
  1. Debe existir una pantalla en el sistema que permita al usuario poder consultar o modificar “ciertos” datos de su cuenta.
  2. Si modifica algún dato el sistema lo validará.
  3. En caso de que la validación sea correcta, se actualizarán sus datos en la base de datos.
- **RF-05: Dar de baja el Usuario:**
  1. El usuario, una vez identificado en el sistema, debe poder darse de baja en el sistema.
  2. Debe existir una pantalla en el sistema que permita al usuario darse de baja.
  3. Si el usuario se da de baja el sistema lo validará, siempre y cuando no tenga ninguna reserva activa.

4. En caso de que la validación sea correcta, se actualizarán sus datos en la base de datos.

- **RF-06: Visualizar reservas del usuario:**
  1. El usuario, una vez identificado en el sistema, debe poder visualizar las reservas realizadas.
  2. El usuario podrá consultar sus reservas activas.
     - Todas
     - Por fecha
     - Por ciudad
- **RF-07: Realizar reserva:**
  1. El usuario, una vez registrado, debe poder realizar reserva de la plaza.
  2. El usuario podrá visualizar las plazas disponibles por fecha y lugar (ciudad).
  3. Comprobar las características de una plaza.
- **RF-08: Cancelar reserva:**
  1. El usuario, una vez visualizadas sus reservas, podrá seleccionar aquellas que desee cancelar.
- **RF-09: Consultar disponibilidad de las plazas:**
  1. El usuario debe poder consultar la disponibilidad de las plazas estando registrado en el sistema. (tipo de plaza, fechas, etc).

- Respecto **al administrador** de la aplicación:

  - **RF-01: Visualizar Usuarios:**
    1. El administrador puede ver los usuarios registrados en el sistema.
    2. El administrador puede modificar datos de un usuario.
    3. El administrador puede dar de baja a un usuario.
  - **RF-02: Consultar Reservas:**
    1. El administrador puede ver las reservas por usuario.
    2. El administrador puede ver las reservas por fecha.
    3. El administrador puede ver las reservas por ciudad y parking.

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 debe ser fácil de usar, para cualquier usuario sin conocimientos de informática.
- **RNF-02:** La aplicación podría ser ejecutada por cualquier navegador que use HTML 4.0 o superior.
- **RNF-03:** La aplicación deberá conectarse a Orion que cumplirá una función de base de datos.
- **RNF-04:** La aplicación no consumirá demasiados recursos en el lado del usuario ni en el lado del servidor.
- **RNF-05:** La aplicación será intuitiva y eficiente.

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 sobre cada usuario registrado: su mail (que usará para iniciar sesión), password, nombre y apellidos, fecha de alta en el sistema, fecha de baja (fecha en la que decide borrar su cuenta).
- **RI-02:** Hay que almacenar información sobre su actividad: reservas hechas, fecha en la que se realizó la reserva, plaza reservada, y período para el que se reserva. También estado de cada reserva (activa, plaza ocupada, finalizada), y fecha en la que terminó la ocupación (que no tiene por qué coincidir con la que había previsto inicialmente).
- **RI-03:** La gestora almacenara información sobre el parking (nombre, calle, categoría, nº plazas totales).
- **RI-04:** La gestora almacenara datos de las plazas del parking (nombre del parking al que pertenece, el nº de plaza, tipo de plaza, estado de plaza).

6.3. Roles

En el documento de requisitos funcionales, se ha indicado que existen dos tipos de usuario en este momento, el usuario con rol de Administrador y el usuario con rol de Usuario. El usuario con rol de **Administrador**, representa a la persona que actualmente realiza las operacines de gestión del parking, como por ejemplo ver todos los usuarios dados de alta en el sistema, modificar datos de un usuario, sonsultar datos de las reservas hechas en el sistema, etc. También el administrador podrá ver y gestionar tres tipos de reservas distintas: Historico de reservas (reservas que se han realizado en el parking), reservas realizadas (Son todas las reservas que estan en proceso o han sido realizadas con exito) y por último tendremos las reservas canceladas (Son todas las reservas que han sido canceladas). El administrador además podrá exportar todos estos datos para las estadísticas del parking.
Por otro lado el usuario con el rol de **Usuario** es el usuario registrado en nuestro sistema, el cual, solo podrá realizar reservas,ver sus reservas y modificarlas al igual que sus datos personales.

6.4. Derechos del RGPD que trataremos

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

- **Derecho de Acceso:** Los usuarios deberán saber para que se están usando sus datos.
- **Derecho de Cancelación:** El usuario puede solicitar la supresión de sus datos que resulten ser excesivos.
- **Derecho de Rectificación:** Los usuarios tienen derecho a poder modificar sus datos si estos son incorrectos.
- **Derecho al Olvido:** Esta derecho nos obliga como empresa a notificar cuando los datos recopilados van a dejar de usarse.

6.5. Resolución de los Derechos del RGPD para los usuarios

- **Derecho de Acceso:** En el prototipo, el tratamiento de los datos que se realizará será para la autenticación de las peticiones de los usuario.
- **Derecho de Cancelación:** Los usuarios podrán borrar total o parcialmente sus datos.
- **Derecho de Rectificación:** Los usuarios tienen derecho a poder modificar sus datos si estos son incorrectos.
- **Derecho al Olvido:** Los datos que recogemos de los usuarios serán utilizados hasta que estos se den de baja.

6.6. Sistema de Entrada y Salida y ocupación de la plaza

La idea es la creación de un parking que permita la entrada de los vehículos de forma automática. El sistema estará formado por un lector que estará conectado a una base de datos y que tendrá que estar en espera a que un tag entre en su área de cobertura.

Esta cámara estará situado a la entrada del aparcamiento, en una posición que posibilite la lectura de las matrículas tanto a la entrada como a la salida del vehículo. Una vez la matricula haya sido identificada se deberá comprobar si está dado de alta en la base de datos con tal de saber si se le puede abrir la puerta.

Cada plaza dispondrá de un sistema de control acceso, un pivote retráctil con una cámara que controlará que cada vehículo ocupe la plaza anteriormente reservada. La cámara al igual que el sistema de entrada/salida comprobara que este guardada en nuestra base de datos.

6.7. Gestor del Parking

Realizaremos un ejemplo de aplicación, con la cual gestionaremos el parking (crearemos parkings y plazas de dicho parking). Esta gestión irá ajena a nuestra aplicación ya que se encargara el ayuntamiento de cada ciudad de la creación de cada parking y dichas plazas. Cada parking tendra que cumplir una serie de requisitos, (nombre del parking, dirección, ciudad y seguridad) al igual que las plazas.

6.8. Posibles Generic Enablers a utilizar

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

Para hacerlo, comencé utilizando el GE de Wilma junto con Orion Context Broker. Luego configuré el GE Identidad de Identity Manager con Keyrock y el GE AuthzForce.

Después de unas semanas de configuraciones y muchos intentos, finalmente no pude hacer que estos tres Generic Eneablers de seguridad funcionaran bien, autentificando y autorizando las solicitudes de Orion Context Broker utilizando PEP Proxy nivel 2, ya que el nivel 2 de autorización no es suficiente para garantizar que la información de un usuario registrado en nuestra aplicación no pueda mantener la seguridad de sus datos ya que la informacion está en los encabezados de la solicitud. Para crear políticas de autorización basadas en encabezado debe usar el nivel 3: autorización XACML (Control de Acceso del lenguaje de marcas Extensibles).

Tras 1 mes de investigación y no conseguir la seguridad necesaria debido a la carencia de una documentación decente, optamos por utilizar .Net que se encargará facilmente de este asunto.

6.9. 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 mismos datos, aunque de diferente manera. Estos datos serían los siguientes.

- **Usuario:** Representa a los usuarios registrados en la aplicación. Su identificador sera el correo electrónico del usuario.
- **Plaza:** Representa las plazas que reservan los usuarios con los vehículos. Una plaza puede tener una o más reservas, pero cada reserva solo tiene una plaza.
- **Reserva:** Representa los vehículos que están estacionados actualmente. Su identificador es la matrícula del vehículo en cuestión.

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

Se plantea hacer una aplicación web desde la cual los usuarios puedan registrarse e identificarse para que estos, los usuarios identificados, puedan reservar una plaza de parking, para que puedan estacionar en una ciudad y, al finalizar la reserva, cobrarles en consecuencia con el tiempo de uso.

Los datos solo se guardarán 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 [22].

6.9.2. Integración media con Orion

En este supuesto lo que queremos hacer es una aplicación para una empresa que tiene los datos del parking y de las plazas de parking guardados en Orion, mientras que nosotros

ofrecemos una página web para que los usuarios puedan registrarse y reservar plazas de parking. En este supuesto se usará una segunda base de datos que se encargará de los datos necesarios para guardar las sesiones, por lo que necesitamos ser 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 será como se muestra en la Figura [14].

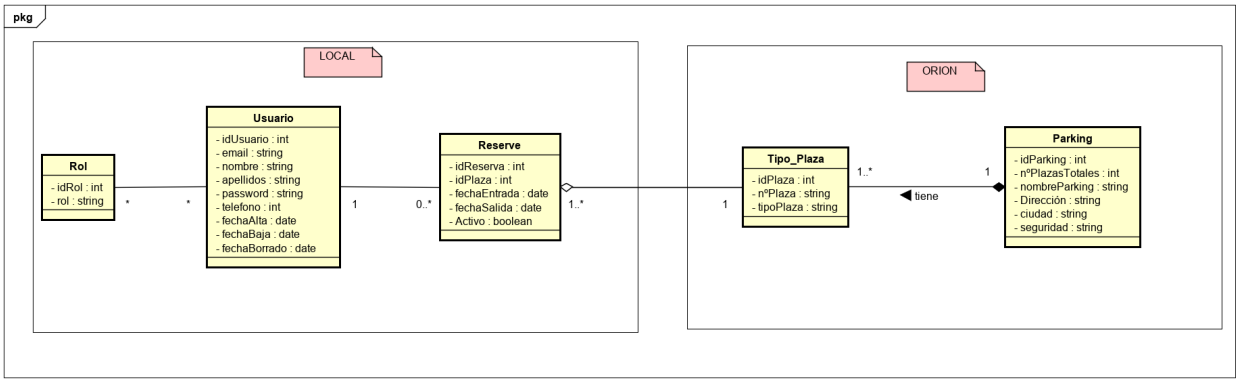

Figura 14: Modelo de datos para la integración completa

Las tablas que obtenemos para la BBDD son las siguientes:

Atributo	Tipo	Clave	Descripcion
IdPerson	Integer(10)	Primaria	Identificador único del usuario
nombre	Varchar(50)	-	Nombre de pila del usuario
apellido	Varchar(50)	-	Apellidos del usuario
Telefono	Varchar(9)	-	Teléfono de contacto del usuario
Email	Varchar(9)	-	Email de contacto del usuario
FechaAlta	Date	-	Fecha de Alta del usuario
FechaBaja	Date	-	Fecha de Baja del usuario
FechaBorrado	Date	-	Fecha en la que se borran los datos del usuario

Tabla 3: Tabla: Tabla Usuario BBDD

Atributo	Tipo	Clave	Descripcion
idRol	Integer(10)	Primaria	Identificador único del Rol
rol	Varchar(50)	-	Rol del usuario

Tabla 4: Tabla: Tabla Rol BBDD

Atributo	Tipo	Clave	Descripcion
IdReserva	Integer(10)	Primaria	Identificador único de la Reserva
IdPlaza	date	-	Identificador de la Plaza
fechaEntrada	date	-	Fecha Inicio de la reserva
fechaSalida	date	-	Fecha Fin de la reserva
Plaza_Id	Integer(10)	Foránea	Identificador único de la plaza de la reserva

Tabla 5: Tabla: Tabla Reserve BBDD

La tabla “Usuario” tiene una relación de muchos a muchos con la tabla “Rol”, ya que un rol tendrá varios usuarios en el sistema, cuya clave foránea es el “id” de la tabla “Rol”.

La tabla “Usuario” tiene una relación de uno a muchos con la tabla “Reserva”, ya que una persona del sistema podrá tener varias reservas. En este caso, al ser una relación de uno a muchos, se genera una tabla intermedia que almacena las claves primarias de ambas tablas y así poder acceder desde un usuario a sus reservas.

En este caso nosotros guardaríamos los datos de la persona y los datos de la reserva ya que serían los datos más importantes y sensibles de la aplicación y nosotros le daríamos toda la seguridad posible. De esta manera dejamos libre de carga a la empresa dueña del parking ya que no trataría ningún dato de carácter sensible.

La tabla “Reserva” tiene una relación de uno a muchos con la tabla “Tipo Plaza”, ya que una reserva tendrá una plaza relacionada, y tambien la misma plaza puede tener muchas reservas, cuya clave foránea es el “id” del tipo de plaza.

La tabla “Parking” tiene una relación de uno a muchos con la tabla “Tipo Plaza”, ya que un parking tendrá asociado muchos tipos de plazas, cuya clave foránea es el “id” del tipo de plaza.

Tenemos que destacar, que ORION, utiliza MongoDB que es una base de datos no relacional, es decir no es como las típicas bases de datos SQL (MySQL, Oracle, PostgreSQL, etc...) donde existen relaciones entre una tabla y otra.

Nosotros vamos a interacionar con nuestra base de datos en local y la base de datos de ORION.

6.9.3. Casos de Uso

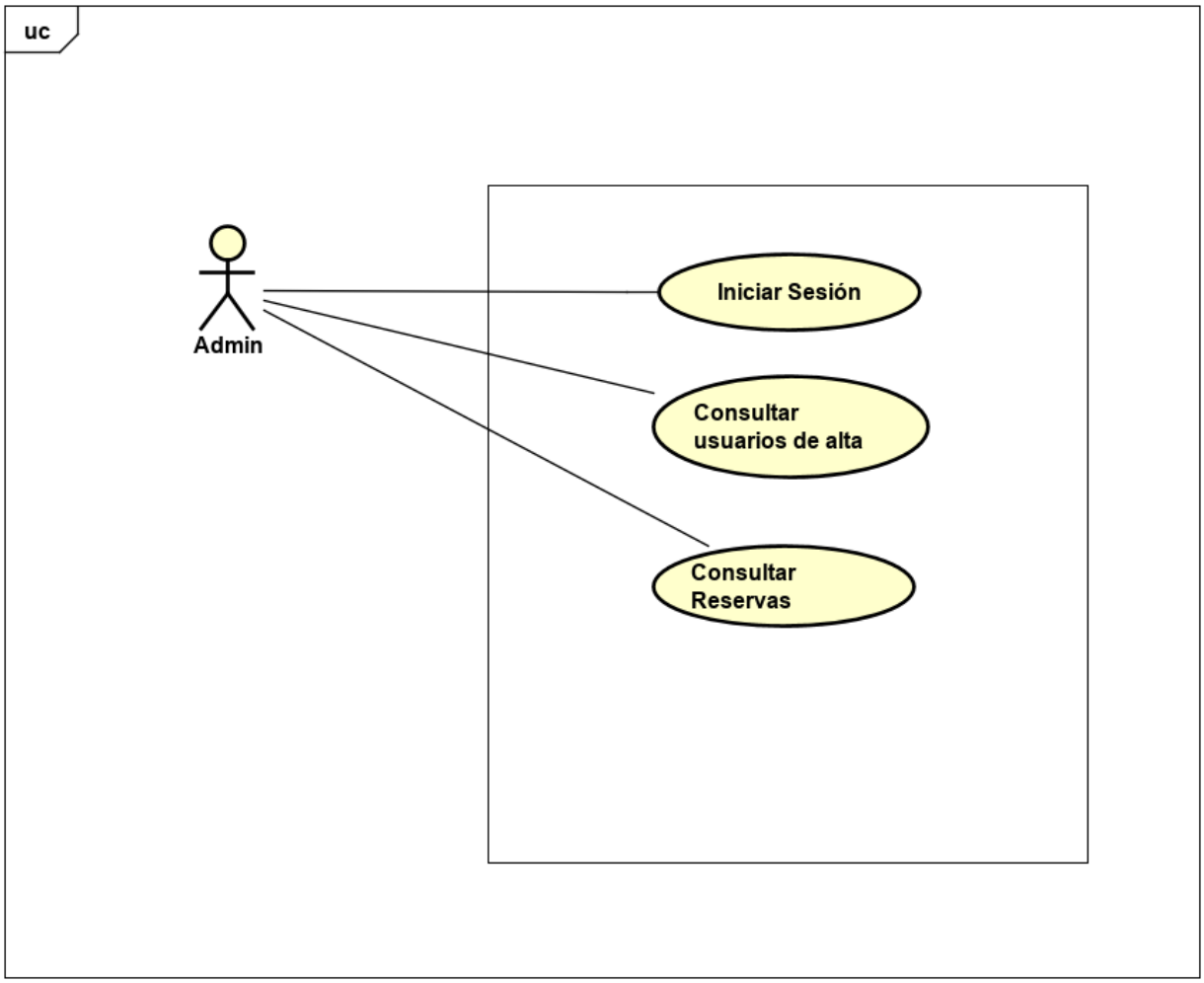

Figura 15: Diagrama de casos de uso del actor Administrador.

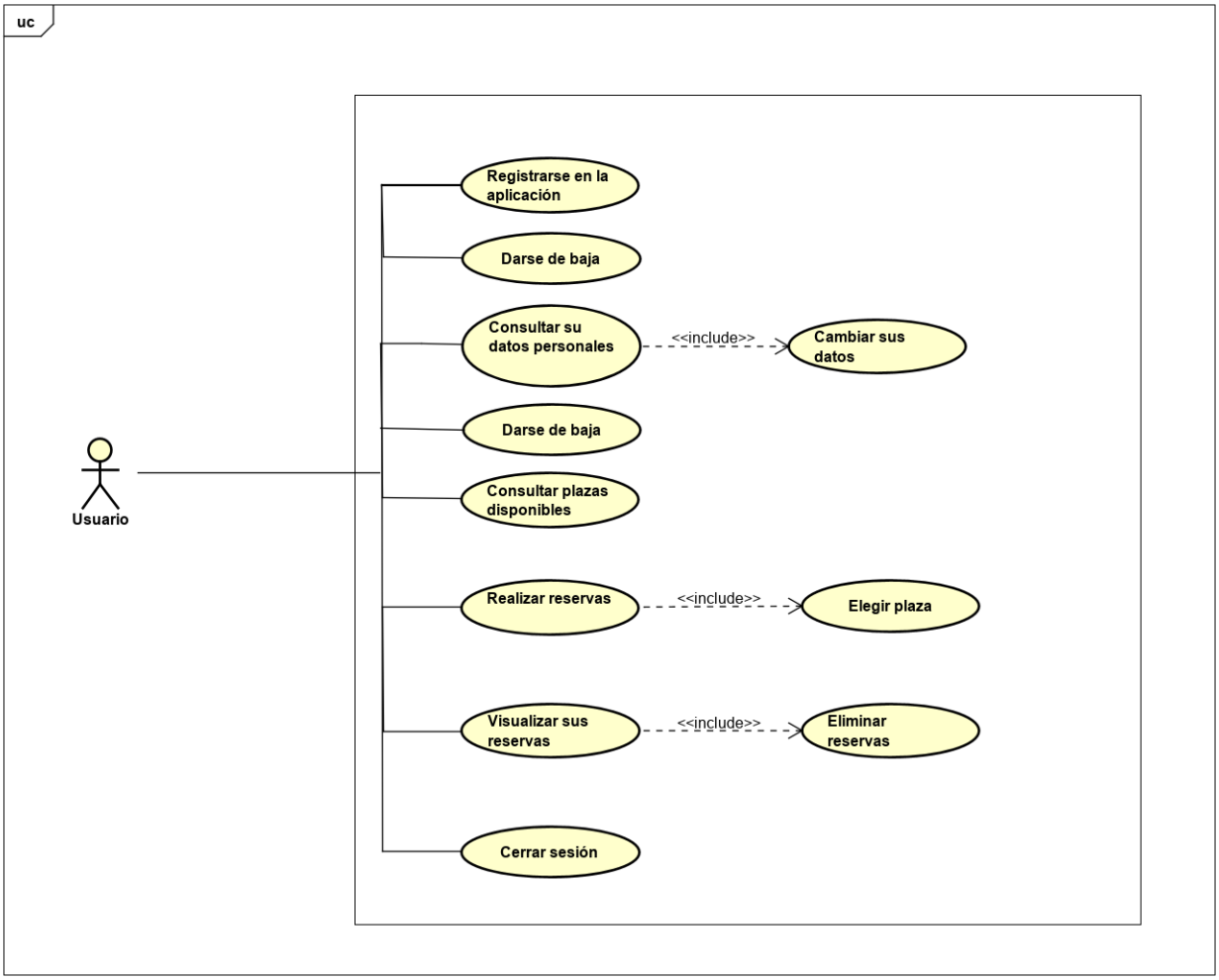

Figura 16: Diagrama de casos de uso del actor Usuario.

En la figura [15] se muestra el diagrama de caso de uso correspondiente a la figura del actor **Administrador** el cual es la generalización del actor principal que tiene la aplicación, el Usuario. En este diagrama se presentan las funcionalidades que realizan ambos roles. En el diagrama de la figura [16], se muestran los casos de uso del actor **Usuario.**

Cada uno de estos casos de uso que se observan en los diagramas, tiene una secuencia de acciones que determinan el escenario o los escenarios en los que se puede afrontar dicho caso de uso. Normalmente, cada caso de uso está formado por un escenario que denominamos como secuencia normal, en la que se termina con éxito lo que se quería realizar y uno o varios escenarios secundarios donde por algún motivo, no se termina de realizar el caso de uso. La descripción de cada uno de los CU se muestra a continuación:

CU-01	Registro en la aplicación	
Descripción	El sistema deberá permitir al usuario registrase en la aplicación.	
Precondición	El usuario debe acceder mediante navegador web.	
Secuencia Normal	Paso	Acción
	1	El sistema solicita diferentes datos personales, entre ellos el email y contraseña.
	2	El actor usuario introduce los datos.
	3	El sistema comprueba si todos los datos del usuario estan introducidos correctamente.
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación y el caso de uso queda sin efecto
	3a	Si las credenciales no son correctas, muestra un mensaje de error y vuelve al paso 1.

Tabla 6: Caso de Uso: Registrarse en la aplicación

CU-02	Iniciar Sesión	
Descripción	El sistema deberá permitir al usuario iniciar sesión.	
Precondición	El usuario debe acceder mediante navegador web.	
Secuencia Normal	Paso	Acción
	1	El sistema solicita las credenciales de acceso, correo del usuario y contraseña.
	2	El actor usuario introduce las credenciales.
	3	El sistema comprueba con las credencia les si el usuario existe en el sistema.
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación, y el caso de uso queda sin efecto.
	3a	Si las credenciales no son correctas, muestra un mensaje de error y vuel ve al paso 1.

Tabla 7: Caso de Uso: Iniciar sesión

CU-03	Consultar datos Personales	
Descripción	El sistema deberá permitir al usuario consultar sus datos personales.	
Precondición		
Secuencia Normal	Paso	Acción
	1	El actor Usuario selecciona consultar sus datos personales.
	2	El actor usuario introduce las modificaciones correspondientes
	3	El sistema validara las modificaciones.
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación y el caso de uso queda sin efecto.
	3a	El sistema no valida las modificaciones y devuelve un mensaje de error.

Tabla 8: Caso de Uso: Consultar datos personales

CU-04	Consultar Plazas disponibles	
Descripción	El sistema deberá permitir al usuario consultar las plazas disponibles.	
Precondición		
Secuencia Normal	Paso	Acción
	1	El usuario solicita el intervalo de fechas en el que se quieren obtener los resultados de las plazas.
	2	El actor usuario introduce las fechas.
	3	El sistema muestra los datos de las plazas con las especificaciones marrcadas en el paso 2
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación, y el caso de uso queda sin efecto.

Tabla 9: Caso de Uso: Plazas disponibles

CU-05	Realizar reservas	
Descripción	El sistema deberá permitir al usuario realizar la reserva de plazas.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	Paso	Acción
	1	El usuario indica las fechas en las que quiere realizar la reserva
	2	El sistema muestra las plazas disponibles en ese rango
	3	El usuario realiza la reserva.
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación, y el caso de uso queda sin efecto

Tabla 10: Caso de Uso: Realizar reservas

CU-06	Visualizar sus reservas	
Descripción	El sistema deberá permitir al usuario visualizar sus propias reservas.	
Precondición	El usuario debe estar identificado en el sistema.	
Secuencia Normal	Paso	Acción
	1	El actor Usuario selecciona ver las reservas realizadas.
	2	El sistema muestra todas las reservas.
	3	El usuario puede ver los detalles de cada reserva
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación, y el caso de uso queda sin efecto.

Tabla 11: Caso de Uso: Visualizar sus reservas

CU-07	Cerrar sesión	
Descripción	El sistema deberá permitir al usuario cerrar sesión.	
Precondición	El usuario debe estar logado en la aplicación	
Secuencia Normal	Paso	Acción
	1	El actor Usuario quiere salir de la aplicación
	2	El actor usuario Pulsa cerrar sesión
	3	El sistema cierra la sesión y carga la pantalla de login
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación, y el caso de uso queda sin efecto.

Tabla 12: Caso de Uso: Cerrar sesión

CU-08	Darse de Baja	
Descripción	El sistema deberá permitir al usuario darse de baja.	
Precondición		
Secuencia Normal	Paso	Acción
	1	El actor Usuario selecciona consultar sus datos personales.
	2	El actor usuario quitara el check de activo
	3	El sistema validara las modificaciones.
Excepciones	Paso	Acción
	2a	El actor usuario cancela la operación y el caso de uso queda sin efecto.
	3a	El sistema no valida las modificaciones y devuelve un mensaje de error.

Tabla 13: Caso de Uso: Darse de baja

7. Arquitectura y Diseño

7.1. 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 sería nuestro Modelo.

- **Arquitectura Rest**

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

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

  - **Nivel 0 (POX)**

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

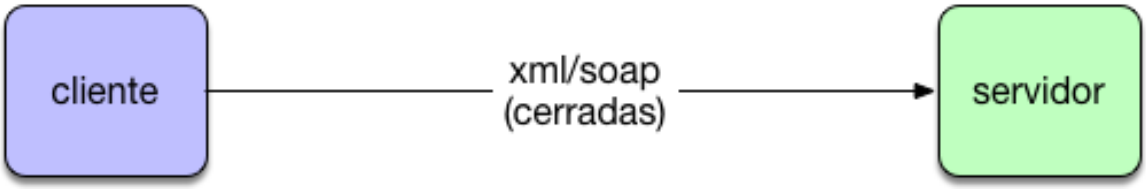

Figura 17: Arquitectura REST Nivel 0. Fuente: Arquitectura Java

  - **Nivel 1 (Recursos)**

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

  - **Nivel 2 (Verbos HTTP)**

    En nivel 2 evoluciona el concepto anterior al homogeneizar las urls y los verbos que se utilizan para cada operación. En este nivel las urls se definen por el nombre del recurso y los verbos están limitados a GET para obtener los datos, POST para añadir nuevos elementos, PUT para actualizar los datos y DELETE para borrarlos.

    Este nivel es el ofrecido por Orion y, por lo tanto, es el que emplearemos.

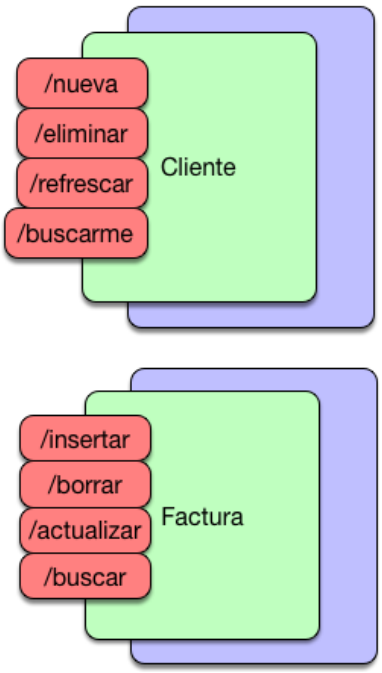

Figura 18: Arquitectura REST Nivel 1. Fuente: Arquitectura Java

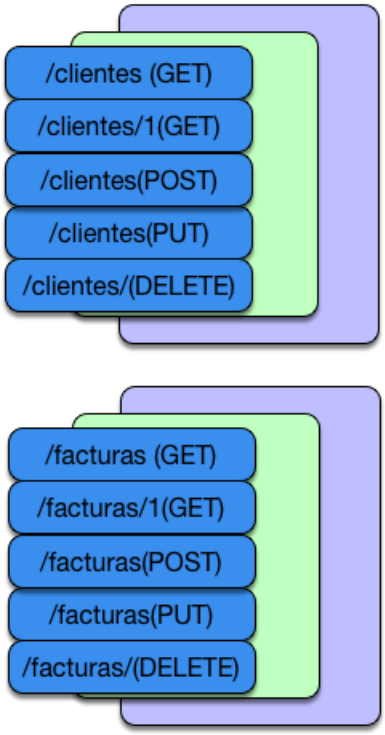

Figura 19: Arquitectura REST Nivel 2. Fuente: Arquitectura Java

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

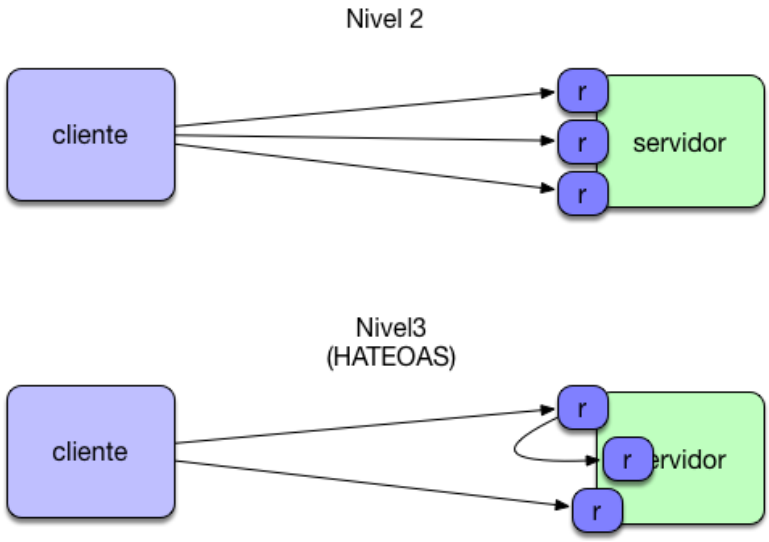

Figura 20: Arquitectura REST Nivel 3. Fuente: Arquitectura Java

En la figura [21] se muestra el diagrama de despliegue del sitema. A continuación, se describen brevemente los componentes principales.

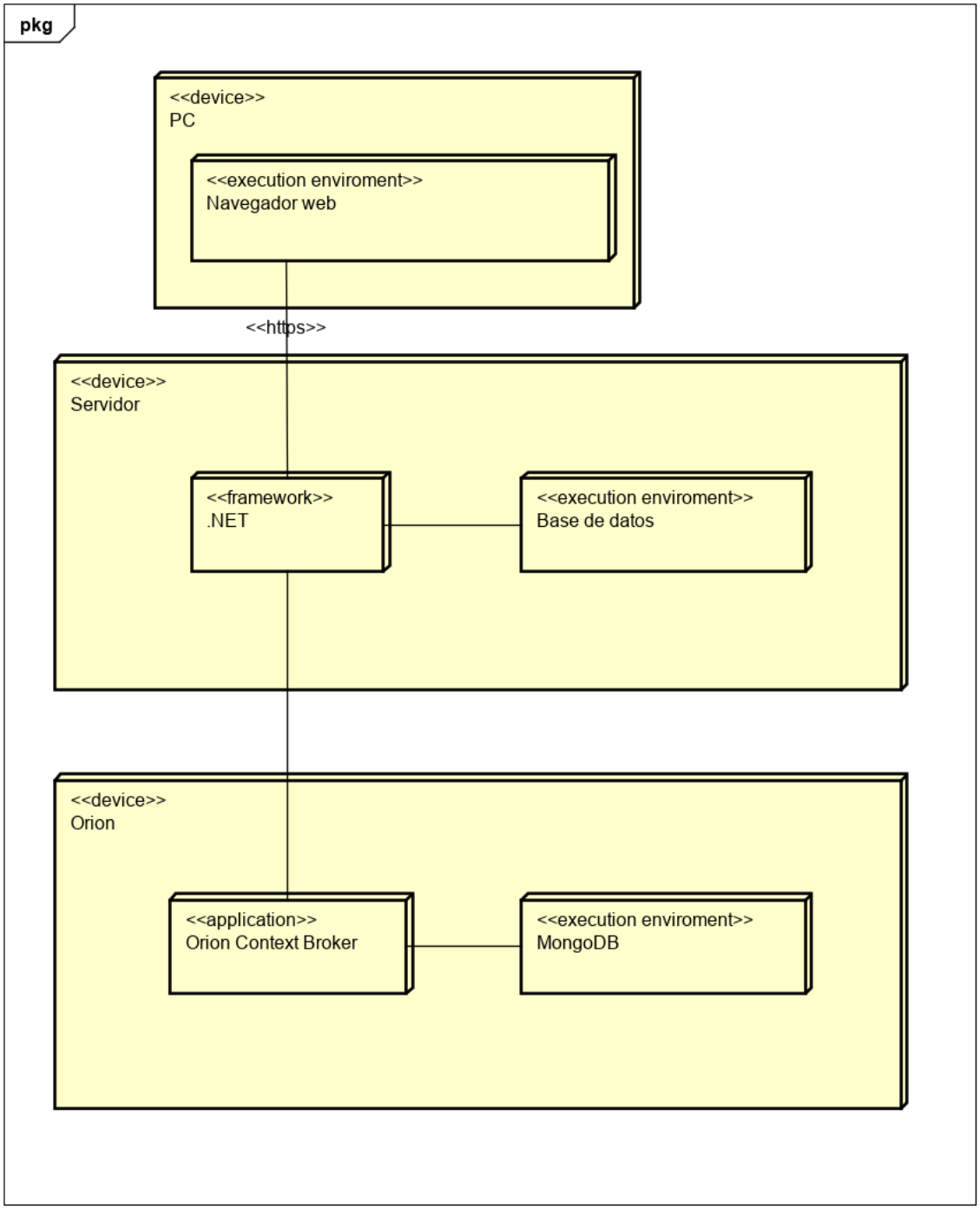

Figura 21: Diagrama de despliegue del Sistema

- **«device» Navegador web.** Es la parte del cliente donde se van a realizar las peticiones.
- **«device»Servidor.** Donde se alojará el código de la aplicación para que sea interpretado por c#.
- **«framework» .NET.** Utilizado para facilitar el desarrollo de la aplicación y que ha servido de esqueleto para realizar todas las funcionalidades requeridas.
- **«execution enviroment» Gestor de Base de Datos.** En este caso, el gestor se sitúa en el mismo servidor donde vamos a tener alojado el código de la aplicación. Si

en algún momento se decidiese cambiar la localización de la Base de Datos, tendríamos que realizar cambios en el fichero de configuración de .NET, más concretamente en el *web.config*.

- **«device» Servidor**. Donde se alojará el código de la aplicación de Fiware.

7.2. Diseño en base a la seguridad

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

- **Balanceo de riesgo y seguridad.** Cuanta más complejidad tenga la aplicación, tendrá mas vulnerabilidades. Hay que considerar las medidas de seguridad necesarias que ayuden a mitigar cualquier característica que la haga vulnerable. Estos son aspectos como la validación de los campos de entrada, y evitar inyecciones SQL. De esto último, en gran medida se ocupa el framework utilizado. Una funcionalidad obvia, pero que necesita de un cuidado especial es el control de acceso de los usuarios. Hay que evitar que un usuario de tipo A acceda a la funcionalidad de un usuario de tipo B.
- **Ataques URL de tipo semántico.** Tener especial cuidado con estos ataques, estos son los que por medio de la manipulación de URLs poder conocer ciertos datos de importancia como podrán ser las credenciales de acceso a la aplicación. En este fragmento de código, vemos que si se utilizará una implementación del formulario mediante el método GET, quedarían a la vista las credenciales de acceso usuario y contraseña.
  “https://localhost:44326/Home/Index/login?user=pepe&password=1234”
  Para evitar esto, utilizaremos POST hacia el servidor en aquellos casos que la información sea confidencial y GET en aquellos casos en los que los datos de entrada no sean tan importantes. Estos últimos podrían ser por ejemplo la filtración del número de plaza.
- **Inyección SQL** Evitaremos que un atacante intente introducir consultas que no sean legítimas por medio del filtrado y escapado de los datos en el lado del servidor.
- **Ataques por fuerza bruta.** Método que utiliza la prueba y error para poder acceder a ciertos datos dentro de la aplicación. En el diseño del control de acceso de los usuarios, si estos introducen mal cualquiera de los dos datos que se piden para acceder (correo electrónico y contraseña), se mostrará siempre el mismo mensaje de error, para así evitar que un usuario que esté intentando acceder por fuerza bruta sepa si está introduciendo bien o mal el correo electrónico o solo la contraseña.
- **Variables de sesión.** Sirven para que un usuario que ha iniciado sesión no tenga que volver a hacerlo dentro de un tiempo razonable. Tenemos que evitar que las cookies y variables de sesión permanezcan activas en un tiempo prolongado. Aunque no existe un criterio claro sobre la duración que debe tener una sesión,

se ha decidido que tenga una duración máxima de treinta minutos por sesión. En el caso que la aplicación manejase transacciones bancarias, habría que reducir el tiempo de esta a unos 15 minutos por sesión.

- **Password Sniffing.** El espionaje de contraseñas por medio de programas de terceros es posible en aquellas aplicaciones en las que los datos que se envían al servidor navegan por el protocolo HTTP y no por el protocolo HTTPS. La conjuración de nuestro servidor web se realizará de tal manera que soporte HTTPS para así poder cifrar el canal de comunicación por donde se enviaran los datos.

Todos estos aspectos se tendrán en cuenta para que la aplicación que se está desarrollando sea lo más segura posible.

7.3. Diseño de la Base de Datos

En esta sección se va a tratar el proceso realizado a la hora de diseñar la Base de Datos de la aplicación.

- Licitud, lealtad y transparencia. Los datos personales deben ser tratados de manera lícita, leal y transparrente. En nuestro modelo conceptual de datos, no tenemos datos personales como tarjetas de crédito, cuentas bancarias, etc. Es verdad que tenemos un campo identificativo para los usuarios que es el password, el cual hay que tener especial cuidado de no mostrar a usuarios que no tengan el consentimiento de visualizarlo.
- Limitación de la finalidad. Los datos que se recojan deben tener fines justificados. Cada uno de los campos que recogen cada una de las entidades del diagrama de la la figura [22] tienen si justificación, es decir, van a ser utilizados de manera lícita y leal para el cometido indicado en el capítulo de análisis.
- Datos limitados a lo necesario. En el modelo creado se recogen los datos mínimos para poder cumplir con los requisitos recogidos en el capítulo 6.2.
- Exactitud. Datos exactos y si fuera posible actualizados. En este caso, de la actualización y validación de los datos se ocupa el administrador.
- Integridad y confidencialidad. Los datos confidenciales que se guarden en la base de datos tienes que estar almacenados de forma segura. Datos tales como las claves de acceso de usuarios van a estar encriptados en la base de datos con una encriptación fuerte.

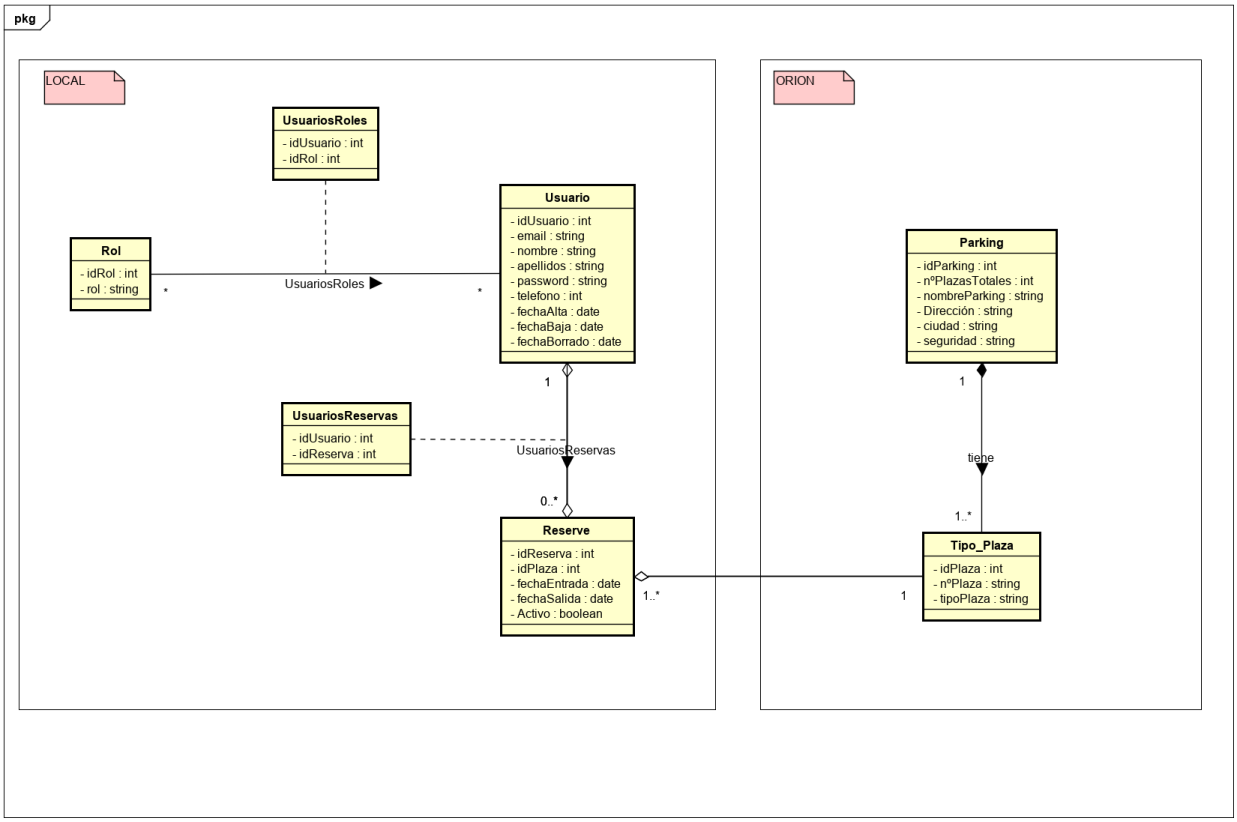

Figura 22: Modelo conceptual de los datos.

Como se puede ver en el modelo conceptual creado, los datos de los usuarios que van a acceder en la aplicación se guarda en el esquema con las entidades que forman la generalización-especializacion Usuario y Reserva. Se ha optado por almacenarlo en la misma base de datos donde se guardaran el resto de elementos como los roles. Siendo asi más facil el mantenimiento de ésta. Por otro lado, muchas de las entidades que se muestran en el modelo de la figura [22] tienen como referencia las clases del modelo de dominio inicial, aunque existan diferencias entre ambos modelos. Por ejemplo, aparecen las entidades de UsuariosRoles y UsuariosReservas.

A continuación se van a detallar las entidades que se considera que necesitan alguna aclaración adicional a lo ya visto en el análisis.

- Parking y tipo de plaza estaran guardados en ORION, hemos decidido hacerlo de esta manera porque nosotros vamos a trabajar con los datos más sensibles y que requieren seguridad. Como hemos explicado anteriormente. Orion va a ser un parking ajeno a nosotros y no vamos a poder manipular los datos.

Las entidades que aparecen nuevas con respecto al modelo de dominio son: UsuariosRoles y UsuariosReserva. Ambas relaciones que tienen un mucho a muchos en ambas direcciones de la relación. Concretamente, la identidad UsuariosRoles es la relación intermedia entre la entidad Rol y la entidad Usuario; y la entidad UsuariosRerservas es la relación intermedia entre la entidad de Usuario y la entidad Reserva. En el modelo relacional de la figura [22] se pueden observar diferentes tipos de relaciones. A continuación se explicará como van a resolver para pasar el modelo de entidad relación al modelo relacional.

Relación binaria de muchos a muchos (..* : ..*)

Para cada relación binaria de muchos a muchos crearemos una tabla intermedia también llamada tabla asociativa. En ésta añadiremos los atributos que formen parte de la relación y los atributos que formen la clave primera dee ambas clases que actuarán de claves foráneas y a su vez como clave principal. Un ejemplo del resultado de aplicar ésta relación en nuestro modelo conceptual es el que se muestra en la figura.

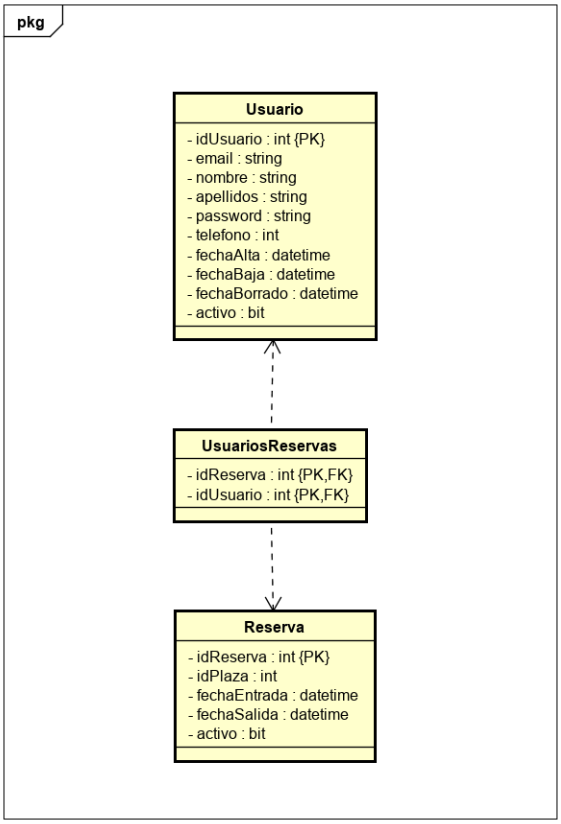

Figura 23: Relación binaria muchos a muchos (..* : ..*).

El resultado de realizar estas transformaciones es el siguiente modelo relacional de la figura [24]

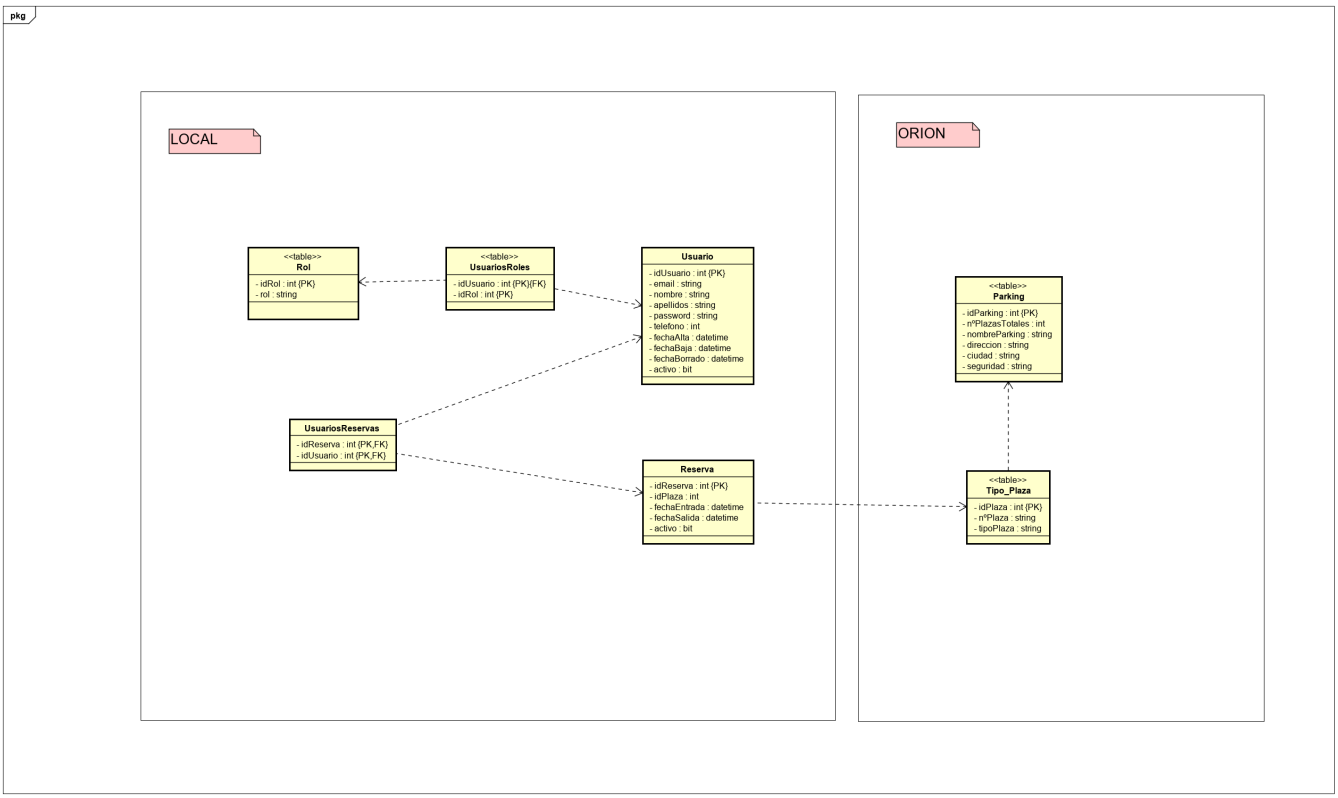

Figura 24: Modelo Relacional.

7.4. Patrón Modelo-Vista-Controlador

Es un patrón cuya función principal es la separación de la lógica de negocio de los datos. Tenemos tres componentes en este patrón, el modelo, el cual representa la información con la que el Sistema trata. El controlador el cual responde a los eventos que se producen en la página web. Este se comunica con la capa de vistas y la capa de modelo. Por último, el paquete con las vistas cuya función es presentar al modelo(información) en el formato correspondiente, tablas, formularios etc.

7.5. Diagramas de secuencia

En este apartado se muestran los diagramas más relevantes para comprender el funcionamiento del diseño planteado. El resto de diagramas tendrían una estructura similar y su comportamiento sería muy parecido a los propuestos.

- **Diagrama de secuencia : Registro**

  En este diagrama podemos ver como se registra un usuario en nuestro sistema. El controlador carga la vista que contiene el formulario de registro, tras introducir los datos el usuario, el controlador recoge los datos y los al método encargado de crear el usuario en la base de datos. Una vez creado el usuario, el controlador muestra la página de login al usuario.

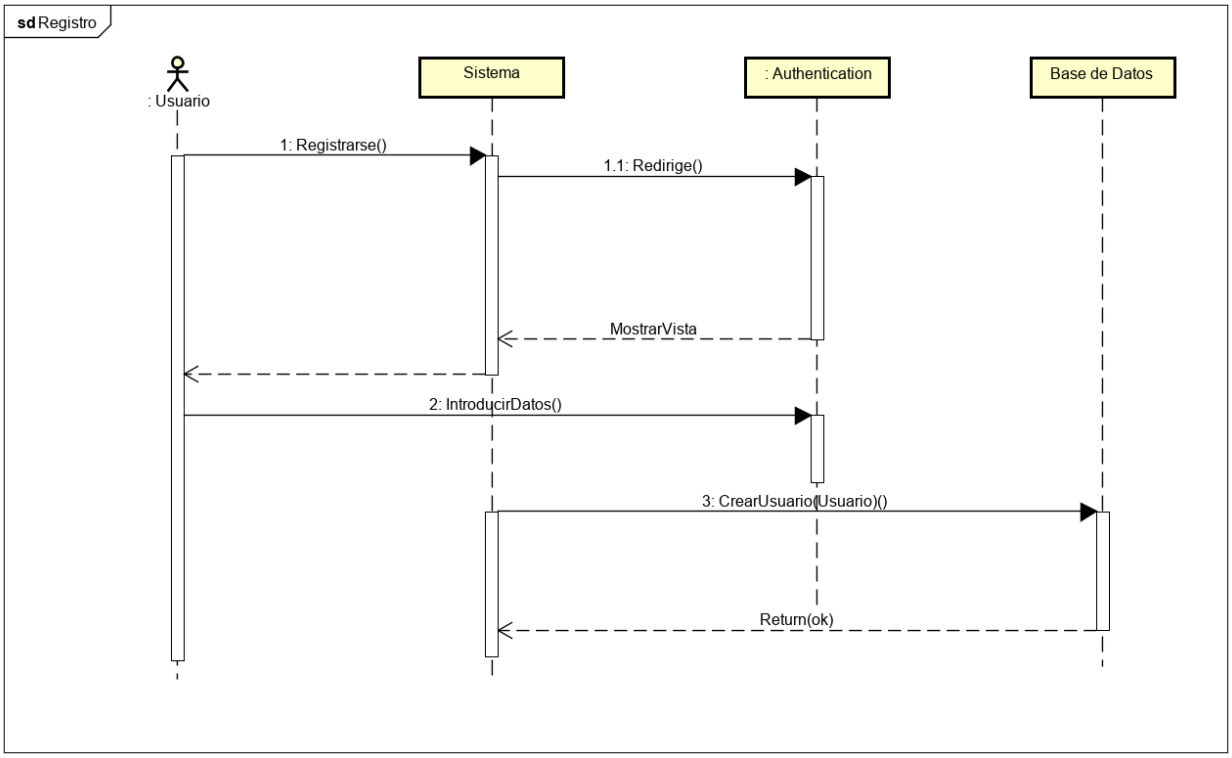

Figura 25: Diagrama de secuencia: Registro.

- **Diagrama de secuencia : Reserva**

  En este diagrama podemos observar como un usuario realiza una reserva. El usuario selecciona la plaza deseada. El sistema manda al controlador la fecha para comprobar las plazas disponibles. El sistema muestra la vista con la plaza seleccionada y el material disponible. El usuario selecciona la plaza y confirma la reserva. El sistema recoge los datos de la vista y se los pasa al controlador para que se encargue de crear la reserva en la base de datos.

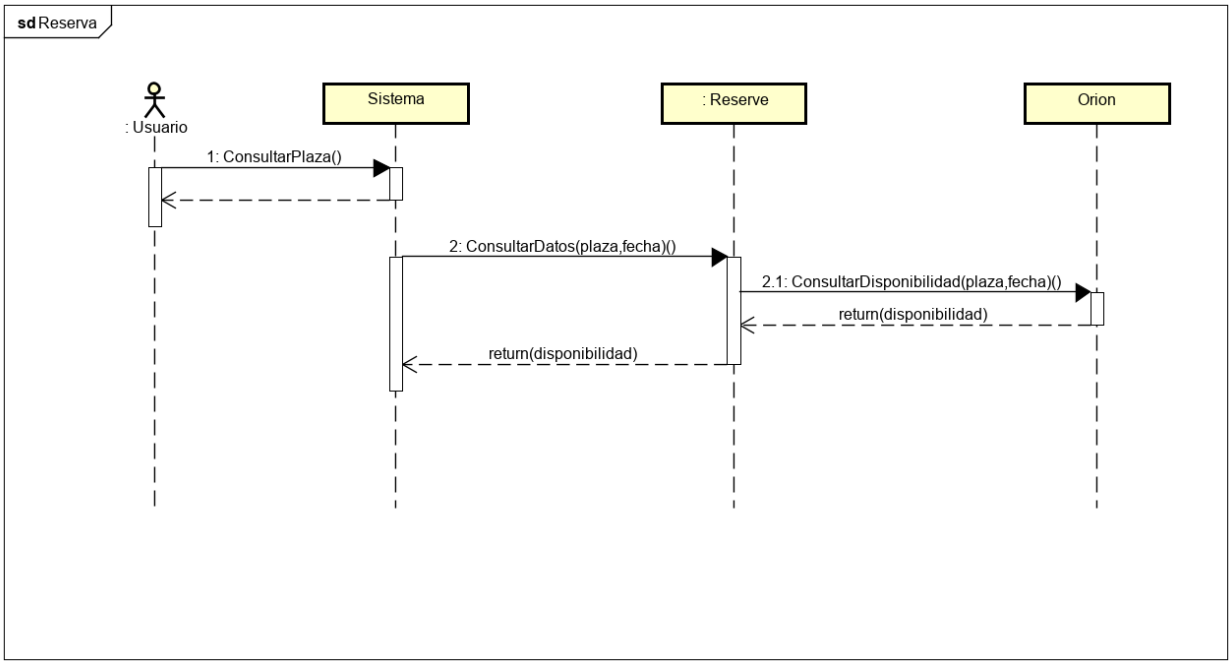

Figura 26: Diagrama de secuencia: Reserva.

8. Desarrollo del prototipo

8.1. Propuesta de prototipo

Como se ha mencionado en la propuesta de solución, se busca adaptar la aplicación web de este proyecto con una interfaz lo más sencilla posible con una navegación intuitiva y que implemente los casos de uso analizados para lograr así un grado alto de usabilidad del producto.

- Eficacia
- Efectividad
- Satisfacción en un contexto de uso especificado.

8.2. Diseño de los subsistemas

8.2.1. Vista de casos de uso. Diseño de la interfaz de usuario.

Los bocetos de la interfaz de usuario han sido realizados con la herramienta *"mockflow"*. Para el desarrollo de los bocetos se han utilizado diseños propios. Una vez implementada la aplicación, se han incluido iconos totalmente personalizados siguiendo el estilo en la medida de lo posible a los bocetos.

8.2.2. Especificación de casos de uso de diseño

En este apartado se especifican los casos de uso que respecta a la interfaz de usuario en la aplicación web, persiguiendo una interfaz sencilla , intuitiva y fácil de aprender.

1. CU-01 Registro en la aplicación

   En este apartado se muestra el registro del usuario en nuestra aplicación. El usuario tendra que rellenar un formulario de registro con una serie de campos, los cuales tendran una serie de restricciones para que el usuario los rellene correctamente.

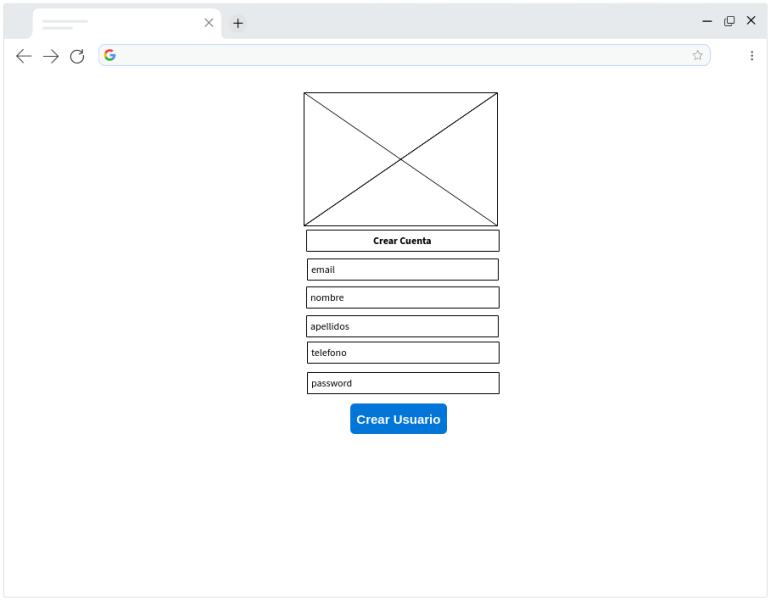

Figura 27: Registro en la app.

2. CU-02 Iniciar Sesión

   Una vez registrado, en la parte central, el usuario tendrá que rellenar el email y contraseña para iniciar sesión.

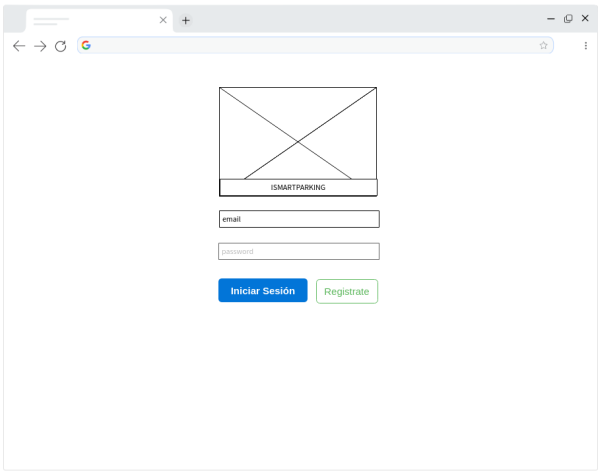

Figura 28: Iniciar Sesión.

3. CU-03 Consultar datos personales

   Dentro de la aplicación el usuario podra utilizar un menu intuitivo para navegar por la aplicación, a la derecha del todo tendremos el apartado (mis datos personales) en el cual, el usuario podra ver en una tabla sus datos personales, y podrá modificarlos.

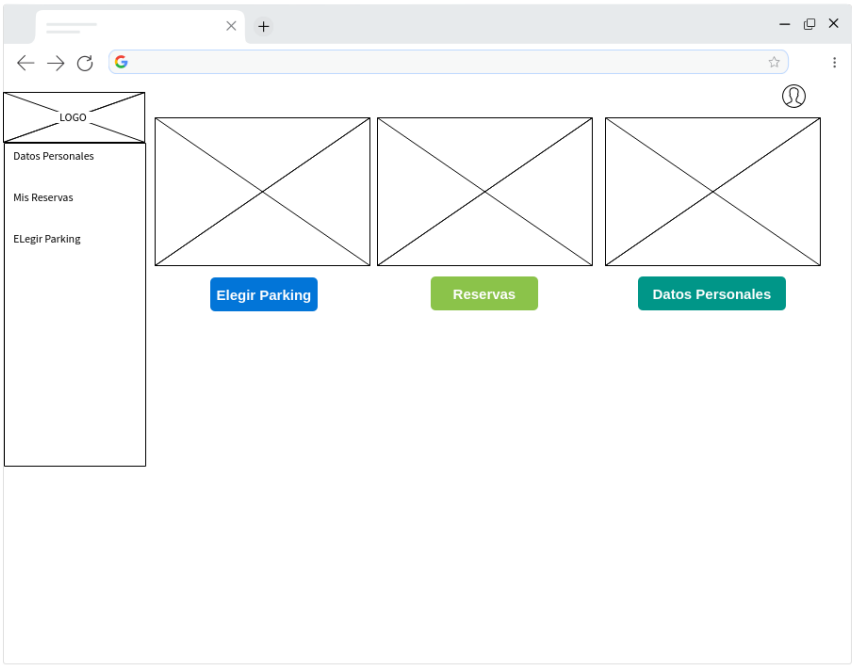

Figura 29: Página principal.

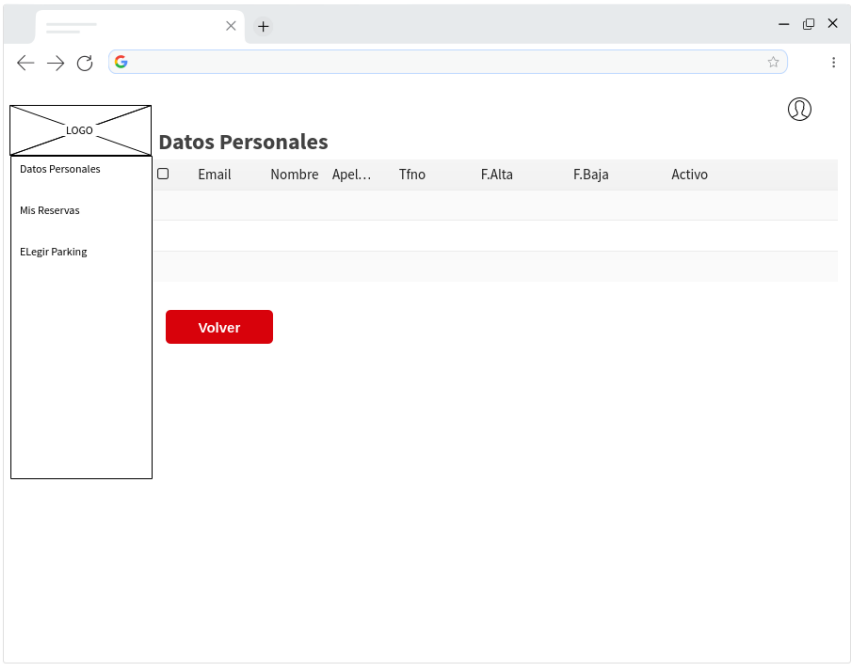

Figura 30: Datos Personales

4. CU-04 Consultar plazas disponibles

   EN el menu principal, consultaremos las plazas disponibles, pero primero escogeremos el parking que queremos y luego veremos las plazas disponibles que podremos reservar, con sus distintos tipos de plazas.

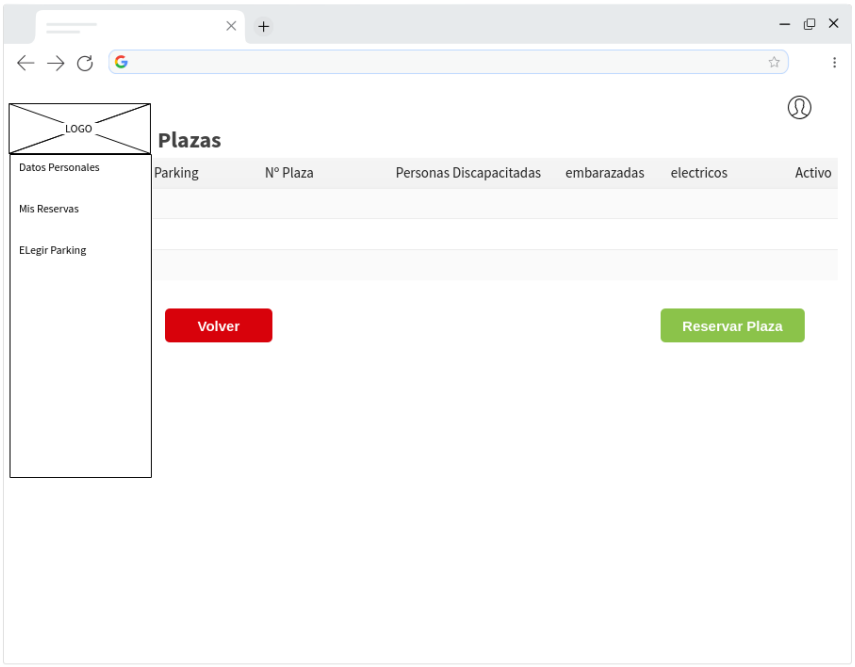

Figura 31: Consultar plazas disponibles.

5. CU-05 Realizar Reserva

   Seleccionaremos la plaza que queremos y realizaremos la reserva, el usuario tendra que completar un formulario, en el cual tendra que rellenar el parking, la plaza y fechas de entrada y salida.

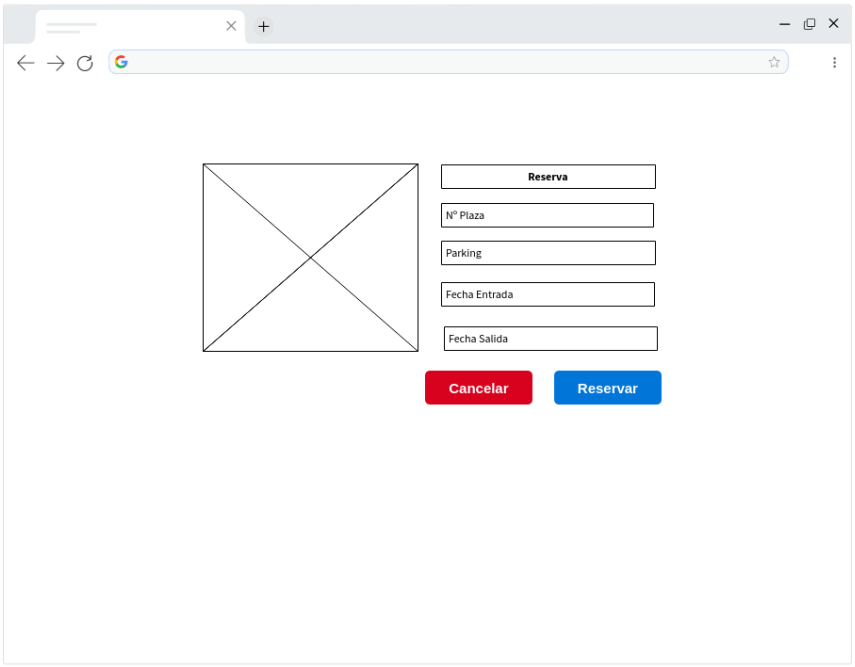

Figura 32: Realizar Reserva.

6. CU-06 Visualizar Reservas

   En el menu el usuario podra consultar sus reservas en todo momento, y modificarlas o cancelarlas segun crea conveniente.

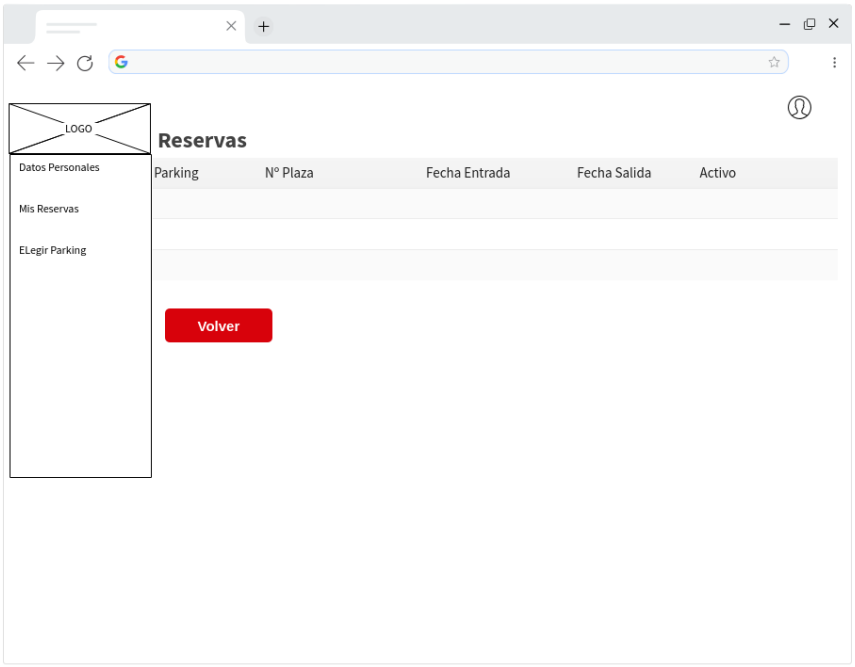

Figura 33: Consultar Mis reservas.

7. CU-07 Cerrar Sesión

   En el menu de la parte superior el usuario se podra deslogar y al aceptar lo llevara a la pantalla de login.

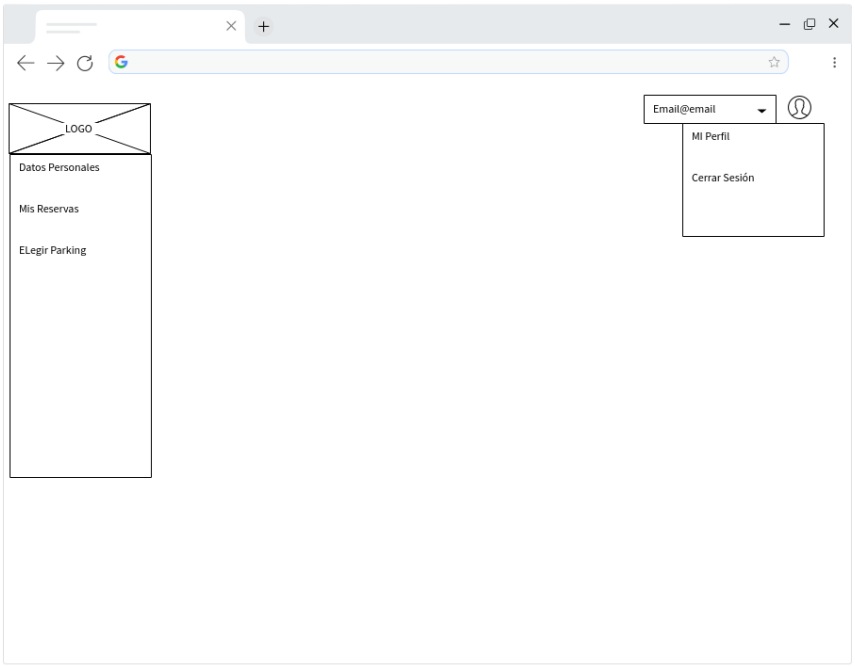

Figura 34: Cerrar Sesión.

9. Herramientas y Tecnologías utilizadas

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 su implementación.

- **Consumir API web en ASP.NET MVC**

  Para consumir la API web en el servidor ASP.NET MVC, podemos utilizar HttpClient en el controlador MVC. HttpClient envía una solicitud a la API web y recibe una respuesta. Luego necesitamos convertir los datos de respuesta que provienen de la API web a un modelo y luego convertirlos en una vista.

  La siguiente figura ilustra el consumo de la API web en ASP.NET MVC.

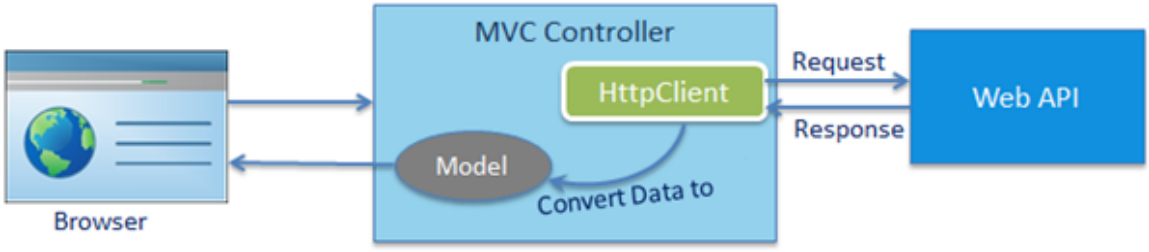

Figura 35: Consumir API web en el lado del servidor ASP.NET MVC

  AngularJS o cualquier otro marco de JavaScript se puede utilizar en la vista MVC y puede acceder a la API web directamente desde la vista utilizando AJAX. Hemos tomado ASP.NET MVC solo para demostrar cómo acceder a la API web desde el código del lado del servidor en caso de que no use ningún marco de JavaScript.

- **Arquitectura cliente-servidor:**

  Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios obtener acceso a la información de forma transparente, aun en entornos multiplataforma. Es la arquitectura tradicional que se aplica a las aplicaciones web

  Dicha arquitectura la forman el cliente y el servidor:

  - **Cliente**

    Es el proceso que permite al usuario formular los requisitos y pasarlos al servidor, se le conoce con el término front-end. Normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI).

  - **Servidor**

    Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso del servidor se le conoce con el término *backend*.

    El servidor maneja normalmente las funciones relacionadas con las reglas de negocio y los recursos de datos.

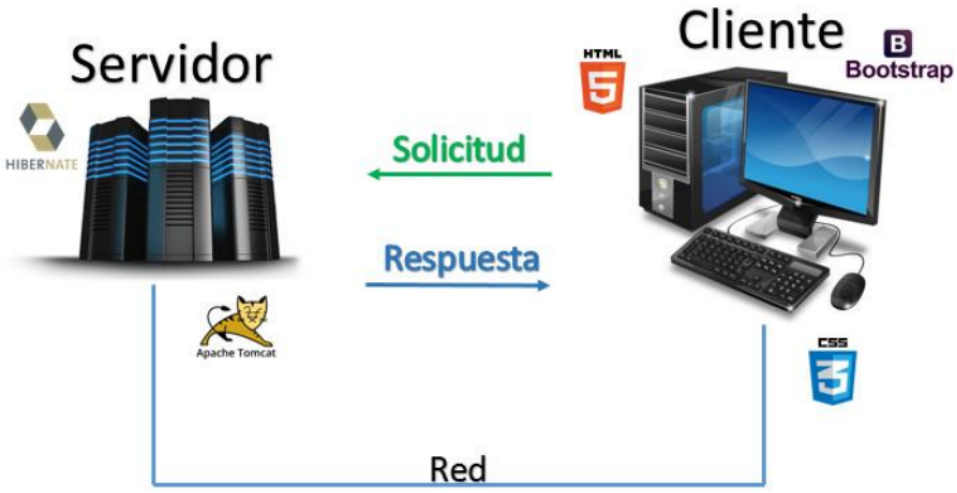

Figura 36: Arquitectura cliente-servidor

Como podemos ver en la imagen anterior, en el modelo cliente-servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio).

- **JSON**

  JSON (JavaScript Object Notation) es un formato de intercambio de datos. Consiste en un formatofácil de leer y escribir por los humanos, así como de analizar y generar por las máquinas.

  JSON se puede definir como un subconjunto del lenguaje de programación JavaScript. Aunque se trata de un formato de texto completamente independiente, utiliza nociones familiares para los programadores de lenguajes de la familia C, incluyendo C, C++ y C#, Java, JavaScript, Perl, Python y muchos otros. Estas propiedades convierten a JSON en un candidato ideal para compartir datos entre lenguajes.

  JSON está basado en dos estructuras:

  - Una colección de pares nombre/valor. En varios lenguajes, esto se realiza mediante un objeto, estructura, diccionario o array asociativo.
  - Una lista ordenada de valores. En la mayoría de lenguajes, esto se realiza mediante un array, vector, lista o secuencia.

  Estas estructuras son universales ya que virtualmente todos los lenguajes de programación modernos las soportan de una u otra manera. En JSON, adquieren estas formas:

  - Un objeto es un par desordenado de nombre/valor. Un objeto comienza con { y termina con }. Cada nombre es seguido por: y el par nombre/valor es separado por coma.
  - Un array es una colección ordenada de valores. Un array comienza con [y termina con]. Los valores son separados por comas.
  - Un valor puede ser un string con dobles comillas, un número, verdadero falso o null, un objeto un array. Estas estructuras pueden ser anidadas.

- Un string es una secuencia de cero o más caracteres Unicode, envueltos en dobles comillas, usando escapes de barra invertida. Un carácter es representado como un string simple. Estas estructuras son muy parecidas a los strings de Java o C.
- Un número es similar a Java o C, con la salvedad de que no se utilizan los formatos octal y hexadecimal.

- **Docker**

  Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

  Docker Compose le permite vincular un contenedor de Orion Context Broker a un contenedor MongoDB en unos pocos minutos. Debe instalar Docker Compose para que este método funcione.

9.1. Microsoft Visual Studio

Entorno de desarrollo integrado o IDE para sistemas operativos Windows que permite ver y editar cualquier tipo de código entre los que cabe mencionar, C++, C#, Visual Basic .NET, F# ,Java, Python, Ruby y PhP. Este programa también tiene la capacidad de depurar, generar y publicar aplicaciones para Android, IOS, Windows, la Web y la nube. url: 1

Este entorno de desarrollo se ha utilizado para escribir todo el código que se ha necesitado para la realización de este proyecto. El código de este proyecto está compuesto de SQL, C#, HTML, CSS, JavaScript.

9.2. C#

Lenguaje de programación simple, orientado a objetos, de tipado seguro, de propósito general y con una fuerte herencia de C/C++. Usa la plataforma .NET de la que toma el modelo de objetos, similar al de Java y además incluye mejoras que derivan de otros lenguajes. La especificación de dicho lenguaje se produjo en el año 2000 y fue diseñado por Anders Hejlsberg (que también participó en el desarrollo de Turbo Pascal y Delphi), Scott Wiltamuth y Peter Golde. Este lenguaje fue desarrollado para soportar las bibliotecas de clase de la plataforma .NET que comentaremos más adelante y que originalmente estaban escritas en un sistema de código gestionado denominado Simple Managed C (SMC). url: 2

Este lenguaje se ha utilizado para desarrollar toda la funcionalidad del proyecto como puede ser el acceso y obtención de datos de la base de datos, gestionar las peticiones del usuario, etc. Así como para realizar algunas tareas en la capa de presentación de los datos como explicaremos más adelante.

9.3. SQL (Structured Query Language

Es un lenguaje específico del dominio que permite el acceso a un sistema de gestión de base de datos relacionales. SQl incluye operaciones para realizar inserción, modificación, eliminación. y consulta de datos como también la creación y modificación de esquemas y el control de acceso. Entre las características que tiene este lenguaje cabe destacar su orientación al manejo de conjunto de registros y el manejo del álgebra y del cálculo relacional que permiten de manera sencilla lograr una mayor productividad. url: 3

9.4. Microsoft SQL Server

Sistema de gestión de bases de datos relacionales desarrollada por Microsoft, se ejecuta con el lenguaje de desarrollo Transact-SQL (TSQL) que es una implementación del estándar ANSI del lenguaje SQL. Este lenguaje nos permite realizar las operaciones generales sobre los datos como son obtener, modificar, eliminar e insertar; crear tablas, así como las relaciones que se producen entre ellas y también la administración del servidor. url: 4

Este producto tiene distintas ediciones y servicios como son: Enterprise, Developer, Standard, Express y SQL Azure:

- Enterprise: Esta versión contiene todas las funcionalidades.
- Versión que contiene las mismas funcionalidades que la versión anterior a diferencia de que esta está pensada para usarla en entornos de desarrollo.
- Standard: Versión que tiene limitada las características del servidor ya que está pensada para instalarla en servidores inferiores.
- Express: Versión gratuita que permite la creación de bases de datos con capacidades limitadas para soluciones simples que no requieren muchos recursos.
- SQL Azure: Versión de SQL Server que se encuentra en la nube por la que pagas mensualidad solo por el uso que haces. Dicho almacenamiento se encuentra en servidores que están localizados en distintos puntos del mundo.

9.5. .NET

Es un Framework desarrollado por Microsft que se caracteriza por su diseño centrado en la transparencia de las redes, la capacidad de crear una solución o una aplicación independiente de la plataforma y el hardware, además de la incorporación de librerías que posibilitan un rápido de desarrollo de las aplicaciones. url: 5

Los tres componentes principales de esta tecnología son:

9.5.1. El conjunto de lenguajes de programación

Gracias a la norma para la infraestructura común de lenguajes o CLI, Microsoft crea un marco de trabajo en .NET que facilita el desarrollo de todas las aplicaciones que soporta la plataforma en más de 20 lenguajes de programación distintos eliminando las diferencias existentes entre las cosas que se pueden hacer con cada uno de ellos.

Entre los lenguajes de programación que podemos encontrar en el marco de trabajo en .NET están: C#, Visual Basic .NET, Delphi (Object Pascal), C++, F#, J#, Perl, Python, Fortran, Prolog(P# y Prolog.NET), Cobol y PowerBuilder.

9.5.2. La biblioteca de las clases base o BCL.

Es la biblioteca donde se encuentran las bibliotecas de clase, interfaces y tipos de valor que son la base del desarrollo de aplicaciones y ofrecen acceso a toda la funcionalidad del sistema.

La biblioteca de clases base están organizadas en cuatro grupos como se indica a continuación:

- ASP:NET y Servicios Web XML
- Windows Forms
- ADO.NET
- .NET

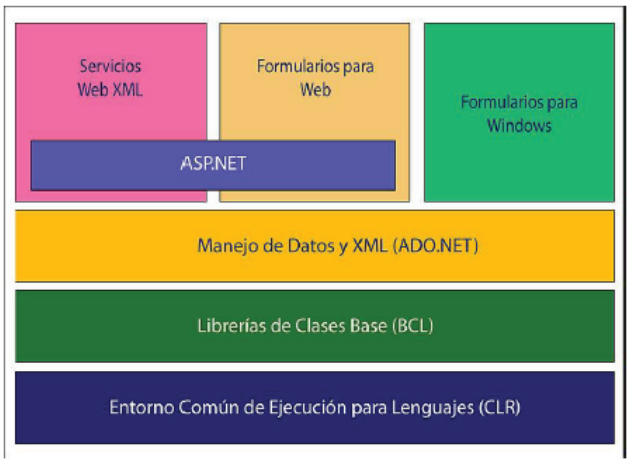

Figura 37: Estructura de clases FRamework .NET

Dentro de cada uno de estos grupos podemos encontrar las siguientes operaciones básicas:

- Interación con los dispositivos periféricos.
- Manejo de datos (ADO.NET)

- Administración de memoria
- Cifrado de datos
- Transmisión y recepción de datos por distintos medios (XML, TCP/IP)
- Administración de componentes Web que corren tanto en el servidor como en el cliente (ASP.NET)
- Manejo y administración de excepciones
- Manejo del sistema de ventanas
- Herramienta de despliegue de gráficos (GDI+)
- Herramienta de seguridad e integración con la seguridad del sistema operativo
- Manejo de tipo de datos unificado
- Interacción con otras aplicaciones
- Manejo de cadenas de caracteres y expresiones regulares
- Operaciones aritméticas
- Manipulación de fechas, zona horarias y periodos de tiempo
- Manejo de arreglos de datos y colecciones
- Manipulación de archivos de imágenes
- Aleatoriedad
- Generación de código
- Manejo de idiomas
- Auto descripción de código
- Interacción con el API Win32 o Windows API
- Compilación de código

9.5.3. El entorno común de ejecución para lenguajes, o CLR(Common Language Runtime)

Es la pieza fundamental del Framework .NET ya que es el programa que posibilita el desarrollo de las aplicaciones en diferentes lenguajes. Para lograr esto, se realiza un proceso que consta de dos fases: La primera fase se produce en tiempo de compilación y consiste en la conversión del código fuente de uno de los lenguajes que son soportados por .NET a un tipo de lenguaje intermedio denominado CIL (Common Intermediate Language). La implementación de este lenguaje intermedio está basada en el estándar Common Language Infrastucture (CLI).

En la siguiente fase se realiza la compilación del lenguaje intermedio (CIL) en el código máquina nativo para el sistema operativo, mediante el compilador del CLR. Entre las ventajas que existe en el código generado en este proceso están la integración entre lenguajes, control de excepciones entre lenguajes, seguridad mejorada, compatibilidad con la implementación y las versiones, un modelo simplificado de interacción y servicios de generación de perfiles y depuración.

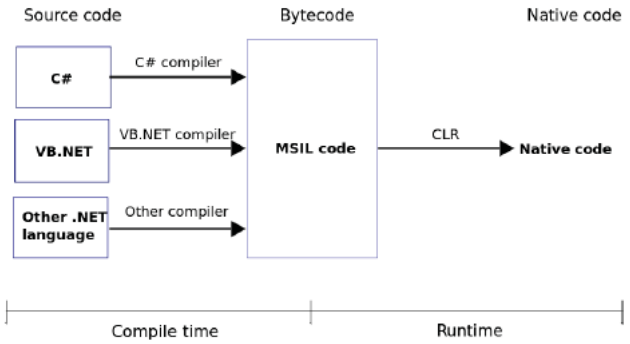

Figura 38: Proceso de compilación .NET

9.6. LINQ (Language Integrated Query)

LINQ es un componente del Framework .NET que fue lanzado el 19 de noviembre de 2007 en la versión 3.5 de dicha plataforma. Este componente ofrece una manera simplificada de realizar la extracción y procesamiento de datos desde diferentes fuentes como pueden ser arrays, clases enumerarles, documentos XML, base de datos relacionales. De esta forma se evita la necesidad de conocer los lenguajes que están asociados a cada uno de los tipos fuentes de datos como pueden ser el caso de SQL con bases de datos relacionales y XQuery para documentos XML.

LINQ este compuesto por un conjunto de métodos denominados operadores de consulta estándar, expresiones lambda y tipos anónimos.

9.7. ASP.NET

ASP.NET es la plataforma web que pone a disposición los servicions de .NET para desarrollar aplicaciones web, sitios web dinámicos y servicios web XML. Esta plataforma surgió en enero de 2002 con la versioón 1.0 de Framework .NEt y fue sucesora de la tecnología Active Server Pages (ASP). Con esta tecnología se logra la creación de páginas activas mediante la inserción de scripts y componentes junto al HTML para generar páginas dinámicamente.

ASP.NET incluye varias extensiones como son: ASP.NET AJAX, ASP.NET MVC Framework, ASP.NET Dynamic Data, ASP.NET web API, ASP.NET SignalR. En este proyecto se ha hecho uso de la extensión ASP.NET MVC Framework para crear la aplicación web usando la arquitectura MVC.

9.8. ADO.NET Entity Framework

Para entender correctamente este Framework es necesario comenzar explicando concepto de ORM.

Un ORM u Object-Relational Mapper es una biblioteca que añade una capa de abstracción en el proceso de acceso a los datos que se produce entre la base de datos y una aplicación que esté desarrollada en un lenguaje orientado a objetos.

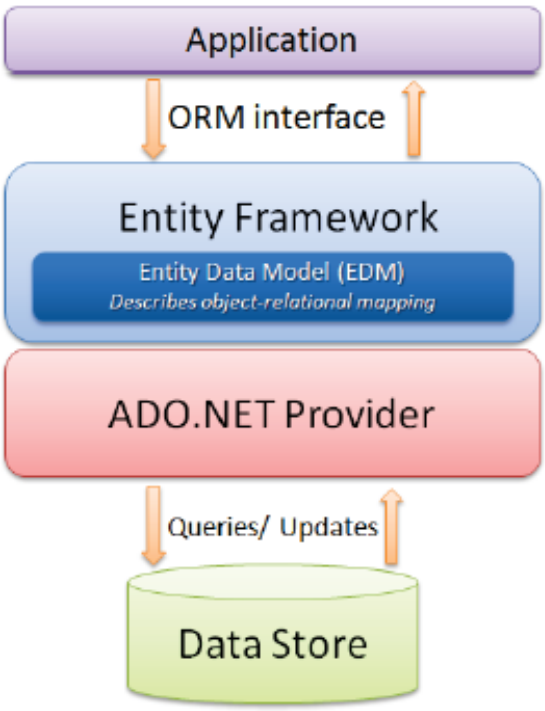

Figura 39: Estructura ORM

El problema que existía en este tipo de casos en el que tenemos que acceder a datos, usando un lenguaje orientado a objetos es que teníamos que crear un objeto especial para realizar la conexión a la base de datos, las consultas SQL que se realizaban se enviaban como cadenas de texto, la representación de los datos es distinta en cada uno de los sistemas ya que en una base de datos tenemos entidades que tienen atributos, éstas entidades se relacionan con otras entidades mediante el uso de claves primarias y foráneas mientras que en un lenguaje orientado a objetos creamos objetos con propiedades que contengan toda la información que necesitamos sin saber cómo relaciona o de dónde proviene esa información. Además, los tipos de datos suelen ser distintos ya sea por su nombre o su representación.

ORM permite crear aplicaciones desarrolladas en un lenguaje orientado a objetos definiendo clases y las propiedades que las componen y que pueden referenciar a otras clases de tal forma que no exista tanta dependencia a cómo está estructurada la base de datos, pudiéndonos centrar en el desarrollo de la aplicación.

Entity Framework es el ORM desarrollado por Microsoft desde sus inicios en agosto de 2008 formaba parte de .NET Framework, aunque en su última versión, Entity Framework 6, se ha separado.

Este framework cuenta con tres modos de trabajo:

9.8.1. Database First

En este modo de trabajo contamos con una base de datos diseñada y que contiene información, por lo tanto, Entity Framework nos aporta una capa para gestionar la conexión y para crear los objetos asociados a las entidades que existen en la base de datos.

9.8.2. Model First

En este caso, Entity Framework te permite diseñar la base de datos de manera visual usando el Diseñador de Modelos de Visual Studio por lo que no necesita escribir nada de código.

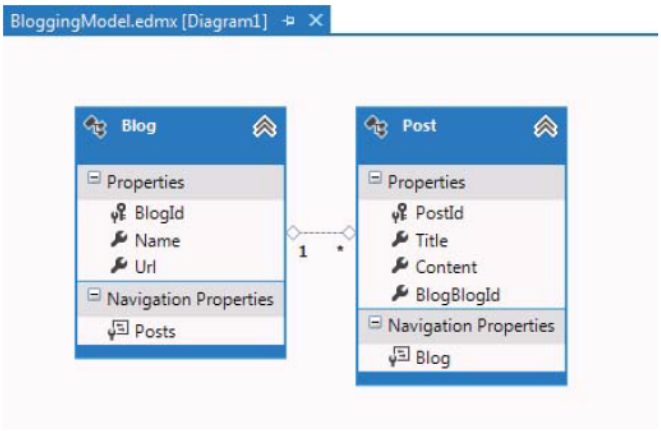

Figura 40: Diagrama de entidades con Model First

9.8.3. Code First

Como se puede intuir, este modo de trabajo parte de las clases y propiedades que se generan en el lenguaje orientado a objetos para crear la base de datos, además de todos los elementos necesarios para el correcto funcionamiento de ésta. Para conseguir las relaciones que existen entre entidades de la base de datos el programador hace uso de sus propiedades que se incluyen en las clases que quieren ser referenciadas. También permiten el uso de anotaciones, denominadas Annotations, que se sitúan encima de las propiedades para añadir información adicional como puede ser el nombre que la propiedad reciba en la base de datos el orden de las columnas de las tablas...

```
[Table("Pasaporte")]
public class Passport
{
    [Key]
    [Column(Order = 1)]
    public int PassportNumber { get; set; }
    [Key]
    [Column(Order = 2)]
    public string IssuingCountry { get; set; }
    public DateTime Issued { get; set; }
    public DateTime Expires { get; set; }
}
```

Figura 41: Ejemplo Annotations Entity Framework

9.9. HTML

Lenguaje de marcado que se usa para la elaboración de páginas web permitiendo la creación de contenido como puede ser texto, imágenes, video, tablas mediante una estructura básica. En 1991, Tim Berners-Lee publicó un libro HTML Tags donde se explicaban el diseño original de HTML, así como las etiquetas que lograban este lenguaje fuera tan simple. Actualmente HTML se ha convertido en el estándar que usan los navegadores para construir las páginas web, dicha estandarización está a cargo de la W3C.

9.10. CSS

Lenguaje que se usa para establecer el diseño visual y la presentación de un documento que haya sido escrito con algún tipo de lenguaje de marcado, por ejemplo, HTML. Como su nombre indica, hoja de estilos en cascada (Cascading Stylesheets), su estructura está basada en prioridades de forma que puedan asignar estilos a etiquetas concretas obteniendo resultados predecibles. Esta tecnología también cuenta con un estándar de la W3C que fue publicado en marzo de 1993.

9.11. Javascript

Lenguaje de programación interpretado, orientado a objetos, débilmente tipado y dinámico. Fue adoptado como estándar por la ECMA en junio de 1997 con el nombre de ECMAScript aunque fue desarrollado por Brendan Eich en 1995. SU uso está centrado en el lado del cliente, aunque también existen formas de Javascript que se usa en el lado del servidor. Este lenguaje ha vuelto a tener auge gracias a la llegada de Ajax provocando la generación de todo tipo de framework y librerías con ese lenguaje.

9.12. Ajax

Técnica de desarrollo web que nos permite crear aplicaciones interactivas mediante el uso de programas escritos en JavaScript.

Su funcionamoento se basa en realizar peticiones asícronas desde el lado cliente, es decir el navegador, hacia el servidor. La respuesta que envía el servidor sería almacenada en una variable de JavaScript y utilizada por el navegador según se desee. En la siguiente imagen se muestra las diferencias que existen entre las diistintas maneras de comunicación que hay en aplicaciones web:

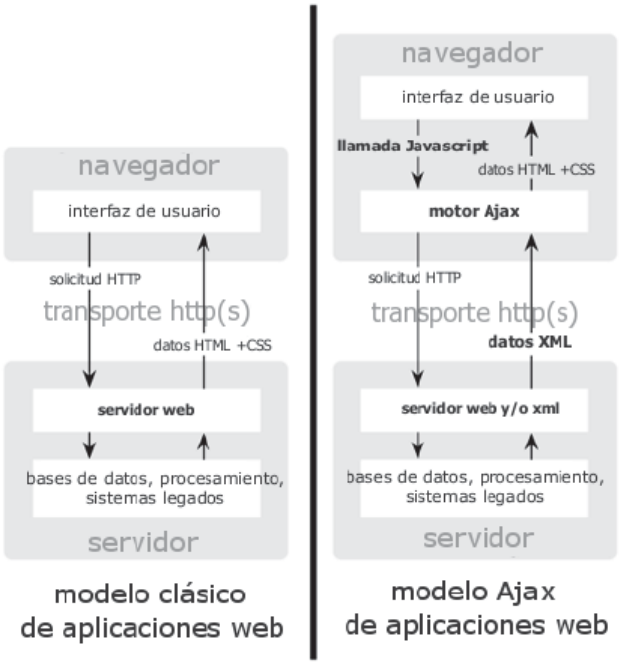

Figura 42: Comparación de tipos comunicación AJAX

9.13. JQuery

Es una biblioteca de código abierto y multiplataforma de JavaScript que permite agregar interactividad a una aplicación web. Es compatible con múltiples navegadores, facilita la inserción de animaciones, el manejo de eventos y la integración con AJAX.

9.14. Bootstrap

Es un Framework compuesto por código CSS y JavaScript que permite realizar diseños simples y responsive. Para ello, cuenta con un conjunto de plantillas de diseño que incluyen el código HTML, CSS y JavaScript con el que pueden crear componentes web que pueden ser tablas, botones, menus, formularios, barras de navegación, modales...

Fue desarrollado por Mark Otto y Hacob Thornton de Twitter y tuvo su origen en las necesidades internas de crear una herramienta que permitiera el un desarrollo más consciente y rápido.

9.15. Insomnia REST

La aplicación de Insomnia es bastante útil para probar APIs REST.

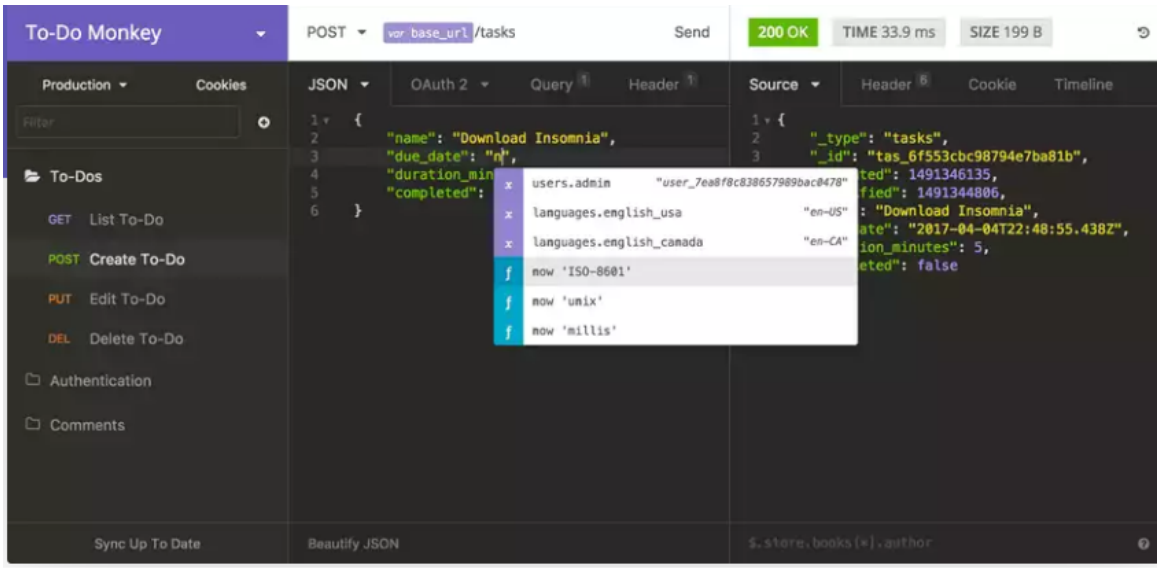

Figura 43: Insomnia REST

Podemos ver tres 3 columnas. Empezando por la izquierda, en la primera columna tenemos una lista con las pruebas que vamos creando. Las pruebas se pueden ir agrupando en carpetas. En la segunda columna podemos especificar el tipo de petición, la URL, los parámetros, el cuerpo, autenticación, etc. y en la tercera columna los resultados que devuelve el servidor tras hacerle esa petición.

9.16. FIWARE

FIWARE es una plataforma para el desarrollo de aplicaciones, sobre todo orientadas al Internet of Things (IoT) y las *Smartcities*. 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** que 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 .NET se encarga fácilmente de este asunto, disminuyendo drásticamente la complejidad del prototipo.

10. Evaluacion de costes

10.1. Costes hardware

Para el desarrollo de la aplicación es necesario un PC sin requisitos mínimos, pero que debe ser suficiente para soportar un procesamiento adecuado de servidor y el motor Visual Studio. Se utilizó un Dell G3 15, adquirido por un valor de 700 €. La vida media estimada para este tipo de dispositivos es de uno 4 años, es decir, 48 meses. El proyecto duro 11 meses, y durante toda su duración se utilizó el portátil.

Concepto	**Coste**
Dell G3 15	(700 € / 48 meses) * 11 meses = 160,41€
TOTAL	**160,42€**

Tabla 14: Costes hardware

10.2. Costes software

Todo el software adquirido para realizar el proyecto es gratuito, salvo Microsoft Visual Studio 2017, Microsoft Word y Astah Professional. El sistema operático estaba instalado de fábrica en el portátil.

Concepto	**Coste**
Microsoft Visual Studio 2017	489 € (Licencia Visual Studio 2017)
Microsoft Word 2016	109 € (Licencia Word 2016)
Astah Professional	299 € (Licencia individual)
TOTAL	**897€**

Tabla 15: Costes software

Finalmente se utilizó una licencia de la Escuela de Ingeniería Informática de la Universidad de Valladolid para adquirir estos 3 productos software, por lo que se obtuvieron gratuitamente, pero se mantienen estos conceptos en el presupuesto.

10.3. Costes de los recursos humanos

Finalmente, el proyecto duro 300 días, con 1,5 horas de trabajo por día. El sueldo medio de un analista/programador ronda los 12 € la hora.

Concepto	**Coste**
Sueldo de analista/programador	12 € / hora * 1,5 horas / día * 300 días = 5.400€
TOTAL	**5.400€**

Tabla 16: Costes recursos humanos

10.4. Costes de mantenimiento

El proyecto se desarrolló en instalaciones de la Universidad de Valladolid, por lo que los costes referidos al mantenimiento y consumo de recursos como la electricidad los consideramos nulos.

10.5. Costes totales

Se muestra el coste total del proyecto a partir de la suma de los costes anteriores.

Concepto	**Coste**
Coste hardware	160,42€
Coste software	897€
Coste recursos humanos	5.400€
Coste mantenimiento	0€
TOTAL	**6.457,42€**

Tabla 17: Costes totales

11. Pruebas

11.1. Pruebas unitarias

Estas pruebas han sido realizadas durante la codificación del sistema, de forma manual al depurar el código. En ellas se probaron los métodos diseñando casos de prueba siguiendo estrategias de caja negra. No se llegaron a implementar pruebas unitarias automáticas como tal.

11.2. Pruebas de caja negra

Las pruebas de caja negra consiste en validar las respuestas y las salidas que muestra el sistema al introducir unas determinadas entradas. Es decir, esta prueba se basa en comprobar que el sistema ha realizado lo que se ha deseado que haga y que produzca las salidas esperadas sin prestar atención en cómo lo ha conseguido internamente.

1. **PCN-01** Registrarse en la aplicación
    - **Objetivo:** Registrar un usuario en la aplicación
    - **Precondiciones:** Usuario que se va a registrar no exista.
    - **Datos de entrada:** Campos del formulario de registro
    - **Resultado esperado:**Pulsar en el botón Crear Usuario tras completar todos los campos del formulario y si el registro es correcto, redirecciona a la página de login
    - **Resultado obtenido:** Correcto
2. **PCN-02** Acceder a la aplicación
    - **Objetivo:** Usuario registrado previamente pueda acceder a su zona privada
    - **Precondiciones:** El usuario esté registrado en la aplicación sin fecha de baja.
    - **Datos de entrada:** Campos del formulario de acceso
    - **Resultado esperado:** Pulsar en el botón Enviar tras completar todos los campos del formulario y redirecciona a la página principal de la aplicación.
    - **Resultado obtenido:** Correcto
3. **PCN-03** Modificar datos
    - **Objetivo:** El usuario pueda modificar los datos con los que se registro en la aplicación.
    - **Precondiciones:** El usuario esté registrado en la aplicación sin fecha de baja.
    - **Datos de entrada:** Campos del formulario de acceso
    - **Resultado esperado:** Pulsar en el botón Guardar tras modificar los campos del formulario y que se muestre un mensaje de confirmación.
    - **Resultado obtenido:** Correcto

4. **PCN-04** Realizar reserva
    - **Objetivo:** El usuario pueda reservar una plaza.
    - **Precondiciones:** El usuario esté registrado en la aplicación sin fecha de baja.
    - **Datos de entrada:** Campos del formulario reserva
    - **Resultado esperado:** Pulsar en el botón Reservar tras completar todos los campos del formulario y confirmar el mensaje de reserva.
    - **Resultado obtenido:** Correcto
5. **PCN-05** Modificar reserva
    - **Objetivo:** El usuario pueda modificar los datos de sus reservas.
    - **Precondiciones:** El usuario esté registrado en la aplicación sin fecha de baja.
    - **Datos de entrada:** Campos del formulario reserva
    - **Resultado esperado:** Pulsar en el botón Guardar tras modificar los campos del formulario de editar reserva y que se muestre un mensaje de confirmación.
    - **Resultado obtenido:** Correcto
6. **PCN-06** Cancelar reserva
    - **Objetivo:** El usuario pueda cancelar sus reservas.
    - **Precondiciones:** El usuario haya realizado una reserva.
    - **Datos de entrada:** Quitar el check del formulario de modificar reserva.
    - **Resultado esperado:** Pulsar en el botón Editar tras modificar los campos del formulario de editar reserva y que se muestre un mensaje de confirmación.
    - **Resultado obtenido:** Correcto

12. Conclusiones y Trabajo Futuro:

Está claro que la entrada en vigor del RGPD supone cambios en muchos aspectos del desarrollo de aplicaciones de software. En nuestro caso, hemos analizado esta incidencia considerando un caso particular de posibles aplicaciones para *Smartcities*, la gestión de un sistema de aparcamiento con reserva anticipada de plazas que tenga en cuenta características específicas de los usuarios.

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

El desarrollo del proyecto nos ha servido para ver de primera mano las dificultades de adaptar un sistema en producción a nuevas características no contempladas durante su diseño, en este caso su adecuación a lo dispuesto en el RGPD. En especial el no poder modificar el formato de los datos guardados en Orion.

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

12.1. Objetivos alcanzados

- Se ha estudiado la estructura de FIWARE y se ha implementado el Generic Enabler llamado Orion [7], que nos permite gestionar información de contexto de una forma altamente descentralizada y a gran escala.

- Se han recogido algunos de los derechos de la RGPD mediante el desarrollo de la aplicación, en la cual, los usuarios pueden borrar total o parcialmente sus datos personales, tambien podran modificarlos si son incorrectos. También avisaremos a los usuarios que sus datos serán utilizados hasta cierto tiempo despues de que estos se den de baja en la aplicación y comunicaremos para que usaremos sus datos personales, antes de que ellos se registren en nuestra aplicación.

- Gracias el uso del framework .NET, la aplicación desarrollada ofrece unas garantías mínimas de seguridad y estabilidad. Por ejemplo, no permite acceder a distintas páginas de nuestra aplicación web por url, si no estas autenticado en la propia aplicación. Tiene encriptado los datos personales de nuestros usuarios.

- El primer objetivo del proyecto se ha logrado de forma satisfactoria, ya que se ha conseguido integrar una plataforma de gestión de contexto operativa, a partir del GE disponibles en el marco de FIWARE con la ayuda de Orion. En cambio, la implementación de los GE de seguridad es uno de los objetivos que no se ha podido implementar, tras varios intentos no podiamos garantizar una seguridad plena en nuestra aplicación, y el uso del framework .NET, como hemos dicho en el punto anterior, nos ha ayudado a ofrecer unas garantías mínimas de seguridad y escalabilidad.

12.2. Trabajo Futuro

En cuanto al trabajo futuro, se puede plantear la posibilidad de incorporar nuevas funcionalidades que harían de nuestra aplicación web una aplicación mucho más completa.

- Lograr la integración completa con Orion, permitiendo suscripciones y acceso a todos los métodos ofrecidos por este.
- Mejora en la interfaz de usuario (pantallas, iconos, botones...).
- Implementar una aplicación para móviles.
- Habilitar una pasarela de pago.
- Acabar de implementar la interfaz del mapa y que te muestre con más exactitud donde está el parking e incluso la plaza seleccionada.

Referencias

[1] A. S. Ahmadian y J. Jürjens. «Supporting Model-Based Privacy Analysis by Exploiting Privacy Level Agreements». En: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. Dic. de 2016, págs. 360-365. URL: https://ieeexplore.ieee.org/document/7830704.

[2] Álvaro Alonso (UPM). *FIWARE Open specification, Identity management*. Accedido el 5-05-2019. URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.IdentityManagement.

[3] Álvaro Alonso. *PEP Proxy - Wilma*. Accedido el 10-04-2019. URL: https://catalogue.fiware.org/enablers/pep-proxy-wilma.

[4] Álvaro (UPM) Alonso. *FIWARE Open specification, PEP Proxy*. Accedido el 10-04-2019. URL: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.PEPProxy.

[5] Ali Alqazzaz1 y col. «SecSPS: A Secure and Privacy-Preserving Framework for Smart Parking Systems». En: *Journal of Information Security* (), págs. 299-314. DOI: 10.4236/jis.2018.94020. URL: https://www.scirp.org/Journal/PaperInformation.aspx?PaperID=87800.

[6] Majed Alshammari y Andrew Simpson. «A UML Profile for Privacy-Aware Data Lifecycle Models». En: *Computer Security*. Ed. por Sokratis K. Katsikas y col. Springer International Publishing, 2018, págs. 189-209. ISBN: 978-3-319-72817-9.

[7] Roberto Bahillo. «Extensión de FIWARE para soporte de privacidad acorde a las nuevas reglas del RGPD relativas a la gestión del flujo de los datos personales». Tesis de mtría. Escuela de Ingeniería Informática, 2018. URL: http://uvadoc.uva.es/handle/10324/33072.

[8] Tânia Basso y col. «Towards a UML Profile for Privacy-Aware Applications». En: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing* (2015), págs. 371-378. URL: https://ieeexplore.ieee.org/document/7363095.

[9] Javier Luis Cánovas Izquierdo y Julián Salas. «A UML Profile for Privacy Enforcement». En: *Software Technologies: Applications and Foundations - STAF 2018,Collocated Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers, LNCS 11176*. 2018, págs. 609-616. URL: https://ieeexplore.ieee.org/document/7363095.

[10] Cyril Dangerville (THALES). *FIWARE Open specification, Authorization PDP*. Accedido el 10-04-2019. URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.AuthorizationPDP.

[11] Cyril Dangerville. *Authorization PDP - AuthZForce*. Accedido el 5-05-2019. URL: https://catalogue.fiware.org/enablers/authorization-pdp-authzforce.

[12] Michele Drgon, Gail Magnuson y John Sabo. *Privacy Management Reference Model and Methodology (PMRM) Version 1.0*. Accedido el 06-05-2019. 2016. URL: http://docs.oasis-open.org/pmrm/PMRM/v1.0/PMRM-v1.0.html.

[13] Torres de Fenicia. Accedido el 06-05-2019. URL: http://torresdefenicia.tripod.com/sitebuildercontent/sitebuilderfiles/MP%5C_2.06.2%5C_Formulario_Tabla%5C_Definicion%5C_Administracion%5C_riesgos.pdf.

[14] Torres de Fenicia. *Formulario Tabla de Definición y Administración de los Riesgos del Proyecto*. Accedido el 7-01-2019. URL: http://torresdefenicia.tripod.com/sitebuildercontent/sitebuilderfiles/MP_2.06.2_Formulario_Tabla_Definicion_Administracion_riesgos.pdf.

[15] FIWARE. *Arquitectura general de FIWARE*. Accedido el 10-04-2019. URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Architecture.

[16] FIWARE. *Firawe-Orion*. Accedido el 10-04-2019. URL: https://fiware-orion.readthedocs.io/en/master/index.html.

[17] FIWARE. *FIWARE About us*. Accedido el 10-04-2019. URL: https://www.fiware.org/about-us/.

[18] Cross-site request forgery. Accedido el 10-04-2019. URL: https://es.m.wikipedia.org/wiki/Cross-site_request_forgery.

[19] Juanjo Hierro. *FIWARE: Value proposition for Smart Cities*. Accedido el 11-04-2019. URL: https://es.slideshare.net/JuanjoHierro/fiware-a-standard-platform-for-smart-cities.

[20] LegalToday. Accedido el 10-04-2019. URL: http://www.legaltoday.com/blogs/transversal/blog-ilp-abogados/plazos-para-conservacion-y-cancelacion-de-datos-personales.

[21] *Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales*. Accedido el 11-04-2019. URL: https://www.boe.es/boe/dias/2018/12/06/pdfs/BOE-A-2018-16673.pdf.

[22] A. Martínez-Ballesté, P. A. Pérez-Martínez y A. Solanas. «The pursuit of citizens' privacy: a privacy-aware smart city is possible». En: *IEEE Communications Magazine* 51.6 (jun. de 2013), págs. 136-141. ISSN: 0163-6804. URL: https://ieeexplore.ieee.org/document/6525606.

[23] E. Mougiakou y M. Virvou. «Based on GDPR privacy in UML: Case of e-learning program». En: *2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*. 2017, págs. 1-8. URL: https://ieeexplore.ieee.org/document/8316456.

[24] Stephan Neuhaus (Zürcher Hochschule der Angewandten Wissenschaften). *FIWARE Open specification, Privacy*. Accedido el 10-04-2019. URL: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.Privacy.

[25] Stephan Neuhaus. *Privacy*. Accedido el 06-05-2019. URL: https://catalogue.fiware.org/enablers/privacy.

[26] Pablo Pérez-Martínez, Antoni Ballesté y Agusti Solanas. «Privacy in Smart Cities: A case study of smart public parking». En: *PECCS 2013 - Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems* (ene. de 2013), págs. 55-59.

[27] FIWARE Equipo Privacy. *FIWARE Academy, Privacy*. Accedido el 10-04-2019. URL: https://edu.fiware.org/course/view.php?id=139.

[28] Blog de Protección de Datos para empresas y autónomos. *Derechos ARCO ¿Qué son? Nuevos derechos del RGPD*. Accedido el 10-04-2019. URL: https://protecciondatos-lopd.com/empresas/derechos-arco-que-son/.

[29] Youyang Qu y col. «Chapter 6 - Privacy Preservation in Smart Cities». En: *Smart Cities Cybersecurity and Privacy*. Ed. por Danda B. Rawat y Kayhan Zrar Ghafoor. Elsevier, 2019, págs. 75-88. ISBN: 978-0-12-815032-0. URL: https://doi.org/10.1016/B978-0-12-815032-0.00006-8.

[30] *RGPD (UE 2016/679)*. Accedido el 11-04-2019. URL: https://www.boe.es/doue/2016/119/L00001-00088.pdf.

[31] 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*. Ene. de 2018, págs. 5797-5805. DOI: https://scholarspace.manoa.hawaii.edu/handle/10125/50615.

[32] Peter Salhofer, Julia Buschsbaum y Michael Janusch. «Evaluating the FIWARE Platform». En: *52st Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019*. Ene. de 2019, págs. 7382-7389. URL: http://hdl.handle.net/10125/60175.

[33] Joaquín Salvachúa y Álvaro Alonso. *Identity Management KeyRock*. Accedido el 5-05-2019. URL: https://catalogue.fiware.org/enablers/identity-management-keyrock.

[34] StatCounter Global Stats. Accedido el 10-04-2019. URL: http://gs.statcounter.com/browser-market-share#monthly-201805-201805-bar.

APÉNDICES

APÉNDICE A: Manual de Instalación

A.1. Introducción

En este apartado de la memoria se describirán los pasos a seguir para la instalación de la aplicación junto con la base de datos, además del software necesario para su correcto funcionamiento.

A.2. Despliegue de Orion Context Broker

Necesitamos un sistema UNIX, en este caso hemos utilizado Ubuntu 16.04.
Para desplegar el Orion, lo primero que debemos tener en cuenta es que la forma más sencilla de desplegarlo es utilizando Docker y Docker compose, el Orion incluye una implementación API REST del Api Estándar NGSI V2 desarrollada con el lenguaje de programación C++, además de utilizar el SGBD MongoDB, en los siguientes pasos te mostraré la forma más sencilla de desplegar una instancia del Orion Context Broker utilizando Docker y Docker compose:

- Docker: Consultar *"https://docs.docker.com/install/linux/docker-ce/ubuntu/"* para instalar docker.
- Docker-compose: Puedes consultar *"https://docs.docker.com/compose/install/"* en el cual encontrarás la documentación para instalar Docker compose.

A.2.1. Descargar el archivo docker-compose.yml

El archivo *docker-compose.yml* contiene dentro las instrucciones para que pueda ser ejecutado el Orion utilizando Docker Compose.

Para descargar este archivo podemos utilizar en la terminal el siguiente comando:

```
$ curl https://raw.githubusercontent.com/telefonicaid/fiware-
orion/master/docker/docker-compose.yml -o docker-compose.yml
```

Figura 44: docker-compose.

A continuación te describiré brevemente el contenido del archivo *docker-compose.yml:*

- Primero podemos observar en el archivo las etiquetas MongoDB y Orion, estos etiquetas nos sirven para nombrar los contenedores que se ejecutarán cuando utilicemos Docker compose.
- La etiqueta image define la imagen docker en la cual estará basada el contenedor.
- La etiqueta command ejecuta un comando dentro del contenedor.

- La etiqueta links sirve para definir alias adicionales mediante los cuales se puede acceder a un servicio desde otro servicio, en este caso para que el Orion pueda acceder al contenedor de mongo.
- Por último la etiqueta ports sirve para asignar los los puertos del host del lado izquierdo y del contenedor del lado derecho respectivamente.

A.2.2. Ejecutar el Orion

Dentro de la carpeta en la que se encuentra el archivo *docker-compose.yml* ejecutaremos en la terminal el siguiente comando:

```
$ docker-compose up -d
```

Figura 45: Ejecutar el Orion.

Se comenzarán a descargar las imágenes docker del Orion Context Broker y de MongoDB, Esto ocurrirá únicamente la primera vez que se ejecute este comando.

Ahora podemos utilizar el siguiente comando para verificar que los contenedores se encuentren ejecutándose:

```
$ docker ps
```

Figura 46: Comando para ver los contenedores que estan ejecutandose.

Ahora revisaremos que el Orion se encuentre ejecutándose ingresando en el navegador la siguiente url; http://0.0.0.0:1026/version y observaremos que la respuesta de Orion es un JSON en el cual sus atributos describen información acerca de la versión que se está ejecutando.

A.2.3. Creación de tu primera Entidad.

Como te mencioné antes el Orion es una implementación API Rest, por esto para utilizar los servicios del mismo debes utilizar peticiones HTTP, para crear una entidad se debe utilizar una petición POST y en el body de la petición debe incluir la entidad que queremos crear en formato JSON y esta entidad debe cumplir con el Api estándar NGSIv2 de Fiware que te describí anteriormente.

Para este ejemplo y como es común en los ejemplos que nos proporcionan Fiware crearemos una entidad de tipo Plaza, esta entidad contendrá los atributos de la plaza. serán ambos de tipo Text.

Como puedes observar al crear la entidad en el Orion, éste nos responde con un status 200 lo que quiere decir que la entidad se creó correctamente, además cabe resaltar que en la cabecera llamada location muestra la ruta que debemos utilizar para consultar la entidad en el Orion, es decir, que para consultar la entidad o realizar otras operaciones sobre la misma

tendríamos utilizar la dirección http://0.0.0.0:1026/v2/entities/.

A.2.4. Detener el Orion

Antes de terminar te mostraré cómo detener el Orion cuando ya no lo utilices, para esto utilizaremos el comando **docker stop.**

A.3. Instalación de SQL SERVER

Necesitamos instalar el entorno para la administracion de SQL Server.

SQL Server Management Studio (SSMS) es un entorno integrado para administrar cualquier infraestructura de SQL, desde SQL Server a Azure SQL Database. SSMS proporciona herramientas para configurar, supervisar y administrar instancias de SQL Server y bases de datos. Use SSMS para implementar, supervisar y actualizar los componentes de nivel de datos que usan las aplicaciones, además de compilar consultas y scripts. Use SSMS para consultar, diseñar y administrar bases de datos y almacenes de datos, estén donde estén, en el equipo local o en la nube.

Una vez instalado SQL server instalaremos el SQL Server Management Studio (SSMS). Ahora tenemos que crear una Base de Datos y ejecutar el script o restaurar el .bak, que es una copia de seguridad de la base de datos.

APÉNDICE B: Manual de Administrador

B.1. Introducción

Nuestra aaplicación ayuda a los usuarios a realizar la reserva de una plaza, comprobando su disponibilidad, en distintos puntos de cada ciudad.

B.1.1. Descripción del documento

Este documento busca detallar la forma de utilización de la aplicación para que el Administrador haga un uso correcto de ella.

B.2. Funcionalidades para el Administrador

B.2.1. Iniciar sesión

Para poder iniciar sesion en la plataforma web de la aplicación hay que acceder a la siguiente url: . A continuación, hay que introducir el email de usuario y password del administrador y mostrara la pantalla de inicio del sitio web. El email de usuario es “admin@admin.es” y el password “admin”.

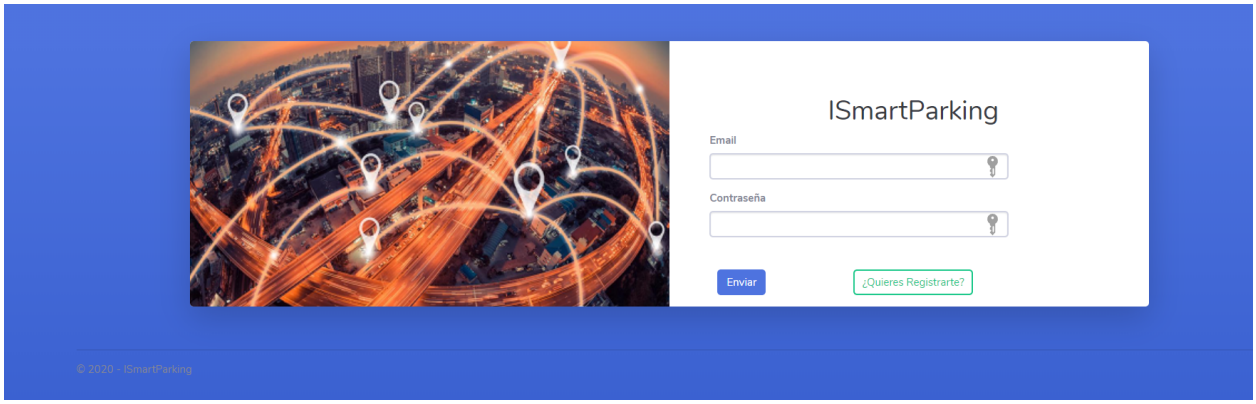

Figura 47: Iniciar Sesión.

B.2.2. Consultar usuarios registrados

El siguiente paso es consultar los usuarios registrados en la aplicación. En la ventana principal damos sobre el boton de Usuarios Registrados y nos aparecerá una tabla con los datos personales de todos los usuarios, incluida fecha de Alta en la aplicación.

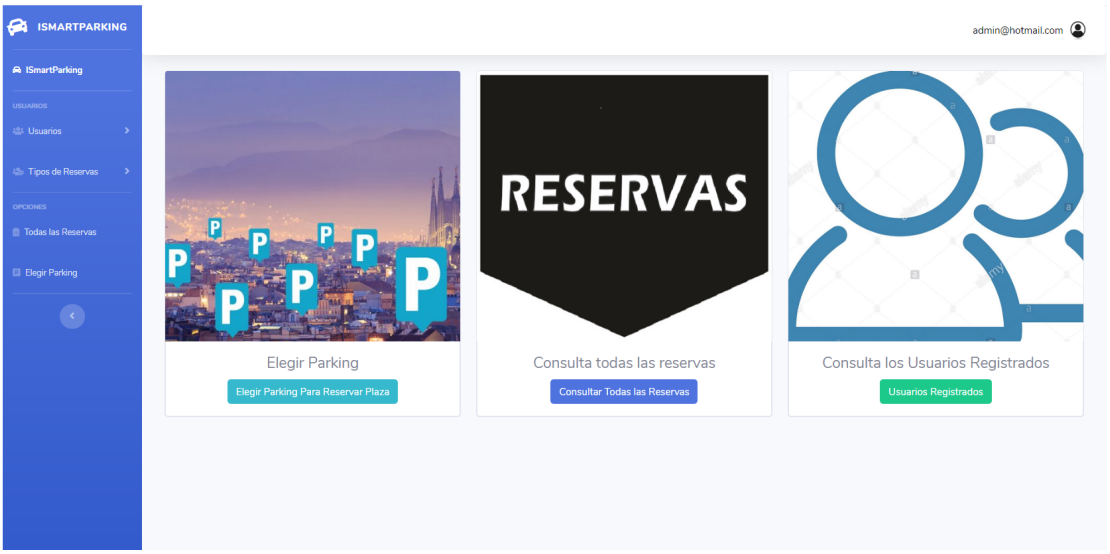

Figura 48: Menu del Admin

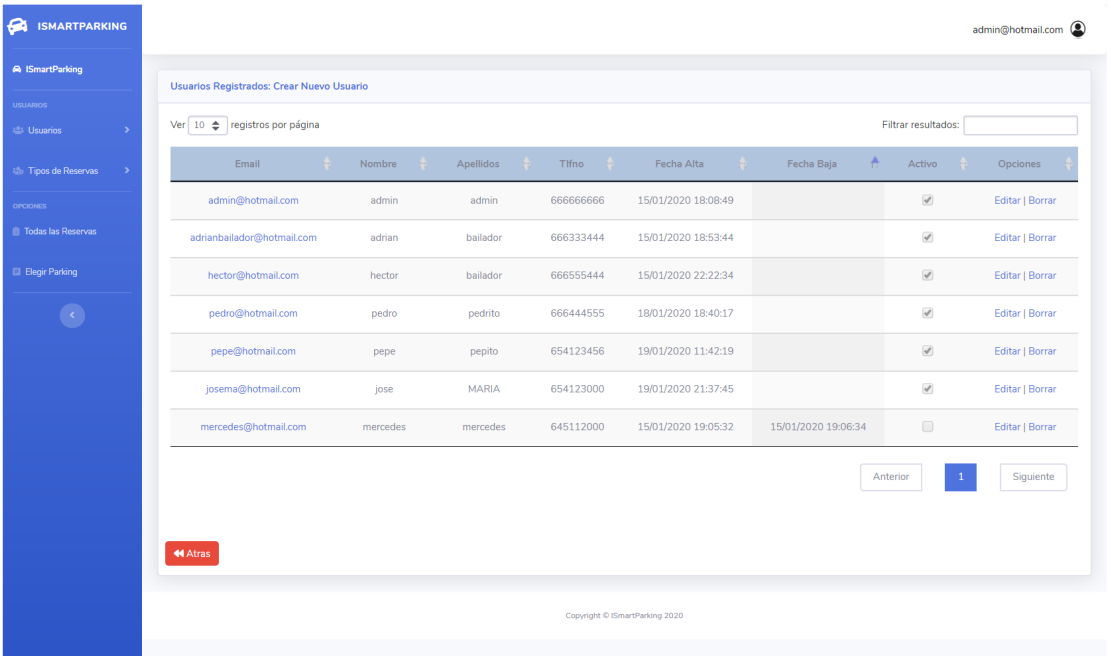

Figura 49: Usuarios Registrados en la aplicación

B.2.3. Modificar usuario

Para modificar un usuario hay que hacer clic en el botón “Editar” de la fila correspondiente al usuario que se quiere editar y a continuación los campos relativos al usuario estarán disponibles para editarse.

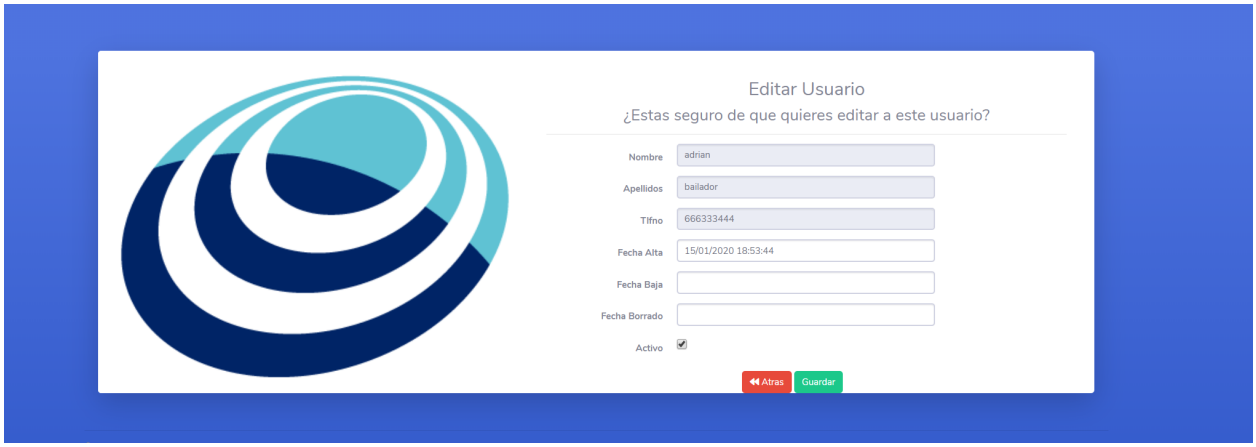

Figura 50: Modificar Usuarios.

B.2.4. Dar de baja al usuario

Para dar de baja a el usuario hay que hacer clic en el botón “Editar” del usuario correspondiente que se quiere dar de baja de la base de datos y quitar el check de “activo"que automaticamente aparecerá la fecha en la que se ha dado de baja el usuario.

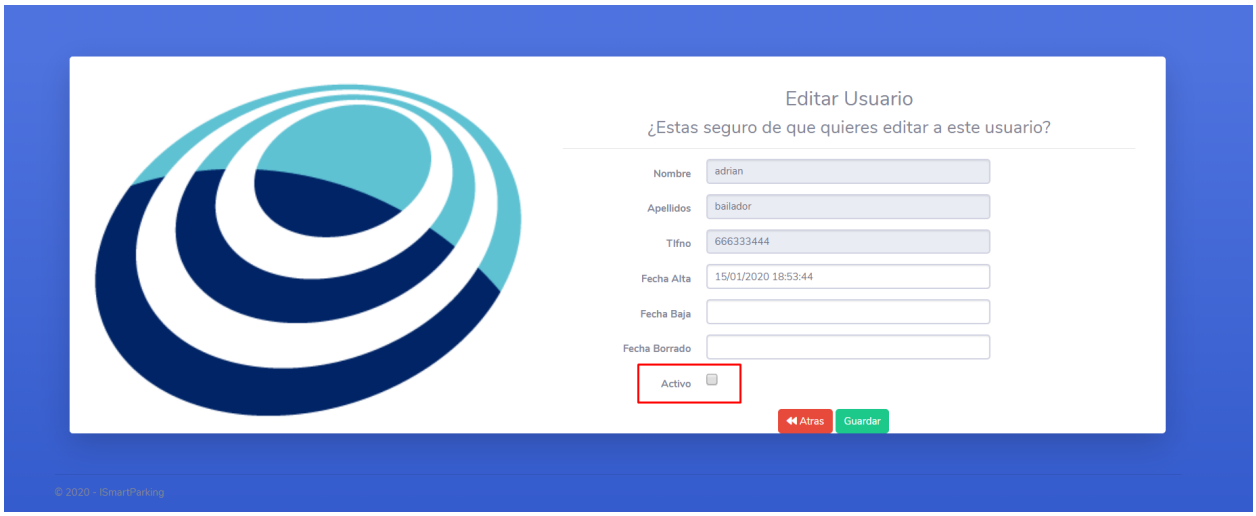

Figura 51: Dar de baja a usuarios.

B.2.5. Consultar reservas

Para consultar todas las reservas, nos vamos a la página de inicio y damos click a çonsultar reservas", en esta página nos apareceran un historico de reservas. Luego en la parte

inferior podremos consultar las reservas realizadas o canceladas. También podremos exportar en diferentes formatos para el tratamiento de estos datos.

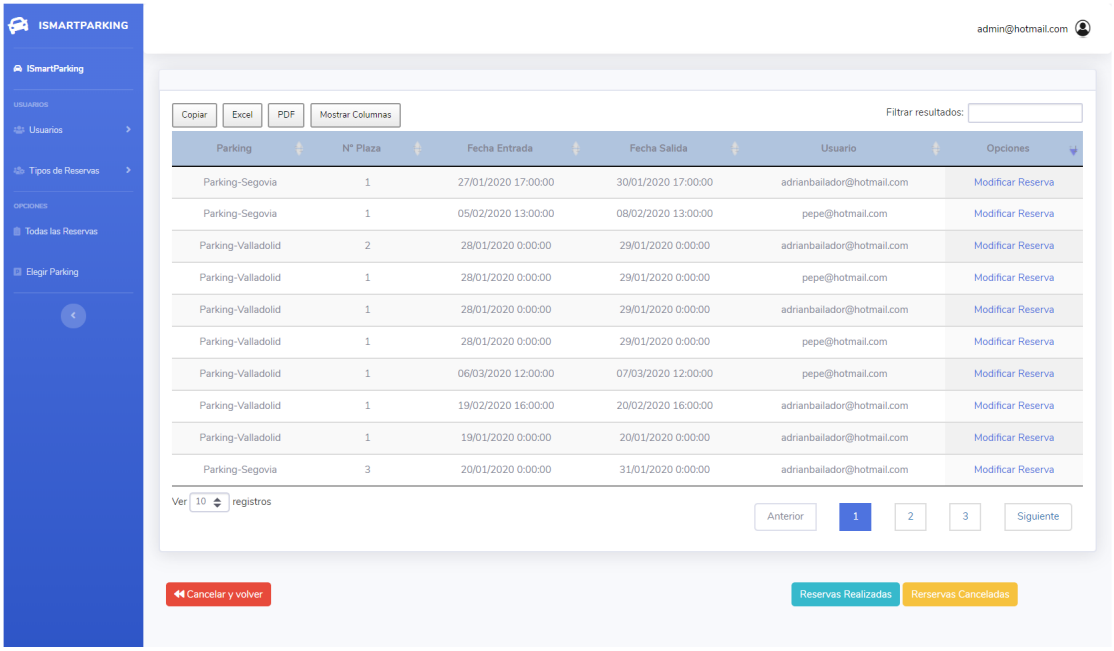

Figura 52: Consultar Reservas.

APÉNDICE C: Manual de Usuario

C.1. Introducción

Nuestra aplicación ayuda a los usuarios a realizar la reserva de una plaza, comprobando su disponibilidad, en distintos puntos de cada ciudad.

C.1.1. Descripción del documento

Este documento busca detallar la forma de utilización de la aplicación para que el usuario haga un uso correcto de ella.

C.2. Funcionalidades para el Usuario

C.2.1. Registro

Para poder hacer un uso correcto de la aplicación el Usuario debe registrarse en la aplicación. El Usuario accede a la página y se le muestra la siguiente pantalla.

Para realizar el registro el Usuario toca en el botón “¿Quieres Registrarte?” y aparece la siguiente pantalla.

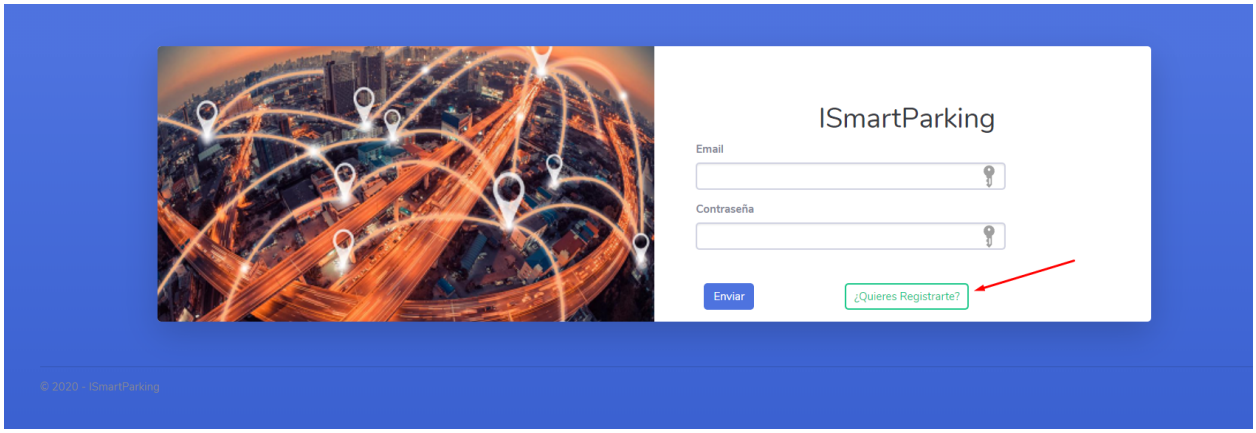

Figura 53: Registrar Usuario.

El siguiente paso es rellenar correctamente los campos requeridos y pulsar Registro. Si todo es correcto nos llevará a la pantalla de iniciar sesión

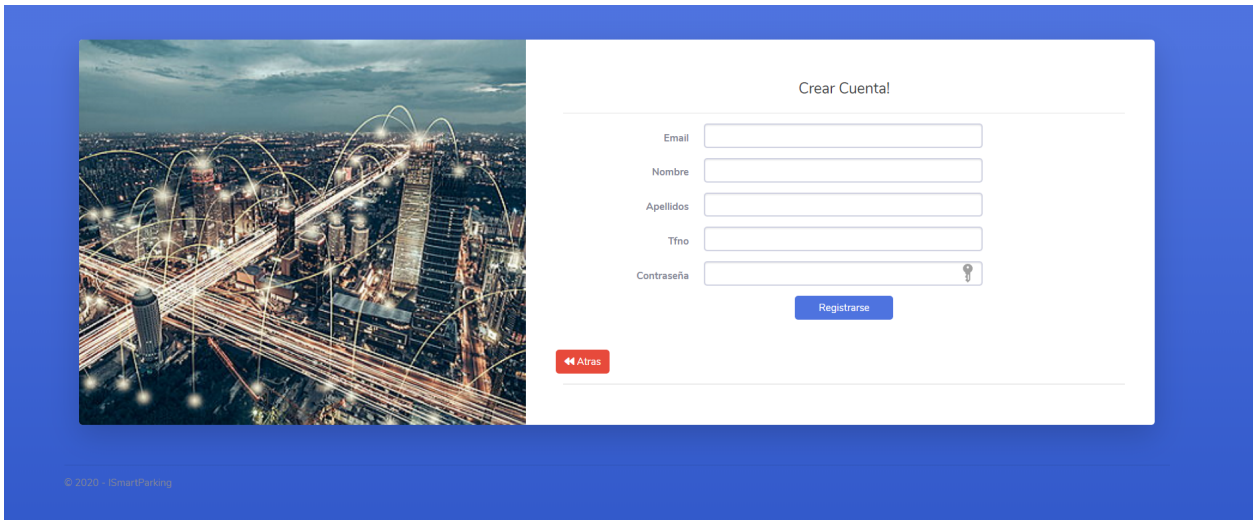

Figura 54: Completar registro de Usuario.

C.2.2. Iniciar Sesión

Para iniciar sesión hay que introducir el correo y la contraseña que se introdujeron en la tarea de registro. Una vez introducido las credenciales, la aplicación comprueba que los campos son correctos, si lo son, se carga la pantalla de inicio del usuario con las distintas opciones que la aplicación ofrece.

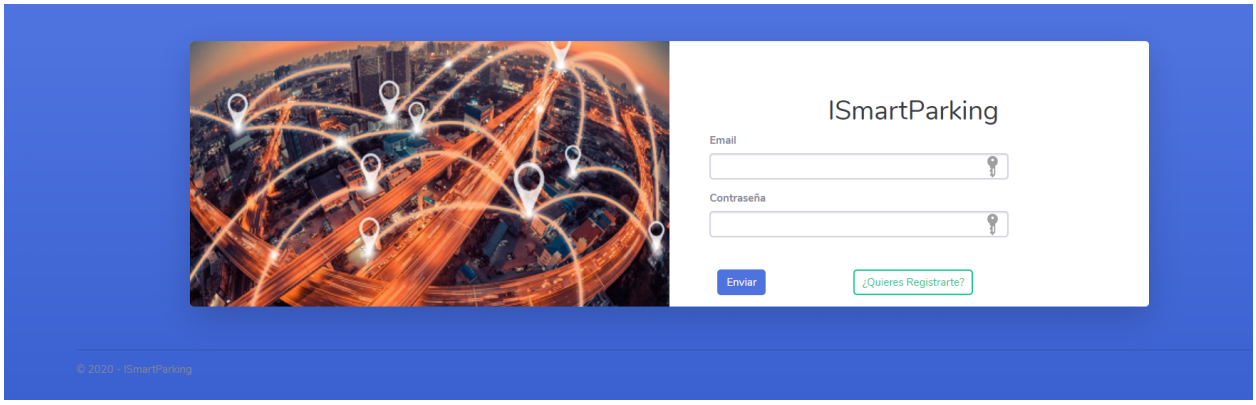

Figura 55: Iniciar Sesión.

C.2.3. Modificar datos personales

En este apartado el usuario podrá realizar modificación de sus datos personales (Nombre, apellidos, telefono, y darse de baja), para darse de baja, tiene que quitar el check de activo.

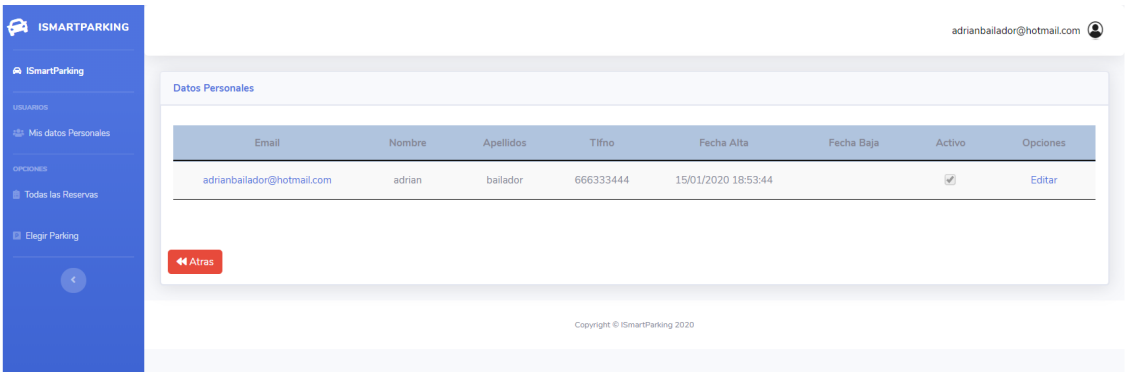

Figura 56: Modificar datos personales del Usuario Registrado.

C.2.4. Realizar Reserva

Para realizar una reserva, en la página principal tenemos que pulsar el botón de “Elegir Parking Para Reservar Plaza”. Nos llevará a otra página en la nos mostrara los parking disponibles, con unas determinadas caracteristicas.

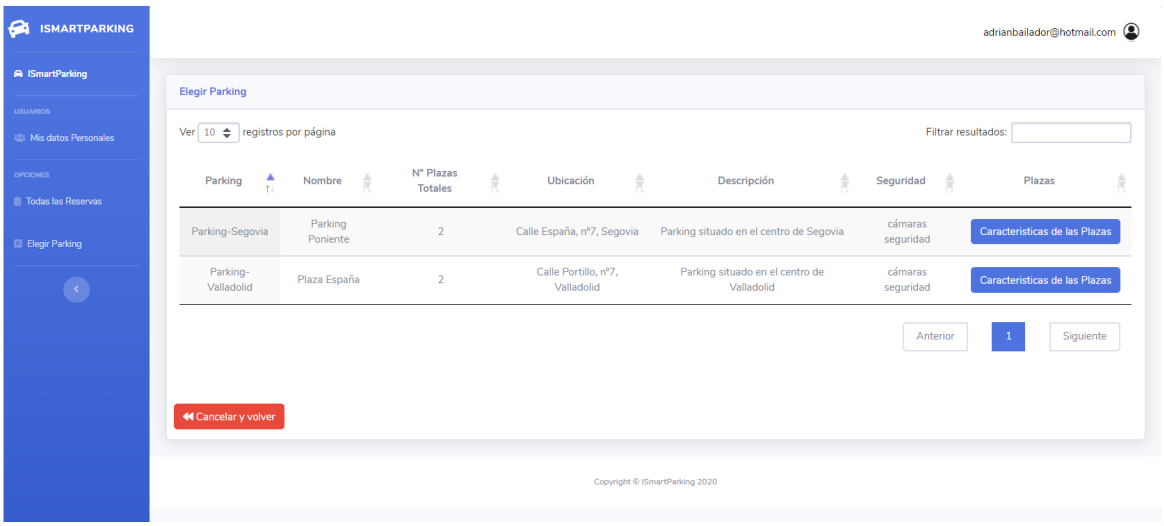

Figura 57: Seleccionar el parking

Para completar la reserva, pulsamos el boton de “Reservar Plaza”.

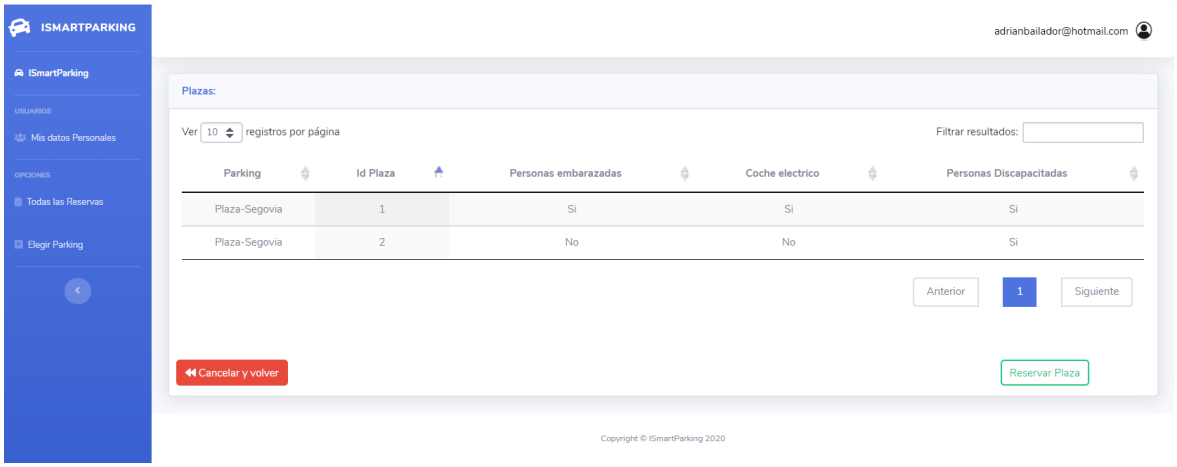

Figura 58: Seleccionar la plaza

Para completar la reserva, completamos los campos correspondientes para realizar la reserva.

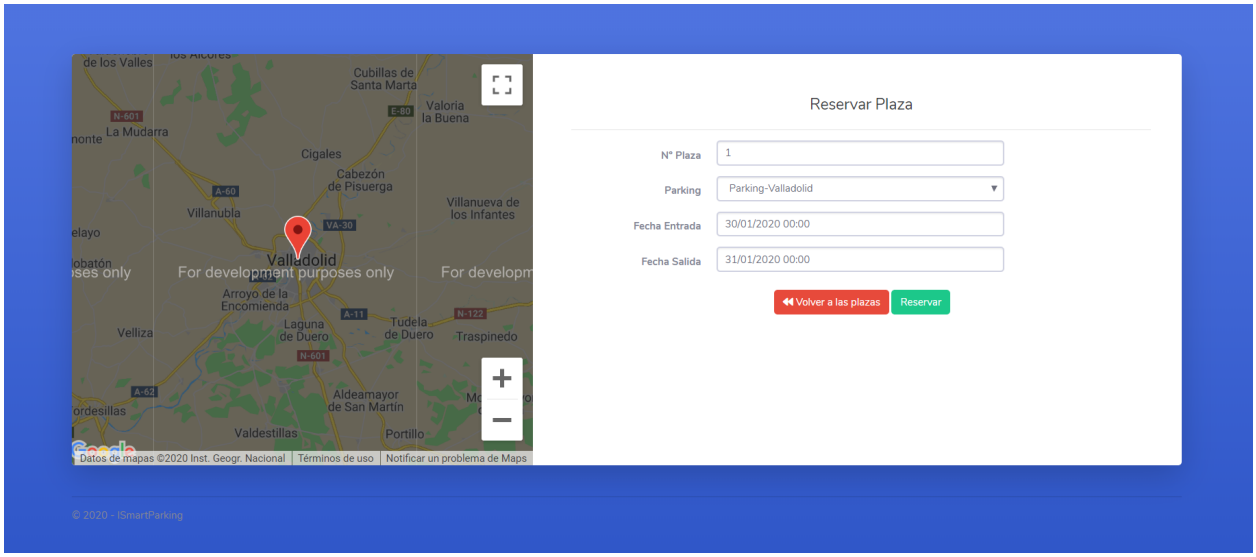

Figura 59: Realizar la reserva de la plaza elegida

En caso de que una plaza este ocupada por otro usuario en las fechas indicadas, nos saltará un aviso con la fecha en la que podremos ocupar dicha plaza.

Acrónimos

RGPD	Reglamento General de Protección de Datos
GE	Generic Enabler
API	Interfaz de Programación de Aplicaciones
CB	Context Broker
CEP	Procesamientos de Eventos Complejos
CPU	Unidad Central de Procesamiento
DB	Base de Datos
DNS	Sistema de Nombres de Dominio
GE	Habilitadores Genéricos
HDFS	Sistemas de Archivos Distribuidos en Aro
HTTP	Protocolo de Transferencia de Hipertexto
HTTPS	Protocolo seguro de transferencia de hipertexto
IdM	Gestor de Identidad
JSON	Notaciones de Objetos JavaScript
PAP	Punto de Administración de Políticas
PDP	Punto de Decisión de Políticas
PEP	Punto de Aplicación de Políticas
SaaS	Software como Servicio
SAML	Lenguaje de Marcado para Confirmaciones de Seguridad
URL	Localizadores Uniformes de Recursos
XACML	Control de Acceso del lenguaje de marcas Extensibles
XML	Lenguaje de Marcas Extensible
CU	Casos de Uso

Contenido del CD adjunto

El CD-ROM contiene en su interior los siguientes ficheros y documentos:

- **Manual de usuario.** Contiene información del rol usuario.
- **Manual de administrador.** Contiene información del rol admin.
- **Manual de instalación.** Contiene información de cómo instalar la aplicación.
- **Memoria.** Versión en PDF de la memoria del TFG.
- **Código fuente.** Contiene todos los ficheros necesarios para el correcto funcionamiento de la aplicación.
- **Copia de seguridad de la BBDD local.** Necesario para que funcione la aplicación.
- **Modelos de entidades de ORION.** Modelo de entidades del Parking y del tipo de plaza.