**UNIVERSIDAD DE CASTILLA-LA MANCHA**

**ESCUELA SUPERIOR DE INFORMÁTICA**

**GRADO EN INGENIERÍA INFORMÁTICA**

**TECNOLOGÍA ESPECÍFICA DE**

**INGENIERÍA DE COMPUTADORES**

**TRABAJO FIN DE GRADO**

# APEX Private Cloud

Miguel Ángel Moreno Chacón

Agosto, 2017


**UNIVERSIDAD DE CASTILLA-LA MANCHA**

**ESCUELA SUPERIOR DE INFORMÁTICA**

**DEPARTAMENTO DE TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN**


**TECNOLOGÍA ESPECÍFICA DE**

**INGENIERÍA DE COMPUTADORES**

**TRABAJO FIN DE GRADO**

# APEX Private Cloud


Autor: Miguel Ángel Moreno Chacón

Director Avanttic: Ángel Freire Ramírez

Director UCLM: Sebastián Reyes Ávila


Agosto, 2017

## PÁGINA DE CALIFICACIÓN

## TRIBUNAL:

**Presidente:**

**Vocal:**

**Secretario:**

## FECHA DE DEFENSA:

## CALIFICACIÓN:

| PRESIDENTE | VOCAL | SECRETARIO |
|---|---|---|
| Fdo.: | Fdo.: | Fdo.: |

## Resumen

Este Trabajo Fin de Grado se desarrolla en el marco de un proyecto de la empresa Avanttic S. L bajo el convenio FORTE de la UCLM, el cual consiste en el desarrollo de una aplicación denominada “APEX Private Cloud” encargada de la instalación de software de bases de datos Oracle junto con la creación de bases de datos en nubes privadas sobre diversos Oracle Linux, realizando dichas operaciones de forma completamente automática y transparente al usuario final mediante una interfaz sencilla e intuitiva desarrollada a través de Oracle APEX. Facilitando la realización de estas tareas a cualquier persona que carezca de los conocimientos técnicos requeridos para llevarlas a cabo. La idea del proyecto fue concebida bajo la necesidad de una herramienta sencilla e intuitiva que pudiera llevar a cabo en su justa medida las mismas acciones que ciertas herramientas propias de Oracle, pero evitando la necesidad de pagar las grandes sumas de dinero de sus licencias.

## Abstract

This End-of-Grade Work is developed within the framework of a project of the company Avanttic S. L. under the agreement FORTE of the UCLM, which consists of the development of an application called "APEX Private Cloud" in charge of installing software Oracle databases along with the creation of databases in private clouds on various Oracle Linux, performing such operations completely automatic and transparent to the end user through a simple and intuitive interface developed through Oracle APEX, facilitating the realization of these Any person who lacks the technical skills required to carry them out. The idea of the project was conceived under the necessity of a simple and intuitive tool that could take to exactly the same actions that certain tools of Oracle own but envying the need to pay the large sums of money of their licenses.

*A mi padre, mi madre y mi hermana, gracias.*

# Índice general

# Índice de ilustraciones

# Índice de tablas

# 1. Introducción

Vivimos una época en la que los datos, la información, lo es todo, el mundo gira en torno al valor de éstos y por ello es crucial contar con un sistema que permita gestionar toda esta información. Un sistema que nos permita almacenar toda esta información a lo largo del tiempo y que nos brinde facilidad para su acceso y uso.

Es aquí donde entran en juego lo que conocemos como bases de datos. Estos sistemas nos permiten tener almacenada información para su posterior uso. Permiten un acceso directo a toda la información que tengamos almacenada, modificarla, prescindir de ella o agregar más a la ya existente. Nos permiten almacenar tanto datos de gran simplicidad como datos que forman complejas relaciones entre ellos, creando un sistema heterogéneo de datos.

Una vez que tenemos la información organizada y estructurada lo que se necesita es tenerla cerca y controlada, físicamente hablando. Es por ello por lo que las organizaciones están optando por soluciones que pasan por tener las llamabas nubes privadas, que no es más que una serie de servidores alojados físicamente en el mismo lugar que la empresa que los posee. Esto permite poder tener almacenada toda la información útil sin depender de terceros obstaculizando así que personas ajenas a la organización puedan tener acceso a información sensible o de valor.

Superadas todas estas adversidades y llegados a la actualidad el problema que se plantea es que para poder hacer uso de todas las ventajas que supone el uso de bases de datos, como por ejemplo Oracle, como medio para almacenar la información de una corporación es que es necesario tener conocimientos técnicos de cómo llevar a cabo el despliegue y la instalación de todo lo necesario para hacer funcionar dicho sistema. Y esto, para la mayoría de personas supone un problema. A esto se le suma la problemática económica que supone el uso de ciertas herramientas corporativas que constan de licencias con un precio prohibitivo.

Si hay datos tiene que haber donde almacenarlos y siempre hay datos. Esto supone que sea cual sea el ámbito en el que te muevas dentro de este mundo antes o después vas a acabar haciendo uso de una base de datos, no siendo siempre posible tener alguien capacitado para realizar la labor que supone dejar una base de datos operativa y funcional, sumado al coste económico de dicho trabajo.

Por ello, lo que busca este proyecto es vencer las problemáticas técnicas y económicas que supone el disponer de bases de datos Oracle [1]. APEX Private Cloud nos permite realizar las siguientes operaciones.

- Instalar todo el software de una base de datos Oracle.
- Crear bases de datos completamente operativas.
- Realizar replicaciones de bases de datos existentes.

Todo ello a través de un sencillo asistente que nos guiará en cada uno de los pasos y que no requerirá de conocimientos de sistemas operativos, gestión de bases de datos. Tampoco tendrá ninguna repercusión económica ya que no necesita de caras y prohibitivas licencias de grandes gigantes de la informática como puede ser Oracle.

## 1.1. Motivación

El presente proyecto se desarrolla dentro del convenio FORTE [2]. Este convenio consiste en un acuerdo entre la Escuela Superior de Informática[1] de Ciudad Real y en este caso la consultora tecnológica Avanttic S.L.[2], con sede en Madrid y Barcelona.

La idea fundamental de los FORTE es que las empresas puedan particularizar un perfil profesional para los alumnos de la escuela de modo que se adapte mejor al puesto de trabajo que demande la empresa o en el cuál están interesados. Los convenios FORTE permiten a las empresas elegir un alumno de una especialidad determinada (llamadas intensificaciones) cuyas características o conocimientos impartidos en ésta esté más cercano al proyecto al cual la empresa quiere que el alumno realice.

Las empresas acogidas a los convenios FORTE ofrecen a los alumnos la posibilidad de realizar las prácticas en empresas, y en ese contexto, realizar también el Trabajo Fin De Grado que les permite terminar su carrera y obtener su título. Se busca que el alumno adquiera unas competencias y unas vivencias en un entorno de trabajo real aparte de la experiencia laboral que supone un extra a la hora de encarar una primera contratación,

[1] http://www.esi.uclm.es/
[2] http://www.avanttic.com/

destacando también que estas prácticas están remuneradas por la empresa que supone una ayuda de cara al desplazamiento del alumno a la sede de la empresa.

Hablando de números los convenios FORTE están compuestos por unas 900 horas de trabajo presenciales y 24 créditos ECTS (12 créditos TFG y 12 créditos de prácticas en empresa). Una vez realizado el TFG y su posterior presentación la empresa se puede plantear contratar al alumno.

Este es el primer proyecto FORTE orientado a la denominada “rama de sistemas” que lleva a cabo Avanttic S.L con la Escuela Superior de Informática.

## 1.2. Avanttic Oracle Platinum Partner

Avanttic S.L. se define así misma como una empresa 100% Oracle siendo Oracle Platinum Partner [3] debido a la experiencia y al trabajo de sus profesionales en los diversos ámbitos de trabajo de Oracle dentro de la informática.

El marco de trabajo de este proyecto viene marcado por la línea de trabajo alineada con Oracle que sigue Avanttic S.L. Por ello para el desarrollo de este proyecto se ha decidido usar ciertas tecnologías y entornos de trabajo que estén relacionadas de forma directa o indirectamente con el gigante de la informática. La finalidad de este proyecto para Avanttic S.L. es ofrecer a sus clientes una herramienta de desarrollo propio pero que continúe en las directrices que ha llevado a Avanttic S.L. a ser Oracle 100%.

## 1.3. Estructura del documento

El presente documento se compone de los siguientes capítulos:

***Capítulo 1: Introducción***

Breve información sobre cuál es el propósito de este trabajo y explicación sobre el documento.

***Capítulo 2: Objetivos***

Lo que se describe aquí es el objetivo que se pretende lograr mediante este trabajo de forma más específica. Se inicia con un objetivo general que explica de forma general el proyecto y unos objetivos específicos que buscan dividir el objetivo general en otros más sencillos que permitan realizar el proyecto de una forma más cómoda y abarcable.

### *Capítulo 3: Estado del arte*

Durante este capítulo se podrá obtener una visión general de cuáles son los temas involucrados en la realización de este TFG.

### *Capítulo 4: Metodología de trabajo*

Definiremos cuál es la metodología que se usará durante el desarrollo de este proyecto. Se pasará por una descripción teórica de la metodología y una implementación real de dicha metodología en el presente proyecto. Finalmente se hará hincapié en cada una de las herramientas utilizadas en el desarrollo del proyecto.

### *Capítulo 6: Resultados*

En este capítulo podremos ver cómo ha sido el desarrollo a lo largo del proyecto. Como veremos pasaremos a lo largo de las diferentes iteraciones contando en cada una cuales son las características o funcionalidades añadidas al sistema.

### *Capítulo 7: Conclusiones y propuestas*

Finalmente se hará una reflexión sobre si los objetivos que se buscaban alcanzar han sido cumplidos junto con una opinión sobre la realización del proyecto.

# 2. Objetivos del TFG

En este capítulo se expondrán todos los objetivos, tanto los generales como los específicos, que fueron propuestos como propósito de este TFG. En ellos se describirá su alcance y dimensión para dejar en detalle la finalidad de este proyecto.

## 2.1. Objetivo general

APEX Private Cloud surge de la necesidad de poner fácil las cosas a aquellos profesionales que demandan el uso de bases de datos para almacenar su información, sea cuál sea el ámbito, pero no disponen de los conocimientos técnicos necesarios para realizar la instalación y puesta en marcha de una base de datos. Conocimientos propios de gente perteneciente en su mayoría al mundo de sistemas y que incluyen entre otros la creación de usuarios del sistema operativo, gestión de procesos o administración de conexiones de red.

Lo que se quiere lograr con este proyecto es hacer de puente entre el usuario final y su necesidad de tener una base de datos completamente operativa en el menor tiempo posible y con las mínimas complicaciones técnicas al igual que realizan diferentes herramientas corporativas de Oracle, pero reduciendo el coste económico de dicha operación al mínimo posible.

APEX Private Cloud se compone de una serie de asistentes virtuales que se encargarán de guiar al usuario desde una fase inicial de instalación del software de bases de datos, hasta un asistente final que permitirá la creación de bases de datos sobre dicho software. Para ello se solicitará al usuario que rellene cierta información trivial como puede ser la ruta de instalación del software (aunque contará por valores por defecto), la cantidad de recursos a asignar a la base de datos o simplemente el nombre de la base de datos.

## 2.2. Objetivos específicos

Para lograr que nuestro proyecto sea abordable y exitoso debemos dividirlo en cargas de trabajo adecuadas. Esto nos permite tener un control total sobre cuál es el avance del

proyecto, adivinando posibles errores que pueden suponer el fracaso del proyecto o demoras insalvables dentro de este. Nos permite ver si éste va en la dirección que queremos, si está cumpliendo las expectativas y los objetivos que habíamos fijado en primera instancia y realizar cambios en consecuencia si lo vemos necesario.

Este proyecto se divide en dos iteraciones diferentes pero complementarias. La primera será el despliegue del software de bases de datos y la segunda la posterior creación de bases de datos. A continuación, se detallarán cada uno de estos objetivos específicos.

### 2.1.1. Despliegue del software para creación de bases de datos

El paso más básico que debemos plantear antes de hablar si quiera de bases de datos es instalar el software que permite que éstas funcionen. Este objetivo consistirá exactamente en eso, en la instalación de los binarios necesarios para la posterior creación de bases de datos Oracle 11g Release 2 [4]. Aquí se desarrollará tanto la parte interna del objetivo como es el desarrollo del código encargado de copiar el software al host remoto y de satisfacer los requisitos del sistema para la instalación del software como creación de usuarios, de directorios, etc. Todo ello siendo transparente al cliente final. Esta parte se realizará a través del software de instalación del propio Oracle el cual permite su instalación a través de líneas de comando y sin intervención de agentes externos. Por otro lado, se desarrollará también la parte externa, la cual está formada por una aplicación WEB desarrollada a través de Oracle APEX [5] que contendrá un asistente que guiará interactivamente al usuario a lo largo de la instalación del software, solicitando a éste información trivial pero necesaria para la instalación.

### 2.1.2. Despliegue y desarrollo de herramienta para creación de bases de datos

Una vez que el software está perfectamente instalado en el sistema deseado todo está listo para que el usuario final pueda crear las bases de datos que crea conveniente. Al igual que

en el objetivo anterior este contará con una parte de desarrollo interna que se encargará de pasar por parámetros las opciones elegidas por el usuario al asistente de creación de bases de datos de Oracle que permite la creación de bases de datos de forma no interactiva y que está contenido en el software instalado previamente. A su vez se desarrollará el asistente de creación de bases de datos en la interfaz web de la aplicación que solicitará al usuario información rudimentaria pero necesaria para la creación de la base de datos, credenciales de administrador, nombre de la base de datos o recursos a asignar a esta entre otros.

# 3. Estado del arte

A lo largo de este capítulo se hará un recorrido sobre los fundamentos sobre los que se asienta este proyecto, como son las bases de datos y la nube. Para ello se hará un pequeño análisis de todas estas tecnologías en el contexto que engloba este proyecto, permitiendo ver el potencial que tiene reunirlas todas juntas en un único sistema.

## 3.1. Bases de datos

### 3.1.1. Introducción

La definición más sencilla que podemos obtener es que una base de datos es una colección de datos organizados. Formados por esquemas, tablas, *queries*, vistas y otro tipo de objetos. Estos datos están organizados para modelar una realidad determinada, como, por ejemplo, modelar la disponibilidad de medicamentos en un hospital.

Un sistema de gestión de base de datos (DBMS[3] en sus siglas en inglés) es una aplicación software que permite la interacción entre el usuario, la base de datos y otras aplicaciones. Un DBMS de propósito general está diseñado para permitir la definición, creación, acceso, actualización y administración de una base de datos. Algunos *DBMS* conocidos son Oracle Database [4], MySQL[4], Postgresql[5] o Microsoft SQL Server[6], entre otros. Normalmente los DBMS son sistemas clasificados acorde al modelo que soportan, la mayoría de los sistemas bases de datos más populares que se utilizan desde 1980 soportan el Modelo Relacional [6] que está representado por el lenguaje SQL [7].

Una base de datos por norma general no soporta la portabilidad entre diferentes sistemas de gestión de bases de datos, aunque estos pueden hacer uso de ciertos estándares y lenguajes comunes como pueden ser el lenguaje SQL o el JDBC[7] que permiten a una misma aplicación interaccionar con diferentes sistemas de gestión de bases de datos.

[3] *Database-Management System*
[4] https://www.mysql.com/
[5] https://www.postgresql.org/
[6] https://www.microsoft.com/es-es/sql-server/sql-server-2016
[7] *Java Database Connectivity*

### 3.1.2. Visión General

Formalmente hablando una base de datos es vista como una serie de datos que están relacionados entre ellos debido a la forma en que se organizan. El acceso a estos datos es normalmente proporcionado por un sistema de gestión de bases de datos que consiste en una serie de softwares que permiten al usuario acceder a todos los datos que están contenidos en la base de datos. Este sistema de gestión proporciona diferentes formas de acceder, almacenar y obtener grandes cantidades de datos a la vez que permite gestionar como los datos están organizados.

Los sistemas de gestión (entendemos que de bases de datos) actuales proporcionan diferentes funciones que permiten la gestión de la base de datos y los datos asociados a estas. Estas funciones se pueden organizar en cuatro grupos principales:

- **Definición** de los datos – Creación, modificación y eliminación de los modelos que definen como se organizan los datos.
- **Actualización** – Permiten la inserción, modificación y eliminación de los propios datos almacenados.
- **Obtención** – Proporcionan la información en una forma que permite su uso directo o su preparación para un uso posterior. La información puede ser obtenida de la forma que está almacenada o a través de la alteración de otra información obtenida.
- **Administración** – Permite la monitorización, seguridad, integridad de los datos, control de concurrencia y recuperación de información en caso de fallo del sistema.

Físicamente hablando un servidor de bases de datos es un ordenador que tiene ejecutándose en su interior una base de datos y su sistema de gestión correspondiente. Normalmente estos servidores son multiprocesador, generosos en memoria y con discos en RAID[8]. Estos discos en RAID son usados para recuperar información en caso de que alguno de los discos falle.

[8] *Redundant Array of Independent Disks*

### 3.1.3. Historia

El uso de las bases de datos se extiende hasta los primeros días de la informática. A diferencia de las bases de datos de hoy en día que están enfocadas a un carácter más general, antaño lo sistemas estaban enfocados a un propósito específico. Inicialmente los sistemas de gestión de bases de datos solo estaban a disposición de las grandes organizaciones que eran las únicas capaces de soportar los costes de las complejas máquinas que albergaban estos sistemas.

## Sistemas de Navegación (1960)

Conforme los sistemas informáticos informáticos fueron mejorando su rendimiento y capacidad aparecieron los sistemas de bases de datos de propósito general. Apareció el interés en crear un estándar que sería conocido como la aproximación CODASYL.

Esta estrategia estaba basada en la navegación completamente manual por un conjunto de datos enlazados en red. Cuando se iniciaba la base de datos el programa devolvía un enlace al primer registro de la base de datos, el cual, a su vez contenía punteros a otros registros. Para llegar a un registro concreto se debía ir saltando y enlazando punteros hasta llegar al destino. No existía aún el concepto de búsqueda, algo que sería impensable en nuestros días.

Finalmente se encontraron soluciones a muchos de los problemas mediante sistemas de gestión basados en árboles binarios[9] que permitían realizar atajos.

## Sistemas Relacionales (1970)

La idea detrás de estos sistemas se basaba en que en vez de usar registros de tipo arbitrario enlazados se usara una tabla de registros de tamaño fijo. Esto define el denominado Modelo Relacional que resuelve este problema dividiendo los datos en una serie de tablas -o relaciones- normalizadas, en las que los elementos optativos han sido extraídos de la tabla

[9] Estructura de datos que mantiene los datos ordenados, permitiendo búsquedas, acceso secuencial, inserciones y borrado en tiempo logarítmico

para que ocupen espacio sólo si lo necesitan. En este modelo los registros se enlazan mediante una "clave".

En el modelo relacional una parte de la información se usa como clave, que identifica de forma única un registro o valor concreto. Por ello, cuando se quiere acceder a cierta información de la tabla se realiza mediante el uso de dicha clave. Por ejemplo, un DNI referencia de forma única una persona determinada.

## Sistemas SQL (finales de 1970)

A principios de 1970 IBM[10] comenzó a trabajar en un prototipo denominado *System R*. Este prototipo se basaba en sistemas multitabla, en los que los datos podían disgregarse de modo que toda la información de un registro determinado no tiene porque estar almacenado en un único trozo grande. Este prototipo junto con la estandarización del lenguaje SQL produjo que se llevar a producción, lo que terminaría siendo la conocida *Database 2*[11] *(o DB2).*

## Sistemas orientados a objetos (1980)

Durante la década de 1980 el auge de la programación orientada a objetos determinó la forma de manejar la información en las bases de datos. Programadores y diseñadores empezaron a tratar los datos como objetos. Esto permite establecer relaciones entre objetos y atributos, más que entre campos individuales.

## Sistemas NoSQL (2000)

El siglo XXI trajo una nueva tendencia en las bases de datos, el denominado NoSQL. Esta nueva tendencia introducía una línea no relacional que difiere con la línea clásica de las bases de datos. Por normal general no requiere esquemas fijos, evitan las operaciones del tipo *join* almacenando datos desnormalizados y están diseñadas para escalar

[10] Reconocida empresa multinacional estadounidense de tecnología y consultoría.
[11] https://www.ibm.com/analytics/es/es/technology/db2/

horizontalmente. La mayoría de ellas pueden clasificarse como almacenamiento del tipo clave-valor o bases de datos orientadas a documentos. Entre este tipo de bases de datos podemos encontrar algunos ejemplos como son MongoDB[12], Apache Cassandra[13] o HBase[14].

### 3.1.4. ACID

En las bases de datos se denomina *ACID* a las características de los parámetros que permiten clasificar las transacciones en los sistemas de gestión de bases de datos. *ACID* es el acrónimo de *Atomicity, Consistency, Isolation and Durability*: Atomicidad, Consistencia, Aislamiento y Durabilidad.

#### *Atomicidad*

Cuando una operación consiste en una serie de pasos, o todos ellos se ejecutan o ninguno, las transacciones nunca son parciales.

#### *Consistencia (Integridad)*

Esta propiedad asegura que solo se empiece a realizar aquello que vaya a ser terminado para evitar resultados parciales. Cualquier transacción debe llevar a la base de datos desde un estado válido a otro también válido. Esto nos permite asegurar que los datos son exactos, que los valores sean siempre los esperados, que no cambien ni varíen de forma que no deberían.

#### *Aislamiento*

Esta propiedad asegura que una operación no pueda afectar al resto de operaciones. Esto asegura que la realización de dos transacciones sobre la misma información sea de forma independiente evitando resultados inesperados o errores. Esta propiedad define cuando los cambios producidos por una operación son visibles al resto de operaciones concurrentes.

[12] https://www.mongodb.com/es
[13] http://cassandra.apache.org/
[14] https://hbase.apache.org/

***Durabilidad (Persistencia)***

Esta propiedad permite asegurar que una vez realizada la operación los cambios que haya efectuado perdurarán y no podrán ser revertidos.

Estos 4 requisitos permiten definir que un sistema de gestión de bases de datos es *ACID Compliant.*

## 3.2. La nube

La computación en la nube o conocido comúnmente como "La nube" es una serie de infraestructuras y softwares que permiten el acceso desde cualquier lugar a una serie de recursos compartidos, como, por ejemplo, redes, servidores, almacenamiento, … Siendo éstos provisionados con el mínimo esfuerzo y configuración, normalmente todo a través de Internet. La nube permite a los usuarios y a las empresas procesar o almacenar sus datos tanto en sus nubes privadas como en servidores de terceros alojados en centros de datos de forma que el acceso a los datos sea más fiable y eficiente. La computación en la nube se basa en la necesidad de compartir recursos como fin de conseguir economía de escala.

Algo a tener muy en cuenta es que la nube permite a las compañías evitar, o minimizar todo lo posible los costes asociados al despliegue de infraestructuras. Relegando estas tareas en un tercero las empresas pueden centrarse en su idea de negocio principal repercutiendo directamente sobre el crecimiento y la economía de éstas.

La disponibilidad de redes de alta capacidad junto con el bajo precio de los ordenadores *low-cost* y los sistemas de almacenamiento han contribuido a la expansión del uso de virtualización del hardware y de la arquitectura orientada a servicio (SOA[15]). Las empresas pueden escalar su potencia de cómputo según demanden mayor capacidad de almacenamiento o potencia de cálculo. La computación en la nube se ha convertido en un servicio altamente demandado debido a las grandes ventajas que supone el tener un alto poder de computación a un bajo coste, con un alto rendimiento, escalabilidad, accesibilidad y disponibilidad.

[15] *Service Oriented Architecture*

### 3.1.5. Modelos de servicio

La nube ofrece diferentes modelos de servicio según sean las necesidades de los usuarios. Estos servicios son los siguientes:

#### *Infrastructure as a service (IaaS)*

Es el modelo de servicio en la nube más básico. Este modelo ofrece servicios a APIs de alto nivel usadas para generar detalles de bajo nivel como pueden ser redes de comunicación, localización, particionado de datos, seguridad, backups… Dentro de este grupo podemos encontrar servicios como son los hipervisores Oracle VirtualBox [8] o Linux Containers[16]. Estos hipervisores pueden compartir unos recursos comunes y pueden escalar la creación de máquinas virtuales según sea necesario. Por otra parte, los contenedores de Linux pueden correr de forma independiente y aislada dentro un mismo sistema sin interferencia entre ellos.

#### *Platform as a service (PaaS)*

Los vendedores de servicios PaaS ofrecen un entorno de desarrollo para los desarrolladores de aplicaciones. Estos vendedores normalmente desarrollan una serie de herramientas y estándares que permitan el desarrollo y una serie de canales que permitan la distribución. En este modelo de servicio la nube prevé de una plataforma que incluye un sistema operativo, un entorno de ejecución de un lenguaje de programación determinado, una base de datos y un servidor web. De esta forma los desarrolladores de aplicaciones pueden desarrollar y ejecutar sus soluciones de software en la nube sin tener que tener en cuenta los costes y complejidad que supone la compra de las capas de hardware y software necesarias para esto. Algunos vendedores que ofrecen estos servicios son Microsoft Azure[17] o Google App Engine[18].

#### *Software as a service (SaaS)*

En este modelo de servicio los usuarios tienen acceso a un software de aplicación determinado y a bases de datos. Los proveedores de *cloud* se encargan de gestionar la infraestructura y la plataforma en la cual se ejecutan las aplicaciones. Normalmente se suele

[16] https://linuxcontainers.org/
[17] https://azure.microsoft.com/es-es/
[18] https://cloud.google.com/appengine/docs/

referir a este tipo de servicio como “software bajo demanda” ya que suele ser tarificado por uso. Este modelo de servicio elimina la necesidad de instalar y ejecutar las aplicaciones en los ordenadores propios de los usuarios, lo cual simplifica el mantenimiento y el soporte. Esto permite una alta escalabilidad ya que en caso de efectuarse una alta demanda de un servicio determinado con el simple clonado de las máquinas virtuales sería suficiente para afrontar la demanda, unos balanceadores se encargan de distribuir la carga entre las diferentes máquinas virtuales, siendo este proceso completamente transparente al usuario.

### 3.1.6. Características

La nube cuenta con una serie de características clave, las cuales son:

#### *Independencia de localización y dispositivos*

Permite a los usuarios acceder al sistema mediante navegadores web sin tener en cuenta cuál es su localización o que tipo de dispositivo usen para conectarse.

#### *Mantenimiento*

El mantenimiento para los usuarios de la nube es sencillo ya que no es necesario que sea instalado todo en el ordenador de cada uno de los usuarios y ésta puede ser accedida desde diferentes localizaciones.

#### *Reducción de costes*

Una de las claves de la nube es que permite un coste inmensamente menor para las organizaciones ya que no es necesario que éstas compren todo el software y hardware que necesitan para sus despliegues, eliminando también el coste asociado al mantenimiento y configuración de dichas capas. Ahorro mediante la economía de escala.

#### *Inquilinos múltiples*

La nube permite *multitenancy*[19] lo cual permite a diferentes organizaciones compartir un mismo recurso, pero de forma aislada entre ellas. Esto permite la centralización de la infraestructura (suponiendo menos costes), una mayor utilización de los recursos y

[19] Una sola instancia de la aplicación se ejecuta en el servidor, pero sirviendo a múltiples clientes u organizaciones.

eficiencia, junto con la posibilidad de soportar picos de carga que se produzcan puntualmente.

***Escalabilidad***

La escalabilidad de la nube reside en su dinamismo a la hora de aumentar o disminuir la cantidad de recursos que se usan según sean las necesidades del usuario. Estos recursos pueden ser generadores en *near real-time* (teniendo en cuenta el tiempo de encendido y arrancado de las máquinas virtuales, la localización de los recursos o el sistema operativo entre otros).

### 3.1.7. Tipos de nube

Podemos encontrar diferentes tipos de nube según sea su tipo de arquitectura o la forma en la que ofrecen sus servicios, no obstante, para no salirnos de los límites que nos concierne se explicarán únicamente la nube privada, la cual se usa en este proyecto y la nube pública por ser la de uso más extendido.

#### Nube privada

La nube privada (o *private cloud)* es aquella que opera solamente para una única organización, siendo gestionada por un tercero o por la misma organización. La nube privada puede ser *on premise,* lo cual significa que el *data center* está alojado en la localización de la propia empresa o *off premise,* lo cual significa que el *data center* está localizado remotamente pero que es de uso exclusivo y privado de la organización, ese hardware no es compartido con otras organizaciones. El despliegue de una nube privada cuenta con unos factores de decisiones más importante, ya que no permite un escalado como si se hiciera uso de una nube pública cuyos recursos son mucho mayores.

#### Nube pública

La nube pública es aquella donde los servicios son prestados a través de una red que es de uso público. Estos servicios pueden ser gratuitos. Técnicamente no hay diferencias a nivel

de arquitectura entre la nube privada y la nube pública, aunque hay que tener en cuenta los problemas de seguridad que puede suponer el uso de una red pública. Algunos proveedores de cloud como puede ser Amazon Web Services[20] (*AWS)* tienen la infraestructura que ofrecen en sus propios *data centers* lo cuales son accedidos por lo normal a través de Internet.

[20] https://aws.amazon.com/es/

# 4. Metodología de trabajo

Una vez que tenemos claro en que consiste nuestro proyecto, cuál es la finalidad de este es muy importante tener claro y saber cómo organizar el trabajo para conseguir que el proyecto prospere y siga adelante en los plazos acordados y cumpliendo los objetivos que nos hemos propuesto.

Tras analizar qué tipo de metodología puede ser la más conveniente para este proyecto, ya sea ágil o tradicional, se ha elegido una metodología ágil ya que este tipo de metodología se adapta mejor a la naturaleza de este proyecto.

Buscamos una metodología que sea flexible y amoldable, que nos permita ir desarrollando los objetivos de nuestro proyecto y comprobando su funcionamiento a corto plazo, basarnos en un desarrollo del tipo iterativo e incremental, donde los requisitos y las soluciones sean evolutivas y abiertas a cambios según sean las necesidades del proyecto en los momentos puntuales a lo largo del desarrollo de éste. Esta forma de encarar el proyecto nos permite adaptarnos a los problemas que pueden ir surgiendo e ir asignando mayor o menor prioridad a las diferentes partes del proyecto, hablamos de una toma de decisiones a corto plazo, una gestión completamente dinámica del proyecto.

Por otro lado queremos que el valor del producto que se va gestando vaya incrementando en pequeñas dosis de lo que podemos denominar “el software que funciona”, aquella parte de software para el cual hemos comprobado su funcionamiento sin errores, esta aproximación permite poder varios resultados de forma temprana, lo cual supone una motivación para el equipo de desarrollo al ver que el trabajo que están llevando a cabo funciona, por otra parte también permite ver si el proyecto va en la dirección que se había propuesto en un inicio o si es necesario una nueva forma de enfocarlo.

Por todas las características que debe tener la metodología de desarrollo de este proyecto se ha optado por una metodología ágil, más concretamente Scrum, a continuación, se detallará que es Scrum y como se ha aplicado a este proyecto. Seguidamente se detallarán las diferentes herramientas que han sido necesarias para el desarrollo de este proyecto.

## 4.1. Scrum

Scrum es una metodología ágil y flexible que permite gestionar el desarrollo del software, cuyo principal objetivo es maximizar el retorno de la inversión para el cliente. Permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Está basado en entregas parciales y regulares del producto final en base al valor que ofrecen a los clientes. Se caracteriza por el uso de un desarrollo iterativo e incremental en lugar de una planificación y ejecución completa del producto, en un solapamiento entre las diferentes fases del desarrollo evitando un desarrollo secuencial o en cascada, rapidez en los resultados, flexibilidad, trabajo en equipo, gran adaptación, control y transparencia. Scrum permite la creación de equipo auto-organizados, impulsando la comunicación verbal continua entre todos los miembros involucrados en el equipo. Es una metodología que está teniendo mucho apoyo debido a su fácil aprendizaje y por el poco esfuerzo que requiere comenzar a utilizarla.

## 4.2. Roles

Scrum se compone de una serie de roles que llevan a cabo las diferentes tareas que componen la metodología, estos se detallan a continuación.

### 4.1.2. Scrum Master

El trabajo del Scrum Master es la de eliminar todos los obstáculos que se producen y que impiden que el equipo alcance los diferentes objetivos. No debemos confundirlo con un líder ya que en Scrum los equipos son auto-organizados, sino que hace de protección entre el equipo y cualquier influencia externa que suponga un problema para ellos. Éste debe asegurar que el proceso Scrum se ejecuta de forma correcta. También trabaja con el producto owner para maximizar la inversión de estos.

### 4.1.3. Product Owner

Representa a los accionistas y clientes de los usuarios finales del software. Este se centra en la parte que concierne al negocio y es el responsable de que el proyecto supongo un valor superior al dinero que se ha invertido en él. También se encarga de trasladar la visión del proyecto al equipo, este escribe las historias de usuario, que representan un requisito escrito en un par de frases en lenguaje común y de priorizarlas por valor de negocio.

### 4.1.4. Team

Se compone de los profesionales que poseen los conocimientos técnicos necesarios para la realización del proyecto en su totalidad (análisis, diseño, desarrollo, pruebas, documentación, …). Estos se encargan del desarrollo del proyecto de manera conjunta produciendo el beneficio que justifica el desarrollo de éste. Transforman en un producto entregable los requisitos del proyecto. Se compone de grupos pequeños, auto-organizados.

## 4.3. Elementos de Scrum

Scrum está formado por una serie de componentes y actividades que definen su marco de trabajo, componentes como el product backlog, los sprints o el sprint backlog, a su vez cuenta con una serie de reuniones o actividades que sirven para organizar el proyecto en sus diferentes fases, estas son el daily scrum, el sprint planning o el sprint review. A continuación, se detallan cada uno de estos elementos.

### 4.4.1. Product Backlog

Está formado por un documento de alto nivel que vale para todo el proyecto. Consiste en todos los requisitos del proyecto mediante descripciones genéricas de las funcionalidades que es deseable que tenga el sistema, priorizadas según sea el retorno de inversión. Representa qué se va a desarrollar a lo largo del proyecto. Es visible por todos y solo puede ser modificado por el product owner. Contiene información a grandes rasgos como el valor

de negocio como el esfuerzo requerido para el desarrollo, esta información es utilizada para poder priorizar las tareas.

### 4.4.2. Sprint

El sprint es el período de tiempo en el cuál se realiza el trabajo. La duración de los sprints es recomendable que sea constante y que sea definida por el equipo basándose en sus anteriores experiencias. Podemos comenzar con una duración predeterminada e ir variándola según las necesidades del proyecto. Al final de cada sprint el equipo deberá tener un entregable que represente los avances logrados y que suponga un incremento del producto que pueda ser entregado al cliente con los menores problemas posibles.

### 4.4.3. Sprint Backlog

Está formado por aquellos requisitos del product backlog que van a ser desarrollados durante el siguiente sprint. Se describe como el equipo va a llevar a cabo la implementación de dichos requisitos durante el desarrollo del sprint. De forma general los requisitos se parten en tareas que son asignadas a horas de trabajo, no siendo ninguna superior a las 16 horas. Dichas tareas nunca son asignadas, son elegidas por los miembros del equipo según les parezca adecuado.

### 4.4.4. Daily Scrum

En cada uno de los días que se realiza un sprint se celebra una "ceremonia" sobre cuál es el estado actual del proyecto. Esta reunión sigue unas reglas específicas:

- La reunión debe empezar de forma puntual.
- Cualquier persona puede asistir, pero únicamente los *stakeholders* pueden hablar.
- La reunión tiene una duración máxima de 15 minutos independientemente del tamaño del equipo.

Durante la reunión deben contestar todos los miembros del grupo las siguientes preguntas:

- ¿Qué hiciste ayer?
- ¿Qué harás hoy?
- ¿Has tenido problemas?

La finalidad es que todos los miembros del equipo tengan conocimientos sobre si se están cumpliendo los tiempos establecidos durante el *sprint*.

### 4.4.5. Sprint Planning

Antes de comenzar cada *sprint* se lleva a cabo una reunión en la cual se pretende:

- Determinar que trabajo se realizará durante el *sprint*.
- Preparar, con el equipo al completo, el *sprint backlog* que dirá el tiempo y el esfuerzo que será necesario para realizar el trabajo.
- Ver e identificar cuanto será el trabajo que se va a realizar durante este *sprint*.
- Se toma aproximadamente como medida de tiempo para esta reunión una hora por cada semana de duración del *sprint*.

### 4.4.6. Sprint Review

Cada *sprint* termina con dos encuentros, el *customer review* y el *sprint retrospective*. Ambas ocurren en el último día del *sprint*. La duración de éstas es aproximadamente de 1 hora por cada semana que dure el *sprint*. A continuación, vemos e n más detalle de que tratan.

***Customer Review***

Esta reunión ocurre el último día del sprint. El propósito de esta reunión es que el equipo enseñe y muestre a los clientes y stakeholders el trabajo que han realizado a lo largo del sprint. Esta reunión es facilitada por el product owner, aunque no es raro que la realice el equipo. El cliente debe revisar lo siguiente:

- El trabajo que el equipo se comprometió a realizar.
- El trabajo que han completado.

- Decisiones clave que han tomado durante el *sprint*.
- Las métricas del proyecto.
- Una demo del trabajo.
- Revisión de la prioridad (para el próximo *sprint*).

Depende del equipo el cliente aceptará el producto justo en la reunión o se le dará una semana para que pruebe el producto y de su aprobación.

***Sprint Retrospective***

Esta reunión suele ocurrir después del *customer review*. El propósito de esta reunión es identificar qué cosas está haciendo bien el equipo y por tanto debe seguir haciendo, que cosas deben empezar a hacer para mejorar y que cosas deben dejar de hacer ya que merman la productividad. La reunión es facilitada por el *scrum master* y el *product owner* no suele estar en ella.

Destacar la importancia de esta reunión, el equipo debe trabajar continuamente en la aplicación de buenas prácticas y en comportamientos que eliminen los malos hábitos.

## 4.2. Herramientas

Para el desarrollo de este proyecto se ha hecho uso de una gran variedad de herramientas y tecnologías que acaparan diferentes ámbitos. En las siguientes hojas se explicará de forma breve pero precisa cual es la función de estas herramientas

### 4.2.1. Tecnologías

#### Oracle APEX

Oracle Application Express [5] (Oracle APEX), es la herramienta de desarrollo de aplicaciones web para la base de datos Oracle. Application Express permite diseñar, desarrollar y desplegar aplicaciones vistosas, sensibles y basadas en bases de datos, tanto en local como en la nube. Con sólo un navegador web y una experiencia de programación

limitada, puede desarrollar e implementar rápidamente aplicaciones profesionales que sean rápidas y seguras para cualquier dispositivo, desde el escritorio al móvil. Oracle Application Express combina las cualidades como productividad, facilidad de uso y flexibilidad, con las cualidades de una herramienta de desarrollo empresarial, como seguridad, integridad, escalabilidad, disponibilidad y construcción para la web.

## Xshell

Xshell [9] es un potente emulador de terminal que soporta diferentes protocolos como SSH, SFTP, TELNET, RLOGIN o SERIAL. Éste incluye diferentes características adicionales a otros como son las pestañas, el *forwarding* de puertos dinámicos, macros y otros.

Xshell cuenta con una versión gratuita de uso escolar de la cuál será uso en este proyecto. En la Ilustración 1 podemos ver la interfaz del programa.

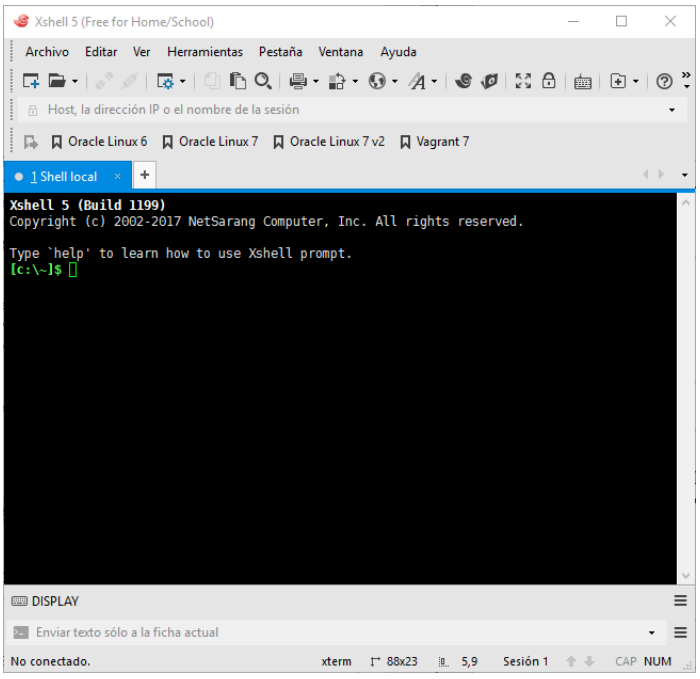


Ilustración 1: Interfaz Xshell

# Eclipse

Eclipse [10] es el IDE[21] más ampliamente usado para desarrollar en Java [11]. Está compuesto de un entorno de trabajo base y una serie de *plugins* que permiten añadir funcionalidades extra al entorno. Está escrito en Java y es usado en su mayoría por desarrolladores de aplicaciones Java, aunque puede ser usado para desarrollar en otros lenguajes. Eclipse será el entorno elegido para desarrollar el código Java de la aplicación debido a la experiencia en su uso durante estos años de atrás.

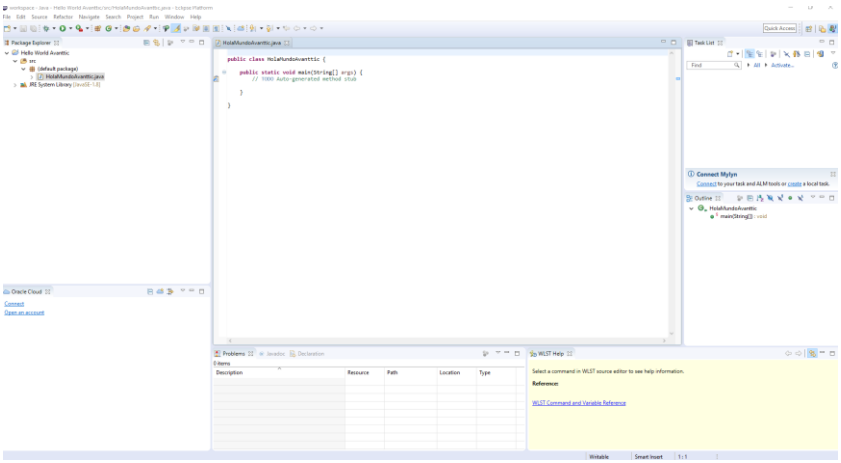

Ilustración 2: Interfaz de Eclipse Neon

# VirtualBox

En este proyecto tenemos la necesidad de usar diferentes sistemas con distintas características, sin embargo, contamos con un único entorno físico. Por ello debemos recurrir al uso de la **virtualización**, que no es otra cosa que una abstracción de los recursos de un ordenador, llamado **hipervisor** crea una capa entre el hardware de la máquina física, llamada "anfitrión" y el sistema operativo de la máquina virtual, llamado "invitado", dividendo los recursos entre uno o más entornos de ejecución.

Esta capa de abstracción que añadimos se encarga de manejar, gestionar y arbitrar los principales recursos del ordenador como son la CPU, la memoria, los periféricos y las

[21] *Integrated Development Environment*

conexiones de red. Así reparte de forma dinámica los diferentes recursos entre los entornos de ejecución que hay en la máquina física.

Esta máquina virtual simula una plataforma hardware autónoma que incluye un sistema operativo completamente operativo y que se ejecuta como si estuviera instalado directamente sobre el hardware.

VirtualBox es un software de virtualización o hipervisor creado originalmente por la empresa alemana innotek GmbH. VirtualBox soporta una gran cantidad de sistemas operativos (en modo anfitrión) como son GNU/Linux, MAC OS X, Microsoft Windows y Solaris/OpenSolaris, entre otros. Actualmente desarrollado por Oracle Corporation como parte de su familia de productos de virtualización. En la Ilustración 3 podemos ver cómo es su interfaz gráfica en su versión para Windows.

Ya que no nos es posible desarrollar y probar este proyecto en un entorno real, entendiendo entorno real aquel donde contamos con diferentes servidores físicos y una infraestructura de red para ellos haremos uso de este software que es gratuito bajo uso personal (Oracle VM VirtualBox) y que nos permite simular este entorno real bajo el cual realizaremos el proyecto.

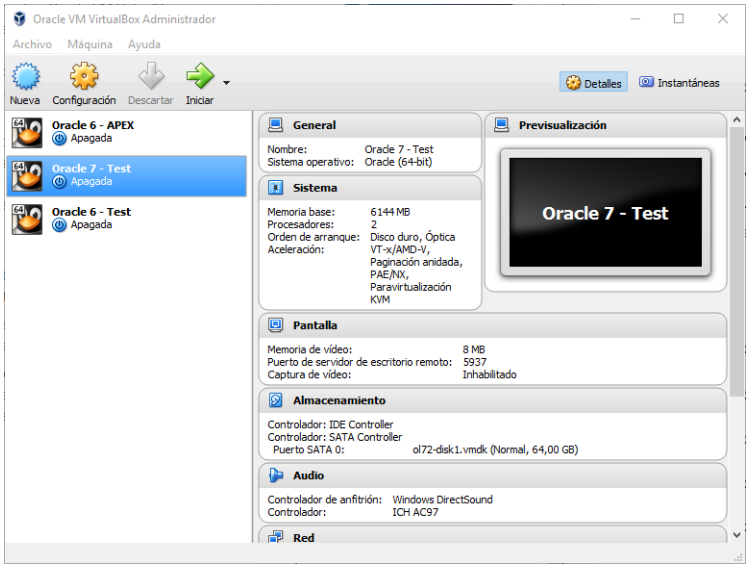


Ilustración 3: Interfaz VirtualBox

# Vagrant

Vagrant [12] es una herramienta que permite la creación y configuración de forma sencilla de entornos virtualizados a través de la línea de comandos. Vagrant es capaz de trabajar con múltiples proveedores de virtualización entre los que se encuentran VMware[22], Amazon EC2[23] o DigitalOcean[24], permitiéndonos trabajar también con VirtualBox. Destacar también que permite usar sistemas de gestión de la configuración como Chef[25], Salt[26] o Puppet[27].

Vagrant es muy importante en este proyecto porque introduce el concepto de *box* o "caja". El concepto de *box* representa un completo de ejecución con determinadas características, este puede ser usado por cualquier persona en cualquier entorno de trabajo. Para entenderlo, estos *boxes* son entornos finales completamente operativos, por ejemplo, pueden ser máquinas virtuales tales como un Debian[28] que viene con un Apache instalado, un Oracle Linux [13] con una base de datos versión 11 o un Ubuntu[29] con Docker[30] instalado. Lo único que tenemos que hacer es acceder a un repositorio públicos de *boxes* y buscar aquel *box* que satisfaga nuestras necesidades, pudiendo crear los nuestros propios y compartirlos.

Una vez que hemos encontrado el entorno que más nos convenga simplemente mediante un par de comandos podremos levantar dicha máquina virtual, que se ejecutará a través de VirtualBox en nuestro caso, y estará completamente operativa.

Vagrant es muy necesario porque permite levantar y apagar las máquinas virtuales de forma muy rápida, destruirlas y crearlas cuantas veces como queramos en unos pocos comandos. Esto nos da una versatilidad enorme ya que si nuestro entorno de pruebas empieza a fallar por cuales sean los motivos únicamente mediante un par de comandos tendremos el entorno de nuevo levantado con la configuración inicial. Además, podemos definir ciertas propiedades del entorno a través de un archivo de configuración llamado *Vagrantfile* que almacena la configuración del entorno de virtualización, propiedades como configuración

[22] https://www.vmware.com/es.html
[23] https://aws.amazon.com/es/ec2/
[24] https://www.digitalocean.com/
[25] https://www.chef.io/chef/
[26] https://saltstack.com/
[27] https://puppet.com/
[28] https://www.debian.org/index.es.html
[29] https://www.ubuntu.com/
[30] https://www.docker.com/

de red, recursos asignados, *hostname* o el gestor de configuración que vayamos a usar (si lo usamos).

## Database Configuration Assistant (DBCA)

*DBCA* [14] es la herramienta desarrollada por Oracle que permite la configuración de una base de datos Oracle a través de un asistente interactivo. Este software está incluido en los binarios del software de bases de datos de Oracle. Puede ejecutarse de forma remota a través del *forwarding* del X11.

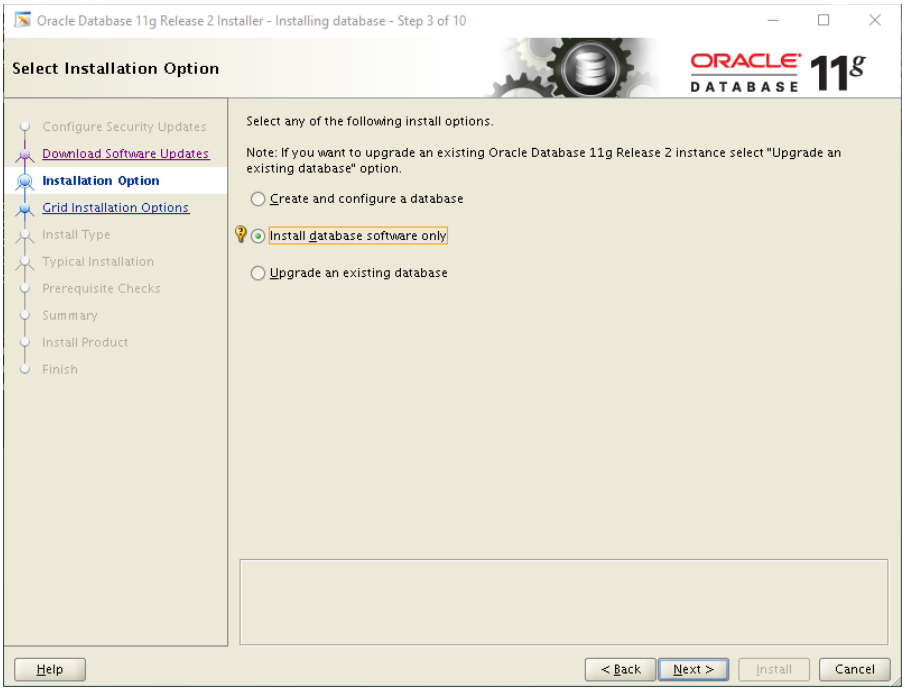


Ilustración 4: Interfaz DBCA

## Oracle Universal Installer

Oracle Universall Installer [15] es la herramienta desarrollada por oracle que permite la instalación de diversos softwares de Oracle a través de un asistente interactivo. Una serie de ventanas te guiarán a lo largo de la instalación solicitando ciertos parámetros de

configuración necesarios para la instalación del software deseado. Puede ejecutarse en remoto a través del *forwarding* del X11.

## SSH

SSH [16] o Secure Shell es como se denomina al protocolo de red criptográfico que permite ejecutar operaciones seguras a través de redes inseguras. SSH provee de un canal seguro a través de una red no segura en una arquitectura del tipo cliente-servidor. Dando nombre también al programa que implementa dicho protocolo, que se localiza en la capa de aplicación del modelo TCP/IP.

SSH sirve para poder acceder a máquinas remotas a través de una red, permitiendo controlar de forma completa dicha máquina a través de la línea de comandos pudiendo ejecutar programas, copiar archivos u otras funciones. También podemos realizar operaciones como gestionar nuestras claves RSA no teniendo así que escribir la contraseña del usuario remoto cada vez que nos conectamos. En resumen, nos permite realizar una administración remota de ordenadores a través de la red de forma segura.

Destacar que SSH usa técnicas de cifrado que hacen que la información que va a través de la red lo haga de manera no legible, evitando así que terceros puedan acceder a los datos.

Aquí se hará uso de OpenSSH (paquete para Linux) que utiliza por defecto la versión 2 de este protocolo y que implemente un mayor nivel de seguridad.

## SQL*Plus

SQL*Plus [17] es una herramienta de línea de comandos de Oracle que permite ejecutar comandos SQL y PL/SQL de forma interactiva o a través de scripts. Ésta es una herramienta simple con una línea de comandos básica. Es usada comúnmente por los DBA's[31] como interfaz para la instalación de software de Oracle.

[31] *Database Administrator*

## Oracle SQL Developer

Oracle SQL Developer [18] es un IDE para trabajar con SQL. Éste suporta diferentes productos de Oracle y una gran variedad de plugins que permite conectarse con bases de datos que no sean de Oracle.

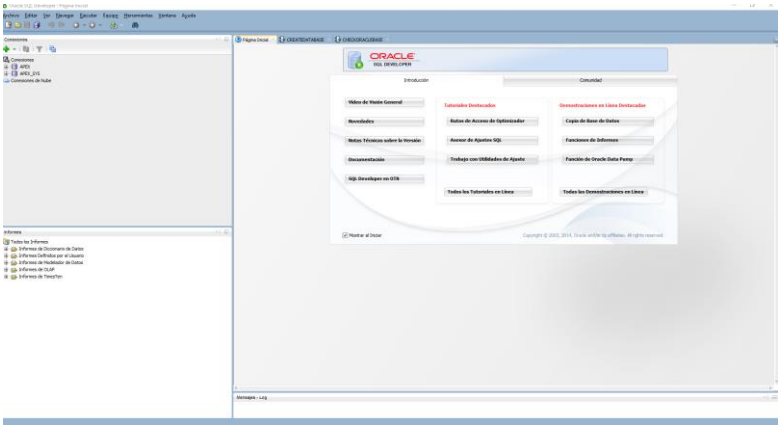

Ilustración 5: Interfaz SQL Developer

### 4.2.2. Hardware

Para el desarrollo del proyecto Avanttic S.L. ha proporcionado el equipo hardware de la Tabla 1. Contamos con un hardware con abundante memoria RAM que nos permitirá correr varias máquinas virtuales simultáneamente y así poder simular el entorno que queremos.

| **LATITUDE E6430s - DELL** |
|---|
| Pantalla LED 14” |
| Intel Core i5-3340M @ 2.70 GHz |
| 16GB – 2 X 8GB SDRAM DDR-1600 MHz |
| SSD 850 EVO 250GB |

Tabla 1: Hardware del sistema

### 4.2.3. Lenguajes

Para el desarrollo del proyecto se ha hecho uso de diferentes tecnologías según sea la función que vayan a desempeñar en el sistema, éstas se detallan a continuación.

#### Java

Java [11] es un lenguaje de programación concurrente y orientado a objetos. Su objetivo principal es la característica de escribir el código una única vez y poder ejecutarlo sobre cualquier sistema, esto significa que el código Java se compila para ser ejecutado en cualquier JVM[32] sin importar la arquitectura sobre la que se ejecute, permitiéndonos abstraernos de lo subyacente.

En 2016 era uno de los lenguajes de programación más usados sobre todo en aplicaciones del tipo cliente-servidor.

Java ha sido el lenguaje utilizado para el desarrollo de la mayoría del código de este proyecto que será explicado en la arquitectura del sistema.

#### Bash

Bash [19] es un intérprete de comandos de los sistemas Unix y a su vez un lenguaje de programación del proyecto GNU.

Normalmente se ejecuta en una ventana de texto donde el usuario se encarga de introducir comandos que provocan diversas acciones como vemos en la Ilustración 6: Sesión Bash. Éste será el medio con el que interactuaremos con el sistema operativo destino a través de los diferentes mecanismos que aporta, como el copiado de archivos, el ejecutado de *scripts* o la gestión de usuarios y permisos.

[32] *Java Virtual Machine*

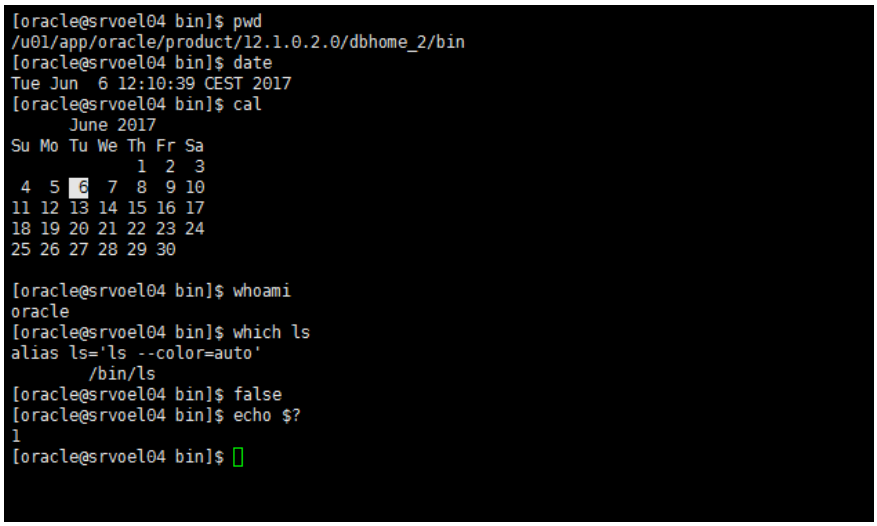


Ilustración 6: Sesión Bash

## SQL

SQL [7] o *Structured Query Language* es un lenguaje específico de dominio que permite dar acceso a un sistema de gestión de bases de datos. Éste permite hacer consultas con el fin de obtener información de bases de datos o modificarla.

## PL/SQL

PL/SQL [19] o *Procedural Language/Structured Query Language* es una extensión procedural de SQL y de las bases de datos relacionales de Oracle. PL/SQL incluye elementos como condicionales y bucles, permite declarar variables, constantes, procedimientos y funciones. También permite usar *arrays* y manejar excepciones.

### 4.2.4. Sistemas Operativos

Debido a que el proyecto se desarrolla en varios entornos haremos uso de varios sistemas operativos, por un lado, usaremos Windows para soportar la virtualización mediante Vagrant y VirtualBox, siendo el sistema final de trabajo Oracle Linux, por un lado, soportando Oracle APEX mediante una base de datos Oracle y por otro lado soportarán los

sistemas finales donde se alojará tanto el software de las bases de datos como las propias bases de datos.

A continuación, se explican brevemente sendos sistemas operativos.

## Microsoft Windows

Microsoft Windows [21] o Windows es una familia de sistemas operativos gráficos desarrollados y vendidos por Microsoft. Creo que no necesitan presentación.

Ilustración 7: Windows 10

## Oracle Linux

Oracle Linux es una distribución Linux gratuitamente distribuida por Oracle bajo licencia GNU. Está compilada a partir de Red Hat Enterprise Linux (*RHEL*). Oracle Linux será usado tanto para albergar la base de datos bajo la que funciona Oracle APEX tanto como sistemas finales donde se hará el despliegue del software de bases de datos y las posteriores bases de datos. En la  Ilustración 8 podemos ver la interfaz gráfica de la máquina virtual de un Oracle 6.

Ilustración 8: Oracle Linux 6

### 4.2.5. Otros

El resto de herramientas que no encajan en ninguna de las otras categorías se explican a continuación.

#### Draw.io

Draw.io [20] es un software online gratuito para crear diagramas de flujos, diagramas de procesos, UMLs y diagramas de red.

#### Microsoft Word

Microsoft Word [21] es un procesador de texto desarrollado por Microsoft. Su versión comercial viene bajo el componente Microsoft Office. Microsoft Word será usado para el desarrollo de la documentación. En la Ilustración 9 podemos ver su interfaz.

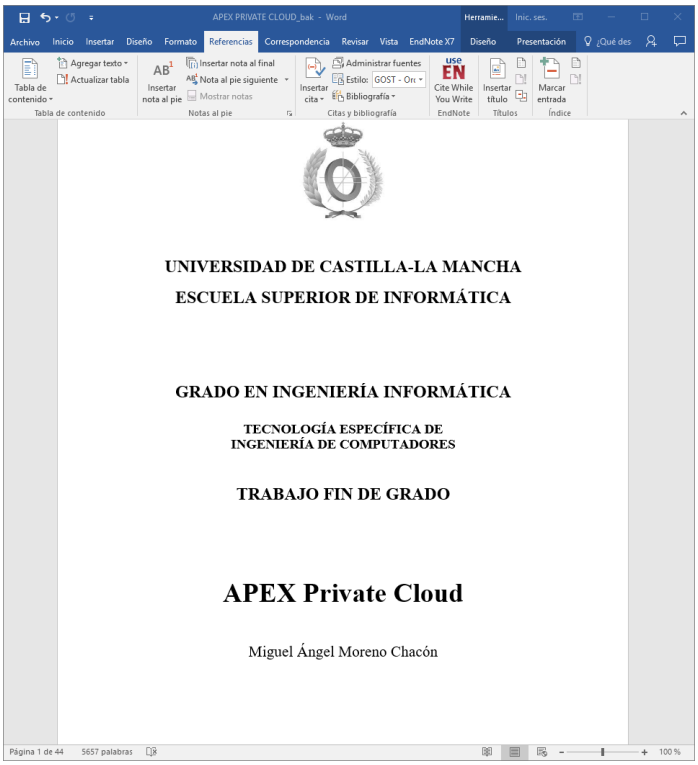


Ilustración 9: Microsoft Word 2010

# Atom

Atom [22] es un editor de texto *open-source* para Windows, Linux y macOS con soporte para *plugins* y escrito en Node.js, cuenta con soporte para *Git*[33] y está desarrollado por GitHub. Atom es una aplicación de escritorio que usa tecnologías web. Está basado en Electron, un *framework* que permite aplicaciones de escritorio usando Chromium y Node.js. Está escrito en CoffeeScript y Less. Atom cuenta por defecto con soporte para la mayoría de lenguajes usados actualmente y para otros tantos a través de *plugins*.

[33] Tecnología de control de versiones


Ilustración 10: Atom

### 4.2.6. Versiones

A continuación, se detallarán en su conjunto las versiones de las herramientas usadas para el desarrollo del proyecto, comentadas anteriormente.

- Oracle APEX 5.1.0
- Xshell 5.0
- Eclipse 3.6.1
- VirtualBox 5.1
- Vagrant 1.9.3
- Oracle Database 12.1.0.2.0
- OpenSSH 1.0
- SQL* plus 12.1.0.2.0
- Oracle SQL Developer 4.0.3
- Java 1.6
- Microsoft Windows 10

- Oracle Linux 6
- Microsoft Office Word 2010
- Atom 1.16.0

# 5. Resultados

A continuación, se presentarán que resultados han sido obtenidos a lo largo del desarrollo del presente proyecto mediante la aplicación de la metodología explicada previamente. Inicialmente se hablará del *sprint* inicial que describe la planificación total del proyecto, seguidamente se detallarán el resto de *sprints* y su desarrollo.

## 5.1. Sprint 0: Planificación Inicial

El *sprint 0* tiene como objetivo principal la total planificación del proyecto, para ello se deben definir el equipo de proyecto, las historias de usuario, el plan de proyecto y las estimaciones temporales del mismo. Las tareas que están asociadas a este *sprint* podemos verlas en la Tabla 2.

| **Tarea** | **Estimación** |
|---|---|
| Redacción del anteproyecto | 20 h |
| Definición del equipo de proyecto | 1 h |
| Definición de la historia de usuario | 12 h |
| Realización del plan de proyecto | 1 h |
| Realización de la estimación temporal | 6 h |
| **Total** | 40 h |

Tabla 2: Tareas asociadas al sprint 0

### 5.1.1. Equipo Scrum

De acuerdo con cómo se ha explicado en el capítulo de Metodología el equipo de trabajo estará formado por los siguientes roles:

- ***Scrum Master:*** Ángel Freire Ramírez
- ***Product Owner:*** Jesús García Hernández
- ***Development Team:*** Miguel Ángel Moreno Chacón

### 5.1.2. Product Backlog

Siguiendo los requisitos que fueron descritos en el documento del proyecto que fue definido por *avanttic* se ha realizado una descomposición de dicho trabajo en historias de usuario asociadas a unos tiempos y una prioridad. En la Tabla 3 podemos ver cuáles han sido estas historias de usuarios formando así el *Product Backlog* de este proyecto.

| Id. | Historias de Usuario | Estimación | Prioridad |
|---|---|---|---|
| 1 | Diseño de la arquitectura | 40 h | Alta |
| 2 | Diseño del entorno de trabajo | 24 h | Alta |
| 3 | Despliegue del entorno de trabajo | 32 h | Alta |
| 4 | Análisis para instalación de software de bases de datos | 30 h | Alta |
| 5 | Implementación de lógica para instalación de software de bases de datos | 150 h | Alta |
| 6 | Diseño e implementación de interfaz para instalación de bases de datos | 40 h | Media |
| 7 | Análisis para creación de bases de datos | 30 h | Alta |
| 8 | Implementación de lógica para creación de bases de datos | 130 h | Alta |
| 9 | Diseño e implementación de interfaz para instalación de bases de datos | 40 h | Media |

Tabla 3: Historias de usuario

Cada una de las historias de usuario se muestra mediante una tabla en su correspondiente *sprint* y estarán compuestas por los siguientes campos:

- **Número:** indica el identificador de la historia de usuario.
- **Nombre:** título que identifica la historia de usuario.
- **Rol:** rol que tiene la persona que abordará la historia de usuario.
- **Prioridad:** prioridad determinada de la historia de usuario.
- **Esfuerzo:** tiempo que se estima para el desarrollo de la historia de usuario (horas).
- **Sprint:** *sprint* en el cuál se realiza la historia de usuario.
- **Descripción:** detalles sobre la historia de usuario.
- **Postcondición:** cuál es el producto obtenido tras la realización de la historia de usuario.

### 5.1.3. Plan de proyecto

Ahora que se ha dividido el trabajo a realizar y que se ha priorizado a través de las historias de usuario en el *Product Backlog* hemos asignado dichas historias de usuario a los diferentes *sprints* de los que se compone el proyecto. En la Tabla 4 podemos verlas.

| Sprint | Nombre | Historias de usuario |
|---|---|---|
| 0 | Planificación inicial | - |
| 1 | Diseño de la arquitectura | 1-2 |
| 2 | Preparación del entorno | 3 |
| 3 | Lógica de instalación de software de BBDD | 4-5 |
| 4 | Interfaz de instalación de software de BBDD | 6 |
| 5 | Lógica de creación de BBDD | 7-8 |
| 6 | Interfaz de creación de BBDD | 9 |

Tabla 4: Sprints del proyecto

### 5.1.4. Estimación temporal

A continuación, se muestra cuál ha sido la estimación de tiempo para este proyecto. Para dicha aproximación se ha tenido en cuenta la fecha de inicio del proyecto que viene dada por mi incorporación a la plantilla de trabajo de Avanttic S.L. y la duración de cada uno de los *sprints* de forma aproximada. Se han tenido en cuenta los festivos nacionales y de la comunidad de Madrid. Para poder estimar la duración de los *sprints* se ha seguido la técnica de *Juicio de Expertos*[34] formado por el tutor en la empresa, arquitecto de infraestructuras y un DBA experimentado también perteneciente a Avanttic S.L. La estimación se ha hecho en función del contenido del *sprint* y de los conocimientos de los expertos basados en su experiencia en el ámbito. En la Tabla 5 podemos ver la estimación temporal de los diferentes *sprints* de los que se compone el proyecto.

[34] Es un conjunto de opiniones que pueden brindar profesionales expertos en una industria o disciplina, relacionadas al proyecto que se está ejecutando.

| Sprint | Nombre | Estimación |
|---|---|---|
| 0 | Planificación inicial | 40 h |
| 1 | Diseño de la arquitectura | 64 h |
| 2 | Preparación del entorno | 32 h |
| 3 | Lógica de instalación de software de BBDD | 180 h |
| 4 | Interfaz de instalación de software de BBDD | 40 h |
| 5 | Lógica de creación de BBDD | 120 h |
| 6 | Interfaz de creación de BBDD | 40 h |
| **Total** | | 516 h |

Tabla 5: Duración sprints

- **Fecha de inicio del proyecto:** lunes 6 de marzo
- **Fecha de fin del proyecto (aprox):** miércoles 7 de junio

## 5.2. Sprint 1: Diseño de la arquitectura

Según el plan de proyecto en este *sprint* se han llevado a cabo las historias de usuario 1 y 2. En estas historias de usuario se han realizado un análisis y estudio de las herramientas que la empresa ha determinado que deben usarse para el desarrollo de la arquitectura de este proyecto. Por ello se ha analizado el funcionamiento de Oracle APEX, la ejecución de Java a través de procedimientos almacenados en la base de datos y el funcionamiento y ejecución de comandos remotos a través de SSH. Posteriormente, una vez analizadas estas herramientas y comprendido su funcionamiento y alcance se ha realizado el diseño de una arquitectura de red que permita la comunicación entre los diferentes componentes del sistema de forma automática y una arquitectura de ejecución que permita que la información fluya desde un extremo a otro de forma transparente al usuario.

### 5.2.1. Refinamiento del Product Backlog

El *Product Backlog* no ha sufrido ninguna modificación tras haber sido revisado.

### 5.2.2. Planificación del sprint

En este *sprint* se ha realizado la historia de usuario 1 y 2, que podemos ver en la Tabla 6 y Tabla 7. La descomposición en tareas podemos apreciarla en la Tabla 8.

| **Historia de usuario** | | |
|---|---|---|
| **Número:** 1 | **Sprint:** 1 | **Esfuerzo:** 40 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Diseño de la arquitectura | | |
| **Descripción:** Se deben estudiar las tecnologías que demanda la empresa y diseñar una arquitectura completa con ellas. | | |
| **Postcondición:** Se debe presentar una arquitectura completamente funcional y válida. | | |

Tabla 6: Historia de usuario 1 – Diseño de la arquitectura

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 2 | **Sprint:** 1 | **Esfuerzo:** 24 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Diseño del entorno de trabajo | | |
| **Descripción:** Se debe diseñar un entorno de trabajo que permite el desarrollo de la idea del proyecto. | | |
| **Postcondición:** Se debe presentar un diseño de entorno operativo y funcional. | | |

Tabla 7: Historia de usuario 2 – Diseño del entorno de trabajo

| Nombre | Estimación | Historia de Usuario | Prioridad |
|---|---|---|---|
| Estudio y análisis de Oracle APEX | 4 h | 1 | Alta |
| Estudio y análisis de SSH | 1 h | 1 | Baja |
| Estudio y análisis de bases de datos Oracle | 3 h | 1 | Alta |
| Estudio y análisis de Oracle Linux | 2 h | 1 | Media |
| Estudio de sinergia entre tecnologías | 10 h | 1 | Alta |
| Diseño de arquitectura de red | 10 h | 1 | Alta |
| Diseño de arquitectura de ejecución | 40 h | 1 | Alta |
| Estudio y análisis de herramientas de virtualización (VirtualBox y Vagrant) | 6 h | 2 | Media |
| Selección de máquinas virtuales para el entorno | 2 h | 2 | Media |
| Diseño de sistema para simulación de entorno real | 16 h | 2 | Alta |
| **Total** | 64 h | | |

Tabla 8: Tareas asociadas al sprint 1

### 5.2.3. Desarrollo del sprint

Para entender bien el funcionamiento del sistema que se va a desplegar debemos dejar completamente definida la arquitectura del sistema, comprender que roles son los que intervienen y cómo interactúan entre ellos. Por ello, a continuación, se explicarán dos tipos de arquitecturas, por un lado, se explicará la arquitectura de red que describe quienes son los diferentes roles que intervienen a nivel de red y como es la conexión entre estos. En el otro lado tenemos la arquitectura a nivel de ejecución donde se describe cuáles son los diferentes procesos que intervienen en la ejecución del sistema y cuál es el cometido de cada uno junto con las relaciones de dependencia entre ellos.

Se hará uso de una serie de diagramas que ayuden al lector a comprender de forma más sencilla el funcionamiento del sistema global.

## Arquitectura de red

Recordando de forma breve el propósito de este proyecto, lo que pretendemos es gestionar nuestra nube privada mediante la posibilidad de crear bases de datos en ésta de forma sencilla e intuitiva y de forma completamente transparente al usuario.

Empecemos por definir que es la nube privada, entendamos la nube privada como una granja de servidores de gran capacidad y potencia que mediante hipervisores nos permite virtualizar una serie de sistemas heterogéneos (o no), en nuestro caso diferentes versiones de Oracle Linux, exactamente Oracle Linux en sus versiones 6 y 7.

En la Ilustración 11 podemos ver cuál es la idea básica detrás del proyecto, tenemos un sistema externo (o interno) a la nube que se encarga de administrarla, entendiendo administrar como la posibilidad de instalar el software de bases de datos en dichos servidores y de la posterior creación de bases de datos.


Ilustración 11: Diagrama sistema básico

Una vez que tenemos claro el concepto general del proyecto entremos un poco más en detalle en los diferentes roles que podemos encontrar en el sistema global que se ha propuesto.

Contamos con los siguientes sistemas y sus respectivos roles.

- **Sistema gestor/administrador** – Es el que toma la iniciativa en la comunicación en un esquema maestro-esclavo dentro del sistema. Desencadena una serie de cambios en el sistema destino cuando el cliente realiza alguna de las acciones disponibles.

Este sistema puede estar dentro o fuera de la nube privada, pero debe tener acceso a ésta.

- **Sistema destino** – Es el que recibe las ordenes/acciones del sistema gestor, es el esclavo en un esquema maestro-esclavo. Este nunca inicia la comunicación, solo se activa cuando recibe acciones por parte del sistema administrador. Está incluido dentro de la nube privada y se entiende como un único host ya sea virtualizado o *baremetal*[35].

En la Ilustración 12 podemos ver un esquema del sistema un poco más completo. Analizamos en más detalle, en la parte superior tenemos el sistema administrador en el cuál se encuentra Oracle APEX que provee de la interfaz web que permite al usuario tanto la instalación del software de bases de datos como la creación de las mismas. El acceso a este sistema puede ser restringido por el administrador de turno. Más abajo tenemos lo que llamamos el *APEX Private Cloud* que está formado por una granja de servidores en los cuáles pueden tener o no software alguno instalado, otros con únicamente el software de bases de datos instalado pero ninguna base de datos o por otro lado servidores que ya cuentan tanto con el software de bases de datos como con diferentes bases de datos ya creadas (representadas por los cilindros de diferentes colores).

[35] Ejecución de un sistema sin uso de un hipervisor.

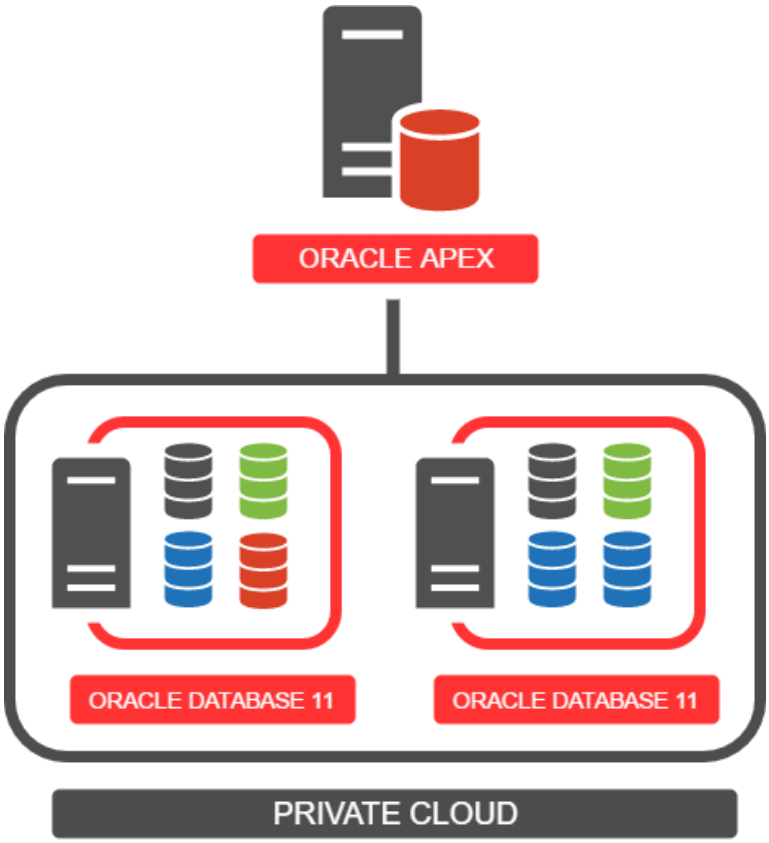


Ilustración 12: Diagrama sistema avanzado

Ahora que tenemos más claro cómo cual es la arquitectura de la red pasemos a ver cómo es la arquitectura de ejecución del sistema.

## Arquitectura de ejecución

Una vez que tenemos claro que es lo que queremos hacer veamos cómo lo hacemos. Para ello vamos a abordar el problema desde un punto de vista con un gran nivel de abstracción e iremos introduciendo cada vez más detalles conforme vayamos añadiendo capas al sistema.

Empezamos por el concepto más básico, tenemos un sistema que debe conectarse a otro de forma remota y provocar cambios en éste mediante la ejecución de determinados comandos y el copiado de archivos, todo ello a través de la red y de forma completamente automática. Para este cometido contaremos con el programa SSH que nos permite ejecutar comandos en un sistema de forma remota sin necesidad de hacer *login*, siempre que hayamos realizado antes las configuraciones previas necesarias.

No queremos que el usuario tenga que ejecutar nada por línea de comandos por lo que debemos abordar el problema de como ejecutar esas sentencias de SSH como si lo hiciéramos a través del terminal como si nos encontráramos conectados directamente al sistema destino. Por ello vamos a recurrir a Java. Cada aplicación Java cuenta con una única instancia de la clase Runtime que permite a la aplicación que hemos desarrollado interactuar con el entorno en el cual está corriendo. El entorno actual puede ser obtenido a través del método getRuntime(), una vez que hemos obtenido el entorno podemos ejecutar el proceso SSH con su respectivo comando a través del método exec() que está contenido en la clase Runtime que hablábamos antes, que devuelve un objeto de la clase Process la cual nos permite obtener el *output* del proceso, que tendrá mucha importancia en nuestro sistema ya que debemos ser capaces de obtener información del sistema remoto tales como valores de los parámetros del kernel, memoria disponible, paquetes instalados… También nos permite detenernos en la ejecución del código hasta que el proceso haya terminado, obtener el valor de retorno del proceso o simplemente terminarlo en un momento dado de la aplicación. Con estas características podremos tener todo lo bueno de ejecutar procesos externos al código Java, pero también lo bueno que ofrece un lenguaje de programación como es Java, teniendo un control total sobre la línea de ejecución del sistema a lo largo del tiempo.

Llegados a este punto debemos aproximarnos más al lado usuario final y debemos ver como permitir la conexión entre la interfaz web (sencilla, intuitiva y transparente) y la ejecución del código Java mencionado anteriormente.

Para entender cómo vamos a abordar veamos la Ilustración 13. Esta imagen muestra lo que está almacenado en la base de datos Oracle 12 del sistema administrador.

- **Oracle APEX**: se ejecuta como una fina capa sobre la base de datos Oracle, de esta manera nuestra aplicación no requiere de scripts complejos para ejecutarse. Oracle APEX ha sido la herramienta elegida para crear nuestra aplicación que contendrá el asistente de creación de bases de datos. El cliente será el que empiece todo el proceso cuando finalice el asistente de instalación del software de bases de datos.
- ***Procedures* y *Functions***: al igual que en otros lenguajes de programación nos permiten modularizar ciertas funciones que van a repetirse con cierta frecuencia. Siendo la diferencia que un *procedure* no devuelve ningún valor y los *functions* devuelven algo siempre.

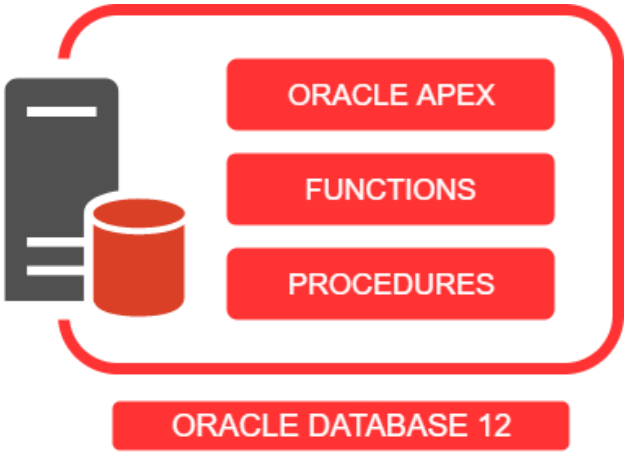


Ilustración 13: Host Administrador

Remarcar que Oracle APEX nos permite ejecutar estos *procedures* a través de sentencias PL/SQL que pueden dispararse con ciertos eventos dentro de la aplicación que contendrá el asistente de instalación.

La conexión más importante viene dada porque podemos ejecutar procedimientos Java que haya almacenados en la base de datos Oracle, lo único que debemos hacer es cargar el código Java que hayamos desarrollado en la base de datos donde se encuentra Oracle APEX y llamar a dichos procedimientos a través de sentencias PL/SQL. Para ello debemos hacer un mapeo entre los procedimientos del código Java y los procedimientos propios de la base de datos. Dichos mapeos deben concordar tanto en tipo de datos, como un número de argumentos o número de elementos devueltos por la función. En la Ilustración 14 podemos ver cómo serían las capas de la aplicación y cada una de sus funciones.

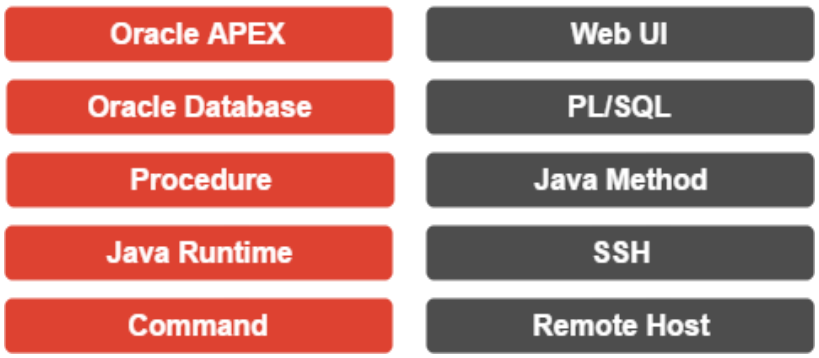


Ilustración 14: Capas de ejecución

Por ejemplo, definimos una clase Java llamada Avanttic

```
public class Avanttic
{
      public static String hola_mundo ()
      {
            return "¡Hola avanttic!";
      }
}
```

Ilustración 15: Avanttic.java

se mapearía en

```
CREATE OR REPLACE FUNCTION hola_mundo RETURN VARCHAR2 AS
LANGUAGE    JAVA    NAME    'Avanttic.hola_mundo    ()    return
java.lang.String';
```

Ilustración 16: Código PL/SQL

Observar como el tipo de dato *VARCHAR2* perteneciente a la base de datos es equivalente al *Jaa.lang.String*.

Lo que esto nos permite es ejecutar el método *hola_mundo* de la clase Java a través de la base de datos mediante una sentencia PL/SQL en Oracle APEX. Primeramente, cargamos la clase Avanttic en la base de datos, donde se almacena como un *Java schema object* que permite que cuando llames al método hola_mundo() mediante PL/SQL, la máquina virtual

de Oracle (JVM) se encarga de localizar las clases Java necesarias, como en este caso la clase *String*. Como hemos comentado existe un mapeo de tipos de datos entre los tipos de Java y los tipos de la base de datos, algunos de ellos, los cuales nos conciernen son:

- java.lang.String y VARCHAR2
- java.lang.Integer y NUMBER
- oracle.sql.ARRAY y VARRAY

Esto nos permitirá que el flujo de información entre la interfaz de la aplicación y el código que se ejecutará de forma remota sea correcto y no se corrompa. Por comodidad, cargaremos el código Java a través del SQL Developer.

Si vemos la cadena final de comunicación sería algo como lo que vemos en la Ilustración 17, que define cuál sería el flujo de ejecución del sistema tomando como referencia temporal el eje horizontal. Decir que la comunicación es bidireccional durante toda la aplicación y que pasas por las mismas capas tanto en una dirección como en otra.


Ilustración 17: Arquitectura de ejecución

Comprendidas la serie de operaciones que son necesarias para llegar desde el usuario final al sistema destino se explicará cómo es el entorno de trabajo sobre el cual se ha desarrollado el proyecto.

# Entorno de trabajo

A continuación, se detalla cuál va ha sido el entorno de trabajo sobre el que se ha desarrollado todo el proyecto

## Sistema global

Buscamos que nuestro proyecto funcione sobre un entorno real y productivo. Para ello, en el desarrollo de las pruebas del proyecto se ha simulado un entorno real a través de sistemas virtualizados como son VirtualBox y Vagrant. VirtualBox se encargará de realizar la función de hipervisor, Vagrant por su parte lo que nos permitirá es levantar sistemas operativos completamente funcionales como si de una "instalación limpia" se tratara cada vez de que lo deseemos, ahorrándonos tiempos de instalación en un momento inicial y permitiéndonos iniciar los sistemas desde cero en el momento que lo deseemos. Esto es muy deseable cuando queramos realizar pruebas sobre entornos que no han sufrido modificaciones y de los cuales sabemos a priori como son, qué contienen o como se comportarán bajo determinados elementos. Supondría un problema y un lastre que cada vez que quisiéramos realizar una prueba de, por ejemplo, instalación del software de bases de datos tuviéramos que realizar la instalación manual de todo el sistema operativo en su totalidad, por eso es tan necesario el uso de Vagrant para este proyecto.

Como hemos comentado previamente hemos contamos con un único sistema físico real, un portátil proporcionado por avanttic que se encargará de soportar toda la virtualización para la simulación del entorno real. Hemos contado con 3 sistemas diferentes, 3 máquinas virtuales corriendo sobre VirtualBox cuyas características se explican más adelante. Dos de estas máquinas han representado nuestro *private cloud*, aunque físicamente se encontrara en la misma máquina. Los 3 sistemas se han encontrado en la misma red y tienen conexión a internet a través del host anfitrión, aunque en un entorno real esto podría no ser recomendable por motivos de seguridad.

Para simular una mayor heterogeneidad del sistema se ha elegido instalar diferentes versiones del sistema operativo de Oracle Linux en los sistemas del *private cloud*, contamos tanto con la versión 6 como la versión 7.

La arquitectura de red final la podemos ver en la **¡Error! No se encuentra el origen de la referencia.**Ilustración 18.

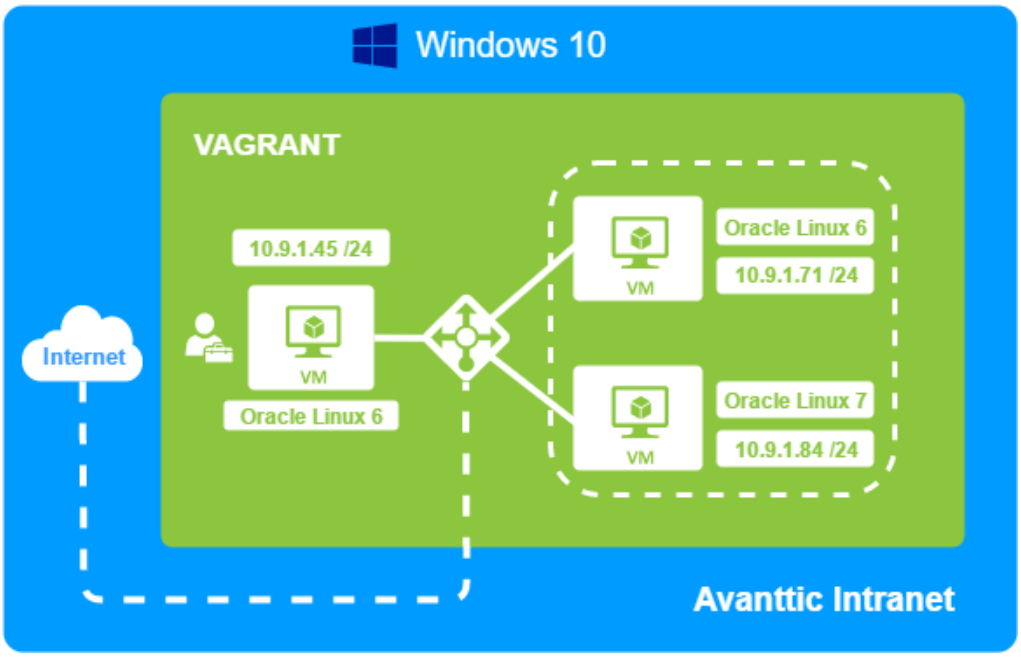


Ilustración 18: Arquitectura de red del sistema

## Roles

A continuación, se describirán las características de los componentes del sistema y las diferentes funciones que han cumplido cada una de las máquinas virtuales desplegadas para simular el entorno real.

### *Administrador*

El rol de administrador (o maestro) es el encargado de iniciar los procesos en el sistema remoto cuando el cliente lo solicite. Este contiene el software de Oracle APEX (sobre la base de datos Oracle) y tiene características más notables que el resto de roles para dar soporte a estas tecnologías. Sus características son las siguientes.

- Oracle Linux 6
- 2 procesadores
- 10 GB RAM
- 2 procesadores
- IP 10.9.1.45/24
- Oracle APEX 5.1
- 125 GB disco

***Esclavo - Oracle 6***

El siguiente rol hace la función de esclavo en el sistema. Se encarga de recibir las acciones propiciadas por el administrador y de ejecutarlas.

- Oracle Linux 6
- 2 procesadores
- 6 GB RAM
- IP 10.9.1.71/24
- 64 GB disco

***Esclavo - Oracle 7***

El siguiente rol hace la función de esclavo en el sistema. Se encarga de recibir las acciones propiciadas por el administrador y de ejecutarlas.

- Oracle Linux 7
- 2 procesadores
- 6 GB RAM
- IP 10.9.1.84/24
- 64 GB de disco

### 1.3.4. Scrum diario

Durante este *sprint* se ha realizado contacto diario entre miembros del equipo, en su mayoría de forma telemática para el planteamiento de dudas y su resolución para permitir un avance en el progreso del proyecto. La comunicación ha sido más intensa debido al empuje inicial que es necesario al enfrentarse a nuevas tecnologías y al inicio del proyecto. Se han desarrollado diferentes charlas que han dotado al equipo de desarrollo de diferentes conocimientos sobre bases de datos que han sido de utilidad para una mayor rapidez en el desarrollo del *sprint*.

### 1.3.5. Revisión del sprint

A la finalización de este *sprint* se ha realizado una reunión entre los diferentes miembros del grupo y otros interesados para comprobar que la arquitectura desarrollada era apropiada para el proyecto. No se han realizado cambios en el *Product Backlog*.

### 1.3.6. Retrospectiva del sprint

Tras la reunión de revisión se ha realizado una nueva reunión con el objetivo de evaluar nuevamente la arquitectura presentada y comprobar su proyección en el proyecto. Se ha obtenido el visto bueno, aunque se han comentado ciertas limitaciones que no afectan al alcance del proyecto.

## 5.3. Sprint 2: Preparación del entorno

Según el plan establecido durante este *sprint* se han llevado a cabo las tareas asociadas a la historia de usuario 3. En esta historia de usuario se han realizado la preparación del entorno de trabajo sobre el cuál se llevará a cabo el proyecto en su totalidad, esto se realizará mediante el levantamiento de máquinas virtuales a través de la herramienta Vagrant y su posterior configuración, como por ejemplo el copiado de las claves SSH entre los diferentes sistemas junto con la configuración de red de las diferentes máquinas.

### 5.3.1. Refinamiento del Product Backlog

El *Product Backlog* no ha sufrido ninguna modificación tras haber sido revisado.

### 5.3.2. Planificación del sprint

En este *sprint* se ha realizado la historia de usuario 3, que podemos ver en la Tabla 9. La descomposición en tareas podemos apreciarla en la Tabla 10.

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 3 | **Sprint:** 2 | **Esfuerzo:** 32 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Despliegue de entorno de trabajo | | |
| **Descripción:** Se debe desplegar y dejar operativo el entorno de trabajo para el desarrollo del proyecto. | | |
| **Postcondición:** El entorno de trabajo debe estar completamente operativo y funcionando | | |

Tabla 9: Historia de usuario 3 – Despliegue de entorno de trabajo

| Nombre | Estimación | Historia de Usuario | Prioridad |
|---|---|---|---|
| Instalación VirtualBox | 1 h | 3 | Alta |
| Instalación Vagrant | 1 h | 3 | Alta |
| Despliegue infraestructura virtualizada | 8 h | 3 | Alta |
| Configuración entorno virtualizado | 4 h | 3 | Media |
| Instalación base de datos Oracle y otros softwares Oracle | 10 h | 3 | Alta |
| Instalación y configuración Oracle APEX | 6 h | 3 | Alta |
| Pruebas y refinamiento | 2 h | 3 | Media |
| **Total** | 32 h | | |

Tabla 10: Tareas del sprint 2

### 5.3.3. Desarrollo del sprint

Durante esta fase se ha realizado la instalación de VirtualBox y Vagrant. La forma de instalación de estas dos herramientas ha sido mediante la descarga de su instalador para Windows y su posterior ejecución.

Una vez que tenemos instaladas en nuestro sistema las herramientas de virtualización se ha procedido a desplegar el sistema junto su posterior configuración mediante una serie de pasos.

#### Selección de las máquinas virtuales

Vagrant es una herramienta que a través de VirtualBox (u otro hipervisor) permite levantar máquinas virtuales completamente operativas en tiempos muy bajos. Una de las características claves de Vagrant y que es muy necesaria en este proyecto es que permite destruir y crear máquinas virtuales de forma sencilla a través de imágenes de sistemas operativos que existen a priori denominadas *boxes*. Vagrant cuenta con un repositorio público donde podemos encontrar estas *boxes* que no son otra cosa que imágenes de máquinas virtuales que ya están operativas y que tienen ciertas características, algunas son Debian, Ubuntu o Oracle Linux, otras pueden contener bases de datos instaladas u otro tipo de software. En nuestro caso se han buscado imágenes que tuvieran ya instalado un Oracle Linux 6 y un Oracle Linux 7. Las imágenes elegidas han sido las siguientes:

- rchouinard/oracle-65-x64
- mleonard87/oracle-linux-70-uek-x64

Para poder añadir dichas imágenes a nuestro sistema, es decir, que se ejecuten en nuestro portátil a través de VirtualBox han sido añadidas a Vagrant. Vagrant es una herramienta que trabaja a través de línea de comandos, por ello debemos acceder a un CMD y ejecutar los comandos que vemos en la Ilustración 19: Añadir imágenes a Vagrant

```
C:\Vagrant\Oracle 6> vagrant box add rchouinard/oracle-65-x64
C:\Vagrant\Oracle 7> vagrant box add mleonard87/oracle-linux-
70-uek-x64
```

Ilustración 19: Añadir imágenes a Vagrant

Seguidamente hemos ejecutado el comando de la Ilustración 20, que crea un archivo *Vagrantfile* en el directorio donde nos encontremos, notar que dicho comando se debe ejecutar en diferentes directorios, uno por cada imagen. Este archivo es importante ya que determina la configuración que tomará la máquina virtual cuando se ejecute. Por defecto trae una serie de parámetros de configuración comentados que podemos tomar como referencia. En nuestro caso hemos añadido una configuración, esta opción, que podemos ver en la Ilustración 21, permite que todas las máquinas virtuales estén conectadas a la misma red y que tengan comunicación entre el *guest*, el *host* y entre los propios *guest*. De esta forma hemos asignado la IP de forma estática para que esta no varíe, al igual que haría en un entorno real.

```
C:\Vagrant\Oracle 6> vagrant init rchouinard/oracle-65-x64
```

Ilustración 20: Inicialización imagen Oracle Linux 6

```
config.vm.network "public_network", ip: "10.9.1.85"
```

Ilustración 21: Configuración máquina virtual Oracle Linux 6

Hemos repetido los mismos pasos para la máquina Oracle 7 pero cambiando la dirección IP de la máquina para que no haya problemas. Con la máquina configurada, hemos procedido

a levantarla mediante el comando de la Ilustración 22. Es importante que en el momento de lanzar el comando nos encontremos en el directorio donde se encuentra el *Vagrantfile*. Este comando nos dará una salida por el terminal con una serie de información que analizaremos ahora, esta salida podemos verla en la Ilustración 23.

```
C:\Vagrant\Oracle 6> vagrant up
```

Ilustración 22: Levantamiento máquina virtual Oracle Linux 6

No mostraremos toda la información si no que nos centraremos en aquella información que hemos necesitado en este momento para poder acceder al sistema que acabos de levantar. Las líneas "`SSH address: 127.0.0.1:2222`" y "`SSH username: vagrant`", nos dicen que para conectarnos a esta máquina debemos hacerlo a través del *localhost* del sistema Windows en el que se está ejecutando Vagrant a través del puerto 22, permitiéndonos loguearnos a través del usuario *vagrant*. Para poder acceder al sistema solo hemos tenido que hacer login a través de algún software de SSH, en nuestro caso Xshell.

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'rchouinard/oracle-65-x64' is up
to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on
configuration...
    default: Adapter 1: nat
    default: Adapter 2: bridged
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few
minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the
installed version of
    default: VirtualBox! In most cases this is fine, but in
rare cases it can
    default: prevent things such as shared folders from
working properly. If you see
    default: shared folder errors, please make sure the guest
additions within the
    default: virtual machine match the version of VirtualBox
you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 4.3.6
    default: VirtualBox Version: 5.1
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => C:/Vagrant/Oracle 6
==> default: Machine already provisioned. Run `vagrant
provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked
to run always will still run.
```

Ilustración 23: Despliegue de máquina Oracle Linux 6 en Vagrant

Una vez que tenemos las máquinas operativas hemos pasado a configurar el entorno para que la comunicación a través de SSH. Hemos copiando las claves del sistema administrador en las máquinas Oracle Linux 6 y Oracle Linux 7. Para ello nos hemos conectado mediante

SSH al sistema administrador y hemos realizado los comandos de la Ilustración 24. Hemos introducido la clave del usuario root, en este caso *vagrant* .

```
$ ssh-copy-id root@10.9.1.71
$ ssh-copy-id root@10.9.1.84
```

Ilustración 24: Copiado de claves

La instalación y configuración tanto de la base de datos Oracle como Oracle APEX en el sistema administrador han sido llevadas a cabo por personas técnicas de Avanttic S.L. y no se recogerán en este documento. Una vez fue instalado y configurado el entorno administrador me fue suministrado para el desarrollo del proyecto.

### 5.3.4. Scrum diario

Durante este *sprint* se ha realizado contacto diario entre miembros del equipo, en su mayoría de forma remota para el planteamiento de dudas y su resolución para permitir el avance en el progreso del proyecto.

### 5.3.5. Revisión del sprint

A la finalización de este *sprint* se ha realizado una reunión entre los diferentes miembros del grupo y otros interesados para comprobar si el sistema que se ha montado para el desarrollo del proyecto es válido y si las tecnologías que se habían propuestos cubren todas las necesidades exigidas. Recalcar que el uso de Vagrant no estaba previsto en un inicio, pero fue propuesto por el equipo de desarrollo siendo posteriormente aceptado como herramienta para el proyecto. No se han realizado modificaciones en el *Product Backlog*.

### 5.3.6. Retrospectiva del sprint

Tras la reunión de revisión se ha realizado una nueva reunión con el objetivo de evaluar y probar el sistema que se ha montado para la realización del proyecto. Se ha determinado que el sistema propuesto es válido para el desarrollo del proyecto y que cumple las expectativas esperadas.

# Sprint 3: Lógica de instalación de software de bases de datos

Según lo planificado durante este *sprint* se han llevado a cabo las tareas asociadas a las historias de usuario 4 y 5. En estas historias de usuario se han realizado tanto un análisis sobre cuáles son las formas de llevar a cabo la instalación de bases de datos Oracle haciendo especial hincapié en la instalación por línea de comandos junto con el diseño e implementación del código para permitir una instalación automática y transparente del software de bases de datos Oracle en los sistemas finales a través del *Oracle Universal Installer*. Se preparará el sistema para posteriormente en el siguiente *sprint* desarrollar la interfaz que permita la introducción de la información por parte del usuario.

## 5.4.1. Refinamiento del Product Backlog

El *Product Backlog* no ha sufrido ninguna modificación tras haber sido revisado.

## 5.4.2. Planificación del sprint

En este *sprint* se han realizado las historias de usuario 4 y 5, que podemos ver en la Tabla 11 y la Tabla 12. La descomposición en tareas podemos apreciarla en la Tabla 13.

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 4 | **Sprint:** 3 | **Esfuerzo:** 30 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Análisis para instalación de software de BBDD | | |
| **Descripción:** Se debe estudiar los métodos para instalación de software de datos en especial por línea de comandos (OUI). | | |
| **Postcondición:** Haber adquirido los conocimientos necesarios para la instalación de software de bases de datos por línea de comandos a través del OUI. | | |

Tabla 11: Historia de usuario 4

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 5 | **Sprint:** 3 | **Esfuerzo:** 150 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Implementación de lógica para instalación de software de BBDD | | |
| **Descripción:** Se desarrollar el código Java y Bash necesario para la instalación automática del software de bases de datos. | | |
| **Postcondición:** Código ejecutable y sin errores que permita la instalación automática del software de bases de datos a través del OUI. | | |

Tabla 12: Historia de usuario 5

| Nombre | Estimación | Historia de Usuario | Prioridad |
|---|---|---|---|
| Estudio y análisis de bases de datos Oracle | 10 h | 4 | Alta |
| Estudio de sinergia entre tipos de datos en Java y tipos de datos en bases de datos Oracle | 5 h | 4 | Baja |
| Estudio y análisis de asistente de instalación Oracle Universal Installer | 1 h | 4 | Baja |
| Obtención de requisitos para instalación del software de bases de datos Oracle | | 4 | |
| Diseño estructural del código a desarrollar | 12 h | 5 | Alta |
| Implementación código Bash | 48 h | 5 | Alta |
| Implementación código Java | 40 h | 5 | Alta |
| Creación de procedimientos en base de datos Oracle | 10 h | 5 | Media |
| Pruebas y refinamiento | 40 h | 5 | Alta |
| **Total** | 180 h | | |

Tabla 13: Tareas asociadas al sprint 3

### 5.4.3. Desarrollo del sprint

Para llevar a cabo esta tarea hemos utilizado el *Oracle Universal Installer*, contenido en el software de bases de datos Oracle 11g Release 2 (11.2). Hemos seguido los pasos de instalación en un sistema donde no existía un software de Oracle previamente instalado.

En los siguientes puntos se explican cada uno de los pasos que se han seguido ordenadamente para llevar a cabo la instalación del software de bases de datos de Oracle. Estos pasos van desde la descarga y copia del software en la máquina remota hasta la ejecución de forma automática del instalador con la información proporcionada por el usuario.

# Copiado del software

Inicialmente hemos descargado el software de bases de datos Oracle, éste podemos descargarlo desde la página web del *Oracle Technology Network (OTN)*, en nuestro caso fue proporcionado por Avanttic S.L. Esta descarga la hemos realizado una única vez, ya que se encuentra de forma persistente en un repositorio local del sistema administrador. Después de la descarga del software debemos contar con dos archivos *p13390677_112040_Linux-x86-64_1of7.zip* y *p13390677_112040_Linux-x86-64_2of7.zip*. Una vez descargados los archivos han sido descomprimidos en nuestro repositorio local donde hemos contado con una carpeta llamada *database* que contiene el ejecutable para la instalación del software de bases de datos, el *Oracle Universal Installer*.

Una vez que tenemos nuestro repositorio local preparado hemos realizado el primer paso que es el copiado del software al sistema destino a través de la herramienta *scp*. En la Ilustración 25 vemos el código encargado de esta tarea.

```
Process theProcess = null;
theProcess = Runtime.getRuntime().exec(new String [] {
      "/usr/fin/scp","-r",
      "/home/grid/tfg/oracle11gr2/software/database",
      "root@"+hostname+":/tmp/database"});
theProcess.waitFor();
```

Ilustración 25: Código Java para copiado del Software

Expliquemos el código un poco más en detalle el código, lo primero que explicaremos será la clase *Runtime* de Java, esta clase nos permite que nuestra aplicación interaccione con el entorno en el cuál la aplicación está corriendo. Cada aplicación tiene una única instancia de esta clase y podemos obtenerla a través del método *getRuntime()* de dicha clase. Una vez que hemos accedido al entorno de ejecución podremos ejecutar a través del método *exec()* el comando *scp* que nos permitirá copiar el software en el destino remoto, lo copiaremos a la carpeta temporal para que este se elimine en caso de reinicio del sistema ya que una vez instalado no será necesario de nuevo.

Recalcar también que la clase *Process* cuenta con una serie de métodos que nos permitirán tener un mayor control sobre los procesos que se ejecutarán remotamente, algunos de esos métodos son:

- **exitValue():** Devuelve el valor devuelto por el proceso ejecutado. Esto nos permitirá saber si el comando ejecutado ha sido exitoso o por otro lado ha tenido errores. Por normal general un 0 significa una terminación normal del programa.
- **getInputStream():** Devuelve el *input stream* el cual está conectado a la salida estándar del proceso a ejecutar. Esto nos permitirá obtener la información de salida del proceso.
- **getErrorStream():** Devuelve el *input stream* el cual está conectado a la salida de error del proceso a ejecutar. Esto nos permitirá obtener la información de error que pueda producir el proceso.
- **waitFor():** Causa que el hilo de ejecución espero hasta que el proceso representado por el objeto *Process* haya terminado.

Debemos esperar a que termine la ejecución para asegurarnos que el software se ha copiado correctamente en el sistema antes de continuar. Una vez que tenemos el software de instalación en el sistema destino pasaremos a ver que necesidades debemos cumplir a nivel de sistema para poder instalar el software de bases de datos.

# Instalación de paquetes necesarios

Seguidamente el requisito que debemos satisfacer es la instalación en el sistema destino del paquete *oracle-rdbms-server-11gR2-preinstall* que creará de forma automática los grupos y usuarios junto con los parámetros del kernel necesarios que son necesarios para las instalaciones de software de Oracle. Al tratarse el sistema destino de un Oracle Linux lo hemos instalado a través del gestor de paquetes *yum* como vemos en la Ilustración 26.

```
$ yum install Oracle-rdbms-server-11gr2-preinstall
```

Ilustración 26: Instalación de paquete en Oracle Linux

# Requisitos previos

Antes de poder realizar la instalación del software hemos tenido que realizar una serie de operaciones en el sistema destino para que la instalación sea correcta y funcione. Los requisitos que debemos satisfacer son:

- Creación carpeta */u01/app* en el sistema destino. Contiene todo el software de Oracle.
- Definir como *owner* de la carpeta al usuario o*racle* , encargado de ejecutar el instalador y por tanto debe tener permisos.
- Permisos de ejecución al usuario *oracle*.
- Comprobación que el *hostname* del sistema se encuentra en */etc/hosts*

Previamente se ha realizado una comprobación de si el sistema cumple con los requisitos a nivel de memoria, pero supondremos que los cumple para mayor simplificación, a su vez también es necesario modificar ciertos parámetros del kernel, pero mediante el paquete que hemos instalado previamente cumplimos esta restricción. También decir que posteriormente en el siguiente *sprint* se realizará la lógica de control que comprobará si dichas carpetas existen o no. En la Ilustración 27 se muestra el código fuente que llevará a cabo estas operaciones.

```
Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      "/bin/mkdir -p /u01/app/", })
.waitFor();
Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      "/bin/chown -R oracle:oinstall /u01"})
.waitFor();
Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      "if [ $(/bin/cat /etc/hosts | /bin/grep -c $(hostname))
];" +
      "then /bin/echo "+hostname+" $(hostname) >> /etc/hosts;
fi"})
.waitFor();
Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      "/bin/chmod -R 775 /u01"})
.waitFor());
```

Ilustración 27: Código Java - requisitos previos

# Instalación del software

Ahora que ya tenemos el software en el sistema destino y que hemos comprobado que cumplimos todos los requisitos necesarios para la instalación procederemos a ver cómo podemos obtener la información necesaria para el proceso, que tipo de información y qué importancia tiene de cara al estado final del software en el sistema. Empezaremos por la creación del *Response File* que permite la automatización del proceso de instalación, posteriormente veremos cómo añadir esta información al instalador por línea de comandos y finalmente ejecutaremos *scripts* de post-instalación para dejar el sistema completamente operativo.

# Preparación del Response File

Nosotros podemos automatizar la instalación y configuración del software de Oracle, en su totalidad o parcialmente, está automatización total es la que nosotros estamos buscando. El *Oracle Universal Installer* usa un archivo denominado *Response File*, el cual contiene una serie de información del tipo clave-valor, información que el instalador nos solicitará de forma interactiva en caso de no habérsela proporcionado.

Normalmente el *Oracle Universal Installer* se ejecuta de forma interactiva lo cual significa que te solicita la información a través de una interfaz gráfica como la de la Ilustración 28.

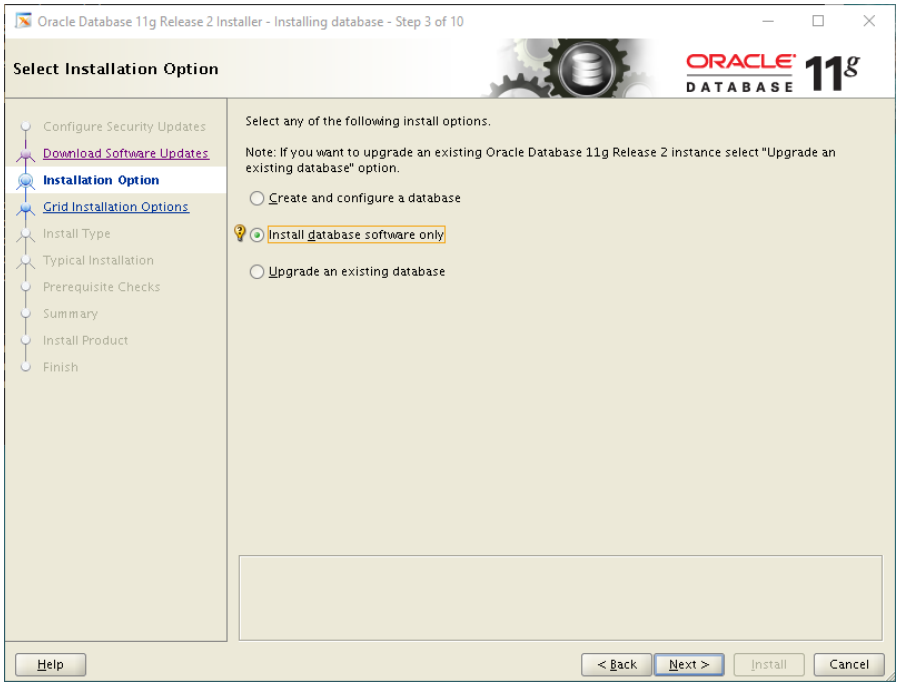


Ilustración 28: Interfaz Oracle Universal Installer

Cuando nosotros utilizamos un *Response File* para proporcionar esta información ejecutamos el *Oracle Universal Installer* a través de un terminal de comandos pasándole como argumento dicho *Response File*, que hemos creado y rellenado previamente, a su vez añadiremos también la opción *silent* que hace que el instalador no muestre ningún tipo de pantallazo, esto es útil cuando

- Queremos automatizar el proceso de instalación sin ningún tipo de interacción.
- Completar diferentes instalaciones en diferentes sistemas sin que intervenga el usuario.
- Instalar en un sistema que no tiene un sistema de ventanas X.

Que son las características que cumple nuestro sistema.

Hemos analizado la información que contiene un *Response File* y hemos visto que información podemos dejar por defecto evitando así al usuario tener que introducirla y cuál tendremos que solicitar que ingrese por ser información crítica. Posteriormente crearemos el asistente de instalación del software que se encargará de recoger dicha información.

Oracle nos proporciona una serie de *Response Files* de ejemplo del tipo *rsp*. El formato de estos archivos lo podemos ver en la Ilustración 29.

```
#------------------------------------------------------------
------
# Specify the installation option.
# It can be one of the following:
#   - INSTALL_DB_SWONLY
#   - INSTALL_DB_AND_CONFIG
#   - UPGRADE_DB
#------------------------------------------------------------
------
oracle.install.option=INSTALL_DB_SWONLY
```

Ilustración 29: Ejemplo Response File

El archivo está formado por una serie de líneas del tipo *CLAVE=VALOR* donde las claves están predefinidas y su valor solo en ocasiones. También tenemos líneas de comentarios que comienzan por # y que muestran información explicativa sobre los tipos de valores que pueden tomar cada una de las claves.

Antes de continuar hemos definido una serie de conceptos clave en las bases de datos Oracle y que nos ha permitido entender más fácilmente los valores que debe tomar el *Response File.*

### *Enterprise Edition*

Oracle Database 11g Enterprise Edition es una base de datos autogestionada que tiene características tales como escalabilidad, rendimiento, alta disponibilidad y seguridad. Este tipo de instalación está diseñada para aplicaciones de nivel empresarial. Es capaz de gestionar *OLTP*[36] con seguridad y se desenvuelve bien en entornos de *data warehouse*.

### *Standard Edition*

Oracle Database 11g Standard Edition es una solución para la gestión de datos idealmente creada para la mediana empresa. Está diseñada para proveer un núcleo de gestión de servicios relacionales. Ésta instala una serie de herramientas de gestión permitiendo características como distribución, replicación, características Web y facilidades para construir aplicaciones críticas a nivel empresarial.

### *Standard Edition One*

Oracle Database 11g Standard Edition One es una solución para la gestión de datos idealmente creada para las necesidades de la pequeña empresa. Esta opción es elegida para departamentos dentro de una misma empresa o aplicaciones web.

Ahora comprendamos algunos valores que el usuario tendrá que introducir y que tienen vital importancia en la instalación del software.

### *Oracle Base*

Especifica una ruta donde almacenar todo el software Oracle y los archivos de configuración relacionados. Por defecto este valor es *mountpoint/app/user* donde *user* es el usuario que está ejecutando la instalación.

### *Oracle Home*

[36] *On-Line Transaction Processing*

Especifica la localización para almacenar los archivos del software separados de los archivos de configuración en el *Oracle Base*. Este debe estar contenido dentro del *Oracle Base*. En nuestra instalación el Oracle Home vendrá definido por el *Software Location*

### *SYSDBA y SYSOPER*

SYSDBA y SYSOPER son privilegios que son requeridos para crear una base de datos usando la autenticación del sistema operativo. Aquellos que sean miembros del grupo OSDBA tendrán privilegios SYSDBA y los miembros de OSOPER tendrán privilegios SYSOPER, que es un subconjunto de privilegios de SYSDBA. El usuario que ejecuta la instalación debe ser miembro de estos grupos.

### *OraInventory*

El OraInventory o *Oracle Inventory* es la ruta donde se almacena toda la información referente a

- Todos los productos software de Oracle instalados en todos los *Oracle Homes* de la máquina.
- Otros productos que no son de Oracle como los entornos de Java Runtime (JRE).

Una vez comprendidos estos conceptos veamos en más detalle cuales son estás posibles claves y los valores que pueden tomar los diferentes atributos del *Response File*. Debemos decidir cuales podemos dejar por defecto ahorrándole trabajo al usuario final y facilitándoselo

### *Oracle.install.responseFileVersion*

Define la versión y tipo de *Response File.* Define como será el contenido que contiene el archivo y que formato tendrá. Este valor será por defecto con valor */oracle/install/rspfmt_dbinstall_response_schema_v11_2_0.*

### *Oracle.install.option*

Especifica qué tipo de instalación será llevada a cabo con el asistente. Permite 3 opciones:

- Instalar únicamente el software de bases de datos.
- Instalar el software de bases de datos y crear una base de datos.
- Actualizar una base de datos.

En nuestro caso será la primera opción y su valor será también por defecto siendo *INSTALL_DB_SWONLY*. Posteriormente deberíamos ejecutar el *Oracle Database*

*Configuration Assistant* para crear las bases de datos, lo cual se cubrirá en la siguiente iteración.

### *Oracle_hostname*

Especifica el *hostname* del sistema que será usado durante la instalación.

### *Unix_group_name*

Especifica el grupo Unix para el directorio del *inventory.*

### *Inventory_location*

Especifica la localización del directorio que almacena los archivos del *inventory*.

### *Selected_languages*

Especifica el lenguaje que será instalado en los diferentes componentes.

### *Oracle_home*

Especifica la ruta completa del *Oracle Home.*

### *Oracle_base*

Especifica la ruta completa del *Oracle Base*.

### *Oracle.install.db.InstallEdition*

Especifica la edición para la instalación. Puede tomar varios valores.

- Enterprise Edition (EE)
- Standard Edition (SE)
- Standard Edition One (SEONE)

Obviamos la versión *Personal Edition* por ser únicamente válida en Windows. El usuario final podrá elegir qué edición quiere instalar.

### *Oracle.install.db.DBA_GROUP*

Especifica el grupo del sistema operativo el cual tendrá privilegios OSDBA.

### *Oracle.install.db.OPER_GROUP*

Especifica el grupo del sistema operativo el cual tendrá privilegios OSOPER.

### *MYORACLESUPPORT_USERNAME*

Especifica el usuario de la cuenta de *My Oracle Support*. En nuestro caso no tomará ningún valor.

### *MYORACLESUPPORT_PASSWORD*

Especifica la contraseña de la cuenta de *My Oracle Support*. En nuestro caso no tomará ningún valor.

### *SECURITY_UPDATES_VIA_MYORACLESUPPORT*

Especifica si activar el usuario para asignar la contraseña para las credenciales de *My Oracle Support*. En nuestro caso no tomará ningún valor que equivale a *false*.

### *DECLINE_SECURITY_UPDATES*

Especifica si el usuario no quiere configurar las actualizaciones de seguridad. El valor de esta variable debe ser *true* si no quieres configurar las actualizaciones de seguridad. En nuestro caso el valor por defecto será *true*.

### *PROXY_HOST, PROXY_PORT, PROXY_USER, PROXY_PWD, PROXY_REALM*

Especifica la información necesaria para el uso de un proxy. En nuestro caso no se hará uso de un proxy por lo que dichos atributos no tomarán ningún valor.

### *COLLECTOR_SUPPORTHUB_URL*

Especifica la URL del *Oracle Support Hub*. En nuestro caso no tomará ningún valor.

### *Oracle.installer.autoupdates.option*

Especifica la opción para las actualizaciones automáticas. Puede tomar los siguientes valores.

- MYORACLESUPPORT_DOWNLOAD
- OFFLINE_UPDATES
- SKIP_UPDATES

En nuestro caso evitaremos las actualizaciones automáticas por ello tomará por defecto el valor *SKIP_UPDTES.*

Ahora que tenemos claro que tipo de información tenemos que suministrar al instalador de Oracle veremos cómo lo hemos ejecutado de forma automática y transparente.

# Ejecución del instalador

Analicemos un poco más en detalle este código ya que es el más importante dentro de esta iteración. Este código es el que se encarga de ejecutar todas las secuencias necesarias para la instalación del software, desde el cumplimiento de los pre-requisitos, la ejecución del instalador y los requisitos post-instalación, por ello vamos a ver que secuencia de operaciones lleva a cabo para entenderlo. El código fuente podemos verlo en la Ilustración 30 (PL/SQL en Ilustración 31).

1. Copia en el sistema destino el software de instalación mediante *scp*.
2. Instala el paquete *Oracle-rdbms-server-11gR2-preinstall* que se encarga de crear los usuarios y grupos necesarios para la base de datos, así como asignar los parámetros del kernel.
3. Crea los directorios donde se va a desplegar el software Oracle.
4. Les asigna los permisos del directorio al usuario o*racle* que será el que lleve a cabo la instalación.
5. Comprueba que el *hostname* del equipo esté dentro el archivo */etc/hosts*, de no ser así lo añade.
6. Ejecuta el instalador *runInstaller* con el usuario *oracle* y con los atributos que ha obtenido del usuario a través de los argumentos del método.
7. Ejecuta los *scripts* de post-instalación como usuario *root*.
8. Añade a las variables de entorno las variables referentes a la base de datos.
9. Devuelve un mensaje de éxito o error según corresponda.

Una vez finalizado el proceso el usuario final verá un mensaje verde que le comunicará que la instalación del software ha sido un éxito.

```
public static String installSoftware (String oracleBase, String
oracleHome, String attrs, String hostname) {
  Process theProcess = null;
  BufferedReader inStream = null;
  int exitValue;
  String line = null, logFilePath;


   try {

   /** Copiamos el software al sistema destino **/
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/scp","-r",
     "/home/grid/tfg/oracle11gr2/software/database",
     "root@"+hostname+":/tmp/database"});
   theProcess.waitFor();

   /** Instalamos el paquete con los pre-requisitos para instalar
el software de Oracle **/
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "/usr/bin/yum install oracle-rdbms-server-11gR2-preinstall -
y"});
   theProcess.waitFor();

   /** Creamos los directorios donde se va a despegar el software
de Oracle ** TODO: Comprobar si existen previamente **/
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "/bin/mkdir -p /u01/app/", });
   theProcess.waitFor();

   /** Cumplimos los pre-requisitos **/
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "/bin/chown -R oracle:oinstall /u01"});
   theProcess.waitFor();

   Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "if [ $(/bin/cat /etc/hosts | /bin/grep -c $(hostname)) ];"
+
     "then /bin/echo "+hostname+" $(hostname) >> /etc/hosts; fi"})
   .waitFor();
```

```
   Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "/bin/chmod -R 775 /u01"})
   .waitFor();


   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "su - oracle -c",
     "\"/tmp/database/runInstaller -silent " + attrs + "\"" });
   theProcess.waitFor();

   // Que ejecute como root los script que dice
   Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     oracleBase+"/oraInventory/orainstRoot.sh" })
   .waitFor();

   Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     oracleHome+"/root.sh" })
   .waitFor();

   String oracleSettings [] = {
     "# Oracle Settings",
     "TMP=/tmp; export TMP",
     "TMPDIR=$TMP; export TMPDIR",
     "ORACLE_HOSTNAME="+hostname+"; export ORACLE_HOSTNAME",
     "ORACLE_UNQNAME=DB11G; export ORACLE_UNQNAME",
     "ORACLE_BASE="+oracleBase+"; export ORACLE_BASE",
     "ORACLE_HOME="+oracleHome+"; export ORACLE_HOME",
     "ORACLE_TERM=xterm; export ORACLE_TERM",
     "PATH=$ORACLE_HOME/bin:$PATH; export PATH",
     "LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib;      export
LD_LIBRARY_PATH",

"CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms
/jlib; export CLASSPATH"
     };

    for (String attr : oracleSettings )
    {
```

```
      theProcess = Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      String.format("/bin/grep              -E              \"%s\"
/home/oracle/.bash_profile", attr)});
      exitValue = theProcess.waitFor();

      if ( exitValue != 0 )
      {
       theProcess = Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      "/bin/bash -c",
      "\"/bin/echo      '"      +      attr      +      "'      >>
/home/oracle/.bash_profile\""});
       theProcess.waitFor();
      }
     }

    return "El software se ha instalado correctamente.";

   } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    return "Ha habido algún error en el proceso.";
   }
  }
```

Ilustración 30: Código Java - installSoftware()

```
create or replace FUNCTION INSTALLSOFTWARE (oraclebase
VARCHAR2, oraclehome VARCHAR2, attrs VARCHAR2, hostname
VARCHAR2) RETURN VARCHAR2 AS
LANGUAGE JAVA NAME
'avanttic.Oracle11Release2.installSoftware(java.lang.String,
java.lang.String, java.lang.String, java.lang.String) return
java.lang.String';
```

Ilustración 31: Código PL/SQL - installSoftware()

# Depuración

Se ha desarrollado un pequeño método para permitir la depuración del sistema, el método podemos verlo en la Ilustración 32 (código PL/SQL en Ilustración 33), su función es la de recibir por argumentos un comando Shell determinado por el usuario y de ejecutarlo

de forma remota devolviendo su salida para la comprobación de si el sistema se comporta como se espera bajo determinadas acciones. Este método nos permite realizar pruebas previas a la implementación del software haciendo así más rápido su desarrollo.

```
public static String depurar (String command) {
   Process theProcess = null;
   String output = "", path="/u01/", line;
   BufferedReader inStream = null;
   int returnedValue;

  try {
   theProcess = Runtime.getRuntime().exec(command);
   returnedValue = theProcess.waitFor();

output.concat(Integer.toString(returnedValue)).concat("\n");
   inStream = new BufferedReader(
     new InputStreamReader(theProcess.getInputStream()));

   while ( (line = inStream.readLine()) != null ) {
    output = output.concat(line).concat("\n");
   }

   inStream = new BufferedReader(new InputStreamReader(
     theProcess.getErrorStream()));

   while ( (line = inStream.readLine()) != null ) {
    output = output.concat(line).concat("\n");
   }
   //inStream = new BufferedReader(
     new InputStreamReader(theProcess.getInputStream()));

  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   return e.getMessage();
  }

  System.out.println(output);
  return output;
 }
```

Ilustración 32: Código Java - depurar()

```
create or replace FUNCTION depurar (COMMAND VARCHAR2) RETURN
VARCHAR2 AS
LANGUAGE    JAVA    NAME    'avanttic.Oracle11Release2.depurar
(java.lang.String) return java.lang.String';
```

Ilustración 33: Código PL/SQL - depurar()

### 5.4.4. Scrum diario

Durante este *sprint* se ha realizado contacto diario entre miembros del equipo, en su mayoría de forma telemática para el planteamiento de dudas y su resolución para permitir un avance en el progreso del proyecto.

### 5.4.5. Revisión del sprint

A la finalización de este *sprint* se ha realizado una reunión entre los diferentes miembros del grupo y otros interesados para comprobar si la interfaz desarrollada es idónea para cumplir con el objetivo que se había propuesto y si es necesario añadir o eliminar elementos de la interfaz. No se han realizado modificaciones en el *Product Backlog.*

### 5.4.6. Retrospectiva del sprint

Tras la reunión de revisión se ha realizado una nueva reunión con el objetivo de evaluar y probar el código desarrollado. Se ha determinado que el sistema cumple con las expectativas acordadas y que su funcionamiento es el esperado.

## 5.5. Sprint 4: Interfaz de instalación de software de bases de datos

De acuerdo con el plan de proyecto durante este *sprint* se han realizado las tareas asociadas a la historia de usuario 6. En esta historia de usuario se ha llevado a cabo el diseño e implementación de la interfaz para la instalación del software de bases de datos mediante un asistente creado a través de Oracle APEX que permitirá al usuario introducir la información necesaria para la instalación del software.

### 5.5.1. Refinamiento del Product Backlog

El *Product Backlog* no ha sufrido ninguna modificación.

### 5.5.2. Planificación del sprint

En este *sprint* se ha realizado la historia de usuario 6, que podemos ver en la Tabla 14. La descomposición en tareas podemos apreciarlo en la Tabla 15.

| **Historia de usuario** | | |
|---|---|---|
| **Número:** 6 | **Sprint:** 4 | **Esfuerzo:** 40 h |
| **Prioridad**: Media | **Rol:** Equipo | |
| **Nombre:** Diseño e implementación de interfaz para instalación de bases de datos | | |
| **Descripción:** Se debe desarrollar una interfaz a través de Oracle APEX que permita la instalación del software de bases de datos de forma remota. | | |
| **Postcondición:** La existencia de un asistente de instalación funcional que permita instalar software de bases de datos de forma remota. | | |

Tabla 14: Historia de usuario 6

| Nombre | Estimación | Historia de Usuario | Prioridad |
|---|---|---|---|
| Diseño de los pasos que seguirá el asistente a desarrollar | 6 h | 6 | Alta |
| Implementación de los pasos del asistente de instalación | 10 h | 6 | Alta |
| Dotación de lógica de control al asistente de instalación | 14 h | 6 | Alta |
| Pruebas y refinamiento | 10 h | 6 | Alta |
| **Total** | 40 h | | |

Tabla 15: Tareas asociadas al sprint 4

### 5.5.3. Desarrollo del sprint

Para la obtención de la información para instalar el software de bases de datos necesitaremos que el usuario introduzca de forma manual cierta información crítica pero trivial. Para la comodidad del usuario se ha creado un asistente de instalación que lo guiará a lo largo del proceso de instalación mediante una interfaz sencilla, simple y entendible. Esta interfaz se ha realizado a través de Oracle APEX. Se ha intentado simplificar lo máximo posible el proceso de instalación para el usuario, que éste tenga que introducir únicamente la información más sencilla y que el resto venga por defecto para permitir una instalación rápida, simple y transparente. A su vez se ha realizado una lógica de control sobre lo que introduce el usuario para evitar errores posteriores en la instalación.

Para tener persistencia de los datos desde el inicio del asistente hasta el final se ha creado una tabla *software_only* que está compuesta de una tupla clave-valor, donde la clave es cada uno de los atributos que puede tomar el *Response File* y el valor como su nombre indica el valor de dicho atributo. Esta tabla se inicializará al principio del asistente con valores por defecto que se irán modificando a lo largo del asistente. Esto nos permitirá volver atrás y adelante a lo largo del asistente sin tener que volver a rellenar cada uno de los campos.

A continuación, se detallan y se muestran cada uno de los pasos del asistente de instalación y las distintas funciones que se han desarrollado para corroborar el correcto funcionamiento del programa.

Inicialmente, como vemos en la Ilustración 34, se le solicitará al usuario que elija que edición de base de datos desea instalar.

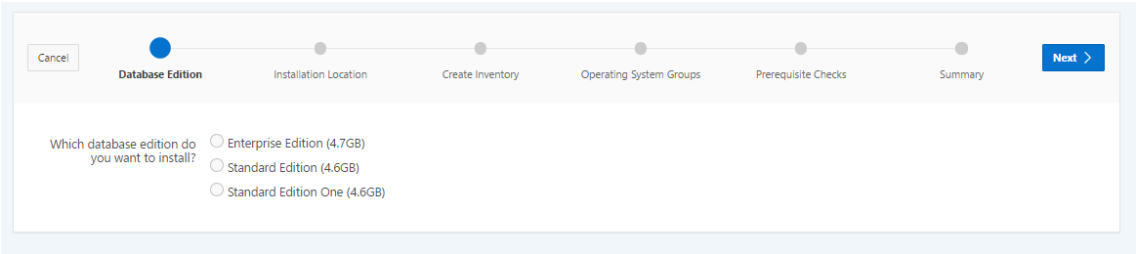


Ilustración 34: Asistente de Instalación - Database Edition

Seguidamente, como vemos en la Ilustración 39, se le pedirá que defina cual será la ruta dentro del sistema en la cual se alojarán tanto el *Oracle Base* tanto el *Oracle Home (Software Location)*. Estas rutas deben seguir una serie de restricciones, las cuales son:

- Ambas rutas deben ser rutas Unix válidas.
- El *Oracle Home* debe estar contenido dentro del *Oracle Base*, no debe existir y en caso de que exista debe ser un directorio vacío.
- El *Oracle Base* no debe existir en el sistema y en caso de existir debe ser un directorio vacío.

Para satisfacer estas restricciones se ha hecho uso tanto de herramientas que provee Oracle APEX como código externo Java. Para la comprobación de que ambas rutas deben ser rutas Unix válidas se ha comprobado el valor introducido con la siguiente expresión regular

```
^(/[^/]+)+/?$
```

En caso de ser positiva la comprobación de los patrones tendremos el visto bueno.

El siguiente paso que comprobar es que el *Oracle Base* sea un directorio válido y que este vacío en caso de existir. Esto lo comprobaremos mediante

```
test -d <path> -a -s <path>
```

incluido dentro del código Java que podemos ver en la Ilustración 35 (mapeo en procedimiento en la Ilustración 36). Viéndolo un poco más en profundidad este comando se encargará de comprobar mediante la opción *-d* que la ruta es un directorio válido y

mediante -*s* que su tamaño es mayor que 0. Seguidamente se esperará a que termine el proceso y se comprobará su valor de retorno, significando que

- 0 – El resultado de *test* es positivo, por tanto, la ruta existe y tiene contenido. La ruta introducida por el usuario no es válida.
- 1 – El resultado de *test* es negativo, por tanto, o bien la ruta no existe, o existe, pero está vacía. La ruta introducida por el usuario es válida.

Depende del valor devuelto se le solicitará al usuario que introduzca otra ruta mediante un mensaje de error.

```
public static int checkOracleBase (String path, String
hostname) {
  int returnedValue = -1;
  Process theProcess = null;

  try
  {
   theProcess = Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      String.format("/usr/bin/test -d %s -a -s %s", path,
path)});
   returnedValue = theProcess.waitFor();

  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
     return returnedValue;
 }
```

Ilustración 35: Código Java - checkOracleBase()

```
create or replace FUNCTION checkOracleBase (path VARCHAR2,
hostname VARCHAR2) RETURN NUMBER AS
LANGUAGE JAVA NAME
'avanttic.Oracle11Release2.checkOracleBase(java.lang.String,
java.lang.String) return java.lang.Integer';
```

Ilustración 36: Código PL/SQL - checkOracleBase()

Para la comprobación del *Oracle Home* hemos utilizado un método similar como el que podemos ver en la Ilustración 37 (código PL/SQL en Ilustración 38). Adicionalmente respecto al código del método anterior se ha comprobado que el *Oracle Home* este contenido dentro del *Oracle Base*. Para ello se ha hecho uso de expresiones regulares mediante la clase *Pattern* de Java y su método *matches()* que permite comprobar si una cadena dada corresponde con una expresión regular. La expresión regular es la siguiente

**^$ORACLE_BASE/?***

```
public static int checkOracleHome (String oracleBase, String
oracleHome, String hostname)
{
  int returnedValue = -1;
  Process theProcess = null;

  boolean b = Pattern.matches(String.format("^%s/?.*",
oracleBase),    oracleHome);

  if ( ! b ) { return 0; }

  try {
   theProcess = Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      String.format("test -d %s -a -s %s", oracleHome,
oracleHome)});
   returnedValue = theProcess.waitFor();

  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
     return returnedValue;
 }
```

Ilustración 37: Código Java - checkOracleHome()

```
create or replace FUNCTION checkOracleHome (oracleBase
VARCHAR2, oracleHome VARCHAR2, hostname VARCHAR2) RETURN
NUMBER AS
LANGUAGE JAVA NAME
'avanttic.Oracle11Release2.checkOracleHome(java.lang.String,
java.lang.String, java.lang.String) return java.lang.Integer';
```

Ilustración 38: Código PL/SQL - checkOracleHome()

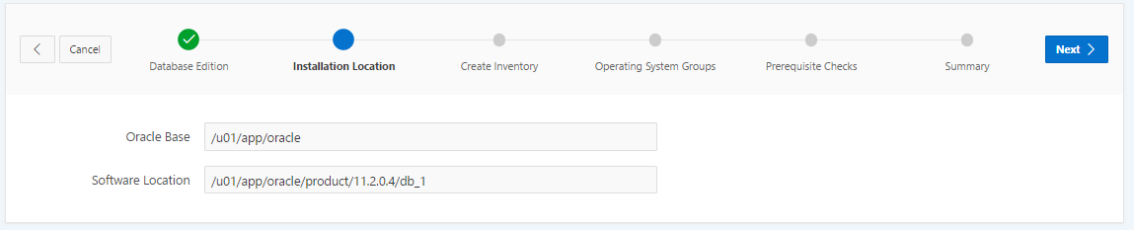


Ilustración 39: Asistente de Instalación - Installation Location

Los valores por defecto del *Installation Location* son:

- **Oracle Base** - /u01/app/oracle
- **Oracle Home** - /u01/app/oracle/product/11.2.0.4/db_1

El siguiente paso será introducir cuál será la ruta del *Inventory* y cuál será el grupo del sistema operativo que será dueño de dicho *Inventory*. Este paso podemos verlo en la Ilustración 40.

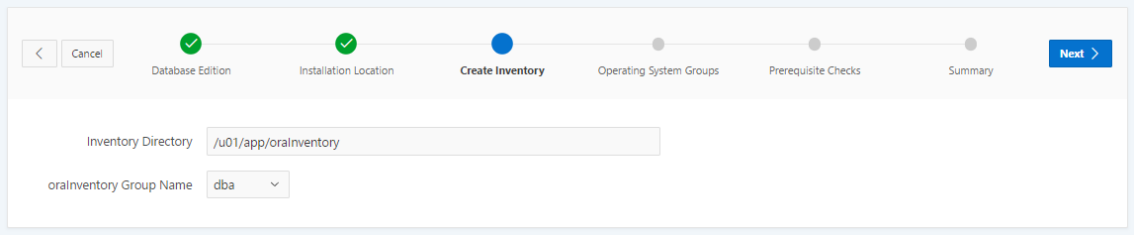


Ilustración 40: Asistente de Instalación - Create Inventory

El *OraInventory* debe cumplir una serie de requisitos, los cuales son:

- Debe ser una ruta unix válida.
- No puede estar contenido dentro del *Oracle Base.*
- La ruta especificada no debe existir o si existe debe estar vacía.

Por ello se ha desarrollado el siguiente código Java para cumplir estos requisitos, el código podemos verlo en la Ilustración 41 (código PL/SQL en Ilustración 42).

```
public static int checkOraInventory (String oracleBase, String
oraInventory, String hostname)
{
  int returnedValue = -1;

  boolean b = Pattern.matches(String.format("^%s/?.*",
oracleBase),    oraInventory);

  if ( b ) { return 0; }

  try {
   returnedValue = Runtime.getRuntime().exec(new String [] {
      "/usr/bin/ssh",
      "root@"+hostname,
      String.format("test -d %s -a -s %s",
oraInventory,oraInventory)})
   .waitFor();

  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
     return returnedValue;
 }
```

Ilustración 41: Código Java - checkOraInventory()

```
create or replace FUNCTION checkOraInventory (oracleBase
VARCHAR2, oraInventory VARCHAR2, hostname VARCHAR2) RETURN
NUMBER AS
LANGUAGE JAVA NAME
'avanttic.Oracle11Release2.checkOraInventory(java.lang.String,
java.lang.String, java.lang.String) return java.lang.Integer';
```

Ilustración 42: Código PL/SQL - checkOraInventory()

Una vez comprobado que el *Inventory* es válido pasaremos a asignar los grupos de administración con los grupos del sistema operativo. Este paso podemos verlo en la Ilustración 43.

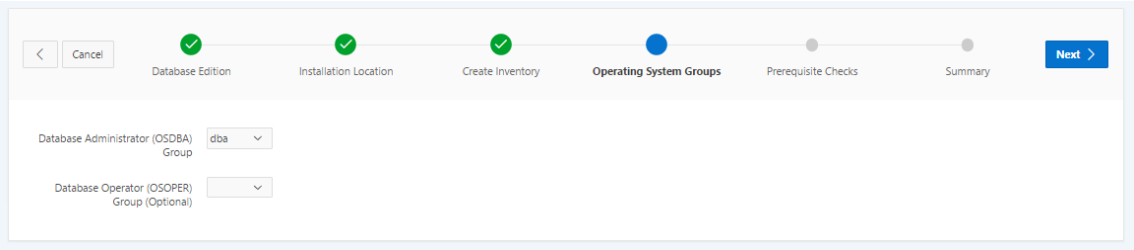


Ilustración 43: Asistente de Instalación - Operating System Groups

Este paso no cuenta con grandes restricciones ya que por defecto los grupos que se pueden elegir vienen delimitados, el *Database Administrator Group* vendrá dado por el grupo *dba* y el *Database Operator Group* vendrá sin valor por defecto ya que es opcional.

El último paso que tenemos es un sumario que contiene información sobre el sistema destino y sobre las opciones elegidas en el software que se instalará en él. Esta información la podemos ver en la Ilustración 44. La información es la siguiente:

- **Target** – IP o hostname del sistema destino.
- **Disk Space** – Espacio de disco requerido para la instalación y disponible en el sistema destino.
- **Source Location** – Define la localización del software de instalación.
- **Database Edition** – Define que versión de base de datos se ha elegido.
- **Oracle Base** – Define la localización del *Oracle Base*.
- **Software Location** – Define la localización del *Oracle Home*.
- **Priviliged Operating System Groups** – Define que grupos del sistema operativo tienen privilegios de administración en la base de datos.

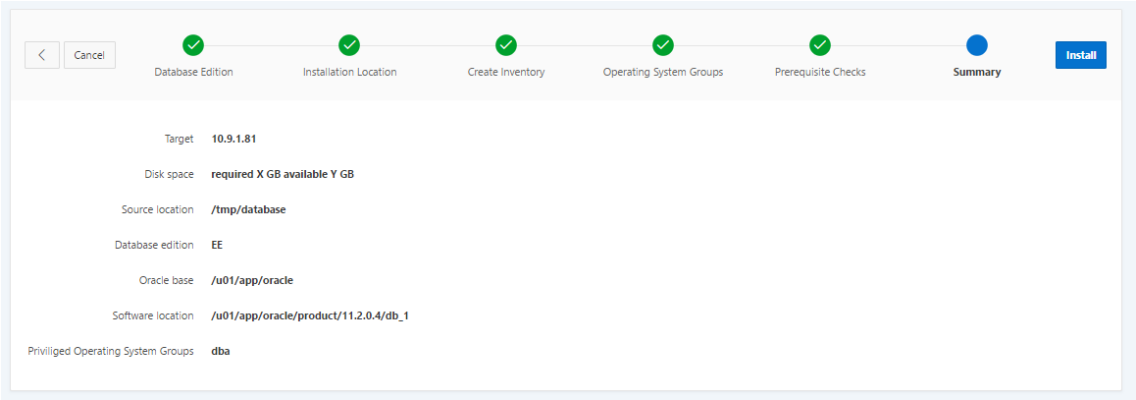


Ilustración 44: Asistente de Instalación – Summary

Una vez que el usuario ha introducido toda la información necesaria y que esta se ha verificado cuando pulse el botón *install* comenzará el proceso de instalación.

Como hemos visto previamente hemos usado lo que denominamos *Response File* para saber qué información necesita el instalador para ejecutarse de forma automática. Para evitar tener que trabajar con archivos y no tener que acceder al sistema donde se encuentra instalado APEX lo hemos hecho es pasar mediante argumentos los atributos clave-valor que contiene el *Response File* con la forma *atributo=valor* separados por espacios entre atributos, de esta forma junto con la opción *silent* evitaremos tener que manejar archivos y podremos ejecutar el instalador directamente desde el código Java.

El proceso se inicia con el código PL/SQL que vemos en la Ilustración 45. Analizando un poco más en detalle este código vemos que su función es la de recorrer la tabla *software_only* la cual contiene pares clave-valor que corresponden con los atributos del *Response File* a los cuales hemos ido asignando valores conforme se iba sucediendo el asistente de instalación. Conforme va recogiendo estos valores va creando cadenas del tipo *atributo=valor* y concatenándolas entre ellas. Cuando finalmente la cadena de atributos está finalizada se llamará al método *installSoftware()* que corresponde al código Java que vemos en la Ilustración 45.

```
DECLARE
   attrs varchar2(8000);
BEGIN
   FOR rec IN (SELECT key, value FROM software_only) LOOP
      attrs := attrs || '\"'||rec.key||'='||rec.value||'\" ';
   END LOOP;

   attrs := installsoftware(:ORACLE_BASE, :ORACLE_HOME, attrs,
:HOSTNAME);
END;
```

Ilustración 45: Código PL/SQL instalación

### 5.5.4. Scrum diario

Durante este *sprint* se ha realizado contacto diario entre miembros del equipo, en su mayoría de forma telemática para el planteamiento de dudas y su resolución para permitir un avance en el progreso del proyecto.

### 5.5.5. Revisión del sprint

A la finalización de este *sprint* se ha realizado una reunión entre los diferentes miembros del grupo y otros interesados para comprobar si la interfaz desarrollada es idónea para cumplir con el objetivo que se había propuesto y si es necesario añadir o eliminar elementos de la interfaz. No se han realizado modificaciones en el *Product Backlog.*

### 5.5.6. Retrospectiva del sprint

Tras la reunión de revisión se ha realizado una nueva reunión con el objetivo de evaluar y probar la interfaz desarrollada y el sistema finalizado de instalación de software de bases de datos. Se ha determinado que el sistema cumple con las expectativas acordadas y que su funcionamiento es el esperado.

## 5.6. Sprint 5: Lógica de creación de bases de datos

### 5.6.1. Refinamiento del Product Backlog

El *Product Backlog* no ha sufrido ninguna modificación.

### 5.6.2. Planificación del sprint

En este *sprint* han realizado las historia de usuario 7 y 8, que podemos ver en la Tabla 16 y la Tabla 17. La descomposición en tareas podemos apreciarlo en la Tabla 18.

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 7 | **Sprint:** 5 | **Esfuerzo:** 30 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Análisis para la creación de bases de datos | | |
| **Descripción:** Se debe estudiar los métodos para instalación de software de datos en especial por línea de comandos. | | |
| **Postcondición:** Haber adquirido los conocimientos necesarios para la creación de bases de datos por línea de comandos a través del *dbca* | | |

Tabla 16: Historia de usuario 7

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 8 | **Sprint:** 5 | **Esfuerzo:** 130 h |
| **Prioridad**: Alta | **Rol:** Equipo | |
| **Nombre:** Implementación de lógica para la creación de bases de datos | | |
| **Descripción:** Se desarrollar el código Java y Bash necesario para la creación automática de bases de datos. | | |
| **Postcondición:** Código ejecutable y sin errores que permita la creación automática de bases de datos a través del *dbca*. | | |

Tabla 17: Historia de usuario 8

| Nombre | Estimación | Historia de Usuario | Prioridad |
|---|---|---|---|
| Estudio y análisis del funcionamiento de bases de datos | 10 h | 7 | Alta |
| Estudio y análisis sobre la configuración de bases de datos Oracle | 5 h | 7 | Baja |
| Estudio y análisis del Database Configuration Assistant | 1 h | 7 | Baja |
| Obtención de requisitos para creación de bases de datos Oracle | | 7 | |
| Diseño estructural del código a desarrollar | 12 h | 8 | Alta |
| Implementación código Bash | 38 h | 8 | Alta |
| Implementación código Java | 30 h | 8 | Alta |
| Creación de procedimientos en base de datos Oracle | 10 h | 8 | Media |
| Pruebas y refinamiento | 40 h | 8 | Alta |
| **Total** | 160 h | | |

Tabla 18: Tareas asociadas al sprint 5

### 5.6.3. Desarrollo del sprint

*Database Configuration Assistant (DBCA)* es el medio preferido para crear una base de datos ya que es la aproximación más automatizada, tu base de datos estará completamente operativa cuando el asistente haya terminado. Esta herramienta está añadida en el software de bases de datos que hemos instalado previamente. Nosotros podemos ejecutar el *DBCA* tanto en modo interactivo como no interactivo, es decir, ejecutándose de forma automática, esto podemos realizarlo mediante el uso de argumentos en la línea de comandos o mediante un *Response File* como vimos en la iteración anterior.

En la Ilustración 46 podemos ver un ejemplo de cómo crear una base de datos a través de la línea de comandos de forma automática.

```
$ dbca -silent -createDatabase -templateName
General_Purpose.dbc
  -gdbname ora11g -sid ora11g -responseFile NO_VALUE -
characterSet AL32UTF8 -memoryPercentage 30 -
emConfiguration LOCAL
```

Ilustración 46: Ejemplo uso dbca por terminal

Ahora analizaremos en más en detalle cual es la información que necesitamos obtener a través del usuario para la creación de la base de datos. Tener en cuenta que para la creación de una base de datos se necesitan una gran cantidad de parámetros diferentes de diversa índole, el asistente por interfaz gráfica permite la introducción de algunos de ellos, sin embargo, por línea de terminal mediante argumentos no se permite introducir tantos como por interfaz gráfica. No debemos alejarnos de nuestro propósito de que el usuario final tenga que preocuparse lo menos posible para conseguir su base de datos funcional, por ello se ha elegido una serie de parámetros como esenciales y críticos, los cuales serán los que se solicitarán al usuario en el asistente de creación de la base de datos. Estos parámetros se han decidido a partir de la opinión de un *DBA (Database Administrator)* con experiencia perteneciente a la empresa avanttic, esta persona tiene experiencia de años en la administración de base de datos y conoce de primera mano cuales son los parámetros más críticos y esenciales en el proceso de creación de una base de datos. Por ello, en el resto de este punto se pasará a explicar cuáles son las opciones o argumentos que se permiten al ejecutar el *dbca* y a que parámetros de la base de datos hace referencia.

### *createDatabase*

Define la opción que permite la creación de bases de datos especificando una serie de parámetros.

### *templateName*

Define la ruta del *template*. Un *template* es como una especie de *Response File*, en nuestro caso se pondrá a *NO_VALUE*, ya que la información se añadirá mediante el uso de argumentos.

### *gdbName*

Define el nombre global para la base de datos.

### *sid*

Define el identificador de la base de datos en el sistema, éste debe ser único y no existir previamente.

***sysPassword***

Define la contraseña para el usuario *SYS*. Este usuario es el encargado de realizar todas las funciones administrativas mediante privilegios *SYSDBA*. Permitiendo realizar funciones como *backups* y *recoveries*.

***systemPassword***

Define la contraseña para el usuario *SYSTEM*. Este usuario puede realizar funciones administrativas salvo *upgrades, backups* y *recoveries*.

***emConfiguration***

Define la opción que permite la gestión del *Oracle Enterprise Manager*, como aquí no hará uso del *OEM* su valor por defecto será *NONE*.

***disableSecurityConfiguration***

Define la opción sobre la configuración de seguridad, su valor por defecto será *NONE*.

***datafileDestination***

Define cuál será la ruta donde se guardarán los archivos de la base de datos, el valor por defecto será *$ORACLE_BASE/oradata*, aunque el usuario podrá definir otro si lo desea.

***recoveryAreaDestination***

Define el área donde se encontrarán los archivos de recuperación. El valor por defecto será *$ORACLE_BASE/fast_recovery_area*, aunque el usuario podrá definir otro si lo desea.

***storageType***

Define la opción del tipo de almacenamiento que usará la base de datos, el valor por defecto será *FS (File System)*.

***characterSet***

Define el *character set* de la base de datos. El valor por defecto es *WE8MSWIN1252*, aunque el usuario podrá elegir entre otros.

***databaseType***

Define el tipo de base de datos que se creará, el valor por defecto será *MULTIPURPOSE*.

Ésta es toda la información que necesitamos aportar mediante línea de comandos para la creación de la base de datos de forma completamente automática y transparente. El código fuente que se encarga de llevar a cabo la ejecución del comando de instalación lo podemos ver en la Ilustración 47 (código PL/SQL en Ilustración 48). Analizando el código vemos que es muy simple, la única función que realiza es la de tomar un comando a través de argumentos y ejecutarlo mediante el usuario *oracle*, los diferentes argumentos y sus valores serán suministrados por la interfaz que verá el usuario.

```
public static void createDatabase (String hostname, String
command) {
  Process theProcess = null;
  BufferedReader inStream = null;

  try {
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "source /home/oracle/.bash_profile && su - oracle -c",
"\"", command,       "\""});
   theProcess.waitFor();

  } catch (Exception e) {
   // TODO: handle exception
   e.printStackTrace();
  }
 }
```

Ilustración 47: Código Java - createDatabase()

```
create or replace PROCEDURE createDatabase (hostname varchar2,
command varchar2)
as language java
name
'avanttic.Oracle11Release2.createDatabase(java.lang.String,
java.lang.String)';
```

Ilustración 48: Código PL/SQL - createDatabase()

### 5.6.4. Scrum diario

Durante este *sprint* se ha realizado contacto diario entre miembros del equipo, en su mayoría de forma remota para el planteamiento de dudas y su resolución para permitir el avance en el progreso del proyecto.

### 5.6.5. Revisión del sprint

A la finalización de este *sprint* se ha realizado una reunión entre los diferentes miembros del grupo y otros interesados para comprobar si el código implementado es idóneo para cumplir con el objetivo que se había propuesto y si es necesario añadir o eliminar algo. No se han realizado modificaciones en el *Product Backlog.*

### 5.6.6. Retrospectiva del sprint

Tras la reunión de revisión se ha realizado una nueva reunión con el objetivo de evaluar y probar el código desarrollado. Se ha determinado que el sistema cumple con las expectativas acordadas y que su funcionamiento es el esperado.

## 5.7. Sprint 6: Interfaz de creación de bases de datos

Según el plan previamente establecido en este *sprint* se han realizado las tareas asociadas a la historia de usuario 9. En esta historia de usuario se llevará a cabo el diseño e implementación de una interfaz que permita seguir al usuario en la creación de bases de datos de forma sencilla y transparente, solicitándole la menor información posible. Al igual que en el *sprint* 4 dicha interfaz se realizará a través de Oracle APEX junto con su lógica de control.

### 5.7.1. Refinamiento del Product Backlog

El *Product Backlog* no ha sufrido ninguna modificación.

### 5.7.2. Planificación del sprint

En este *sprint* se ha realizado la historia de usuario 9, que podemos ver en la Tabla 19. La descomposición en tareas podemos apreciarlo en la Tabla 20.

**Historia de usuario**

| | | |
|---|---|---|
| **Número:** 9 | **Sprint:** 6 | **Esfuerzo:** 40 h |
| **Prioridad**: Media | **Rol:** Equipo | |
| **Nombre:** Diseño e implementación de interfaz para creación de bases de datos | | |
| **Descripción:** Se debe desarrollar una interfaz a través de Oracle APEX que permita la creación de bases de datos de forma remota. | | |
| **Postcondición:** La existencia de un asistente de instalación funcional que permita la creación de bases de datos de forma remota. | | |

Tabla 19: Historia de usuario 9

| Nombre | Estimación | Historia de Usuario | Prioridad |
|---|---|---|---|
| Diseño de los pasos que seguirá el asistente a desarrollar | 3 h | 9 | Alta |
| Reunión con *DBA* para análisis de información más relevante | 3h | 9 | Media |
| Implementación de los pasos del asistente de creación de bases de datos | 10 h | 9 | Alta |
| Dotación de lógica de control al asistente de creación de bases de datos | 14 h | 9 | Alta |
| Pruebas y refinamiento | 17 h | 9 | Alta |
| **Total** | 40 h | | |

Tabla 20: Tareas asociadas al sprint 6

### 5.7.3. Desarrollo del sprint

Para la obtención de la información para la creación de bases de datos necesitaremos que el usuario introduzca de forma manual cierta información crítica pero trivial referente a parámetros que conciernen a las bases de datos. Para la comodidad del usuario se ha creado un asistente de instalación que lo guiará a lo largo del proceso de creación mediante una interfaz sencilla, simple y entendible. Esta interfaz se ha realizado a través de Oracle APEX. Se ha intentado simplificar lo máximo posible el proceso de creación para el usuario, que éste tenga que introducir únicamente la información más sencilla y que el resto venga por defecto para permitir un despliegue rápido, simple y transparente. A su vez se ha realizado una lógica de control sobre lo que introduce el usuario para evitar errores posteriores en la instalación.

Para tener persistencia de los datos desde el inicio del asistente hasta el final se ha creado una tabla *database_deployment* que está compuesta de una tupla clave-valor, donde la clave es cada uno de los atributos que puede tomar el *dbca* por línea de comandos como atributo y el valor como su nombre indica el valor de dicho atributo. Esta tabla se inicializará al principio del asistente con valores por defecto que se irán modificando a lo largo del asistente. Esto nos permitirá volver atrás y adelante a lo largo del asistente sin tener que volver a rellenar cada uno de los campos.

A continuación, se detallan y se muestran cada uno de los pasos del asistente de creación de la base de datos y las distintas funciones que se han desarrollado para corroborar el correcto funcionamiento del programa dotándolo de lógica de control.

Inicialmente se solicitará al usuario que introduzca el nombre de la base de datos y el *SID* que se rellenará automáticamente con el valor del nombre de la base de datos, aunque puede ser modificado si el usuario lo desea. En la Ilustración 49 podemos ver este paso del asistente.

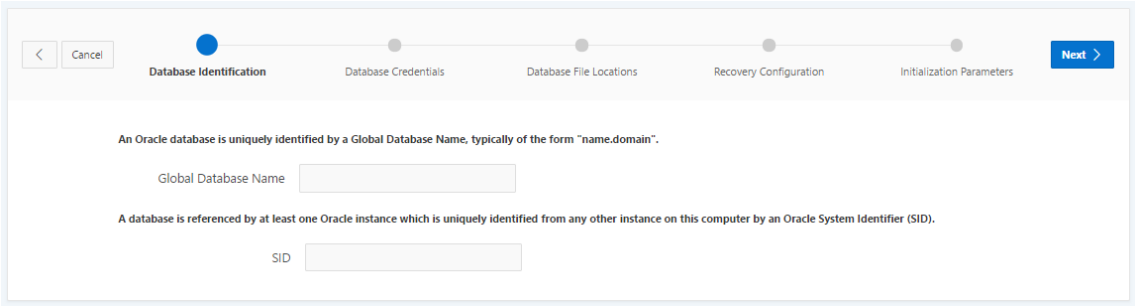


Ilustración 49: Asistente de creación de bases de datos - Database Identification

Este paso del asistente debe cumplir la restricción de que el nombre de la base de datos no exista en el sistema ya que estos identificadores deben ser únicos. Para ello se ha desarrollado el código que podemos ver en la Ilustración 50, este se encarga de buscar en el archivo */etc/oratab*, en el cuál se almacenan los nombres de las bases de datos del sistema, si existe previamente el nombre elegido, en caso de no existir se permite continuar con el asistente, de no ser así se mostrará un mensaje de error.

```
public static int checkIfDatabaseExist (String dbName, String
hostname)
{
  Process theProcess;
  int returnedValue;

  try {
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     String.format("/usr/bin/cat  /etc/oratab  |  /usr/bin/grep
^%s:", dbName)});
   returnedValue = theProcess.waitFor();

   // Existe la base de datos
   if ( returnedValue == 0 ) {
    return 0;
   }
```

```
  // No existe la base de datos
  else {
   return 1;
  }

 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }

 return 0;
}
```

Ilustración 50: Código Java - checkIfDatabaseExist()

El siguiente paso en el asistente es la introducción de las credenciales para los usuarios *SYS* y *SYSTEM* que como vimos en el *sprint* anterior son los encargados de la administración de la base de datos. En este paso se comprueba mediante validaciones de Oracle APEX que las contraseñas concuerden entre ellas. Podemos ver el paso en la Ilustración 51.

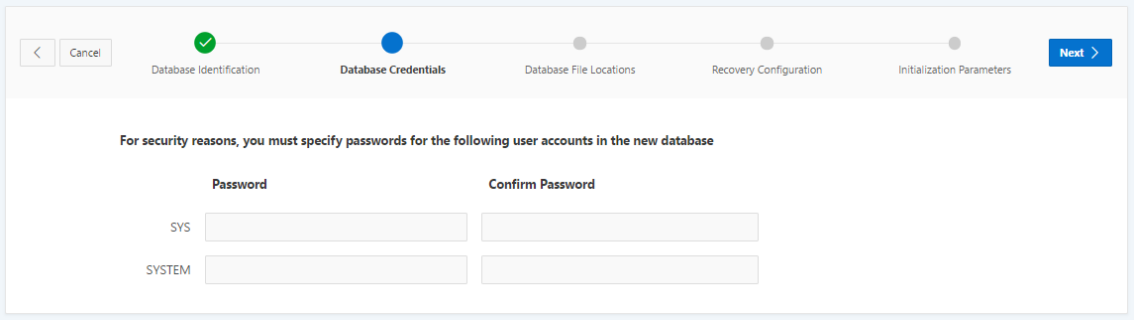


Ilustración 51: Asistente de creación de bases de datos - Database Credentials

El siguiente paso con el que nos encontramos es donde se alojarán los archivos de la base de datos, este traerá como valor por defecto *$ORACLE_BASE/oradata*, aunque el usuario puede cambiarlo si lo desea. Podemos ver este paso en la Ilustración 52.

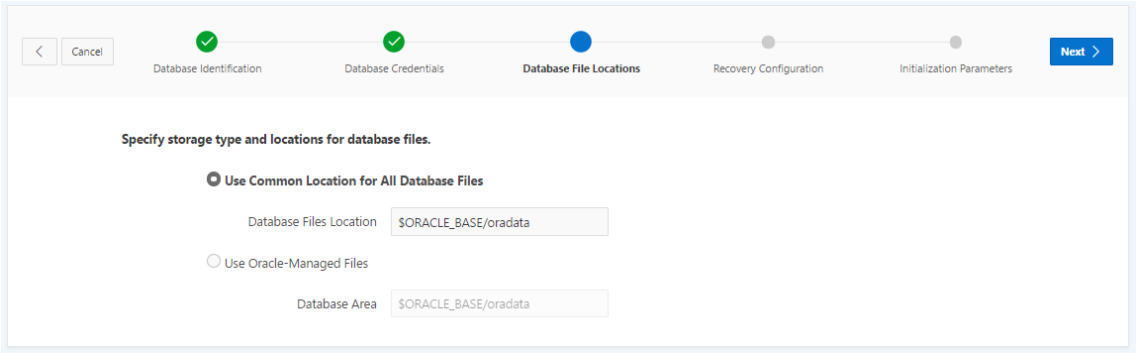


Ilustración 52: Asistente de creación de bases de datos - Database File Locations

Seguidamente se pasa a definir cuál será la localización del *fast recovery area* y cuál será el tamaño de ésta. Podemos ver este paso en la Ilustración 53.

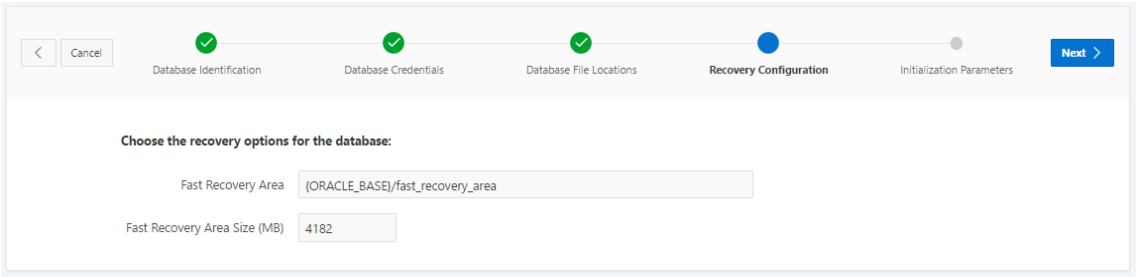


Ilustración 53: Asistente de creación de bases de datos - Recovery Configuration

Finalmente, como último paso se solicitará al usuario que introduzca cuál es la memoria que desea asignar al sistema, definiendo también entre las opciones posibles el *database character set* y el *national character set*. Se informa a su vez al usuario cual es la memoria disponible en el sistema en el momento de la creación de la base de datos para que pueda tomar dicho valor como referencia, para ello se ha desarrollad el código que podemos ver en la Ilustración 55 (código PL/SQL en Ilustración 56). Podemos ver también dicho paso del asistente en la Ilustración 54.

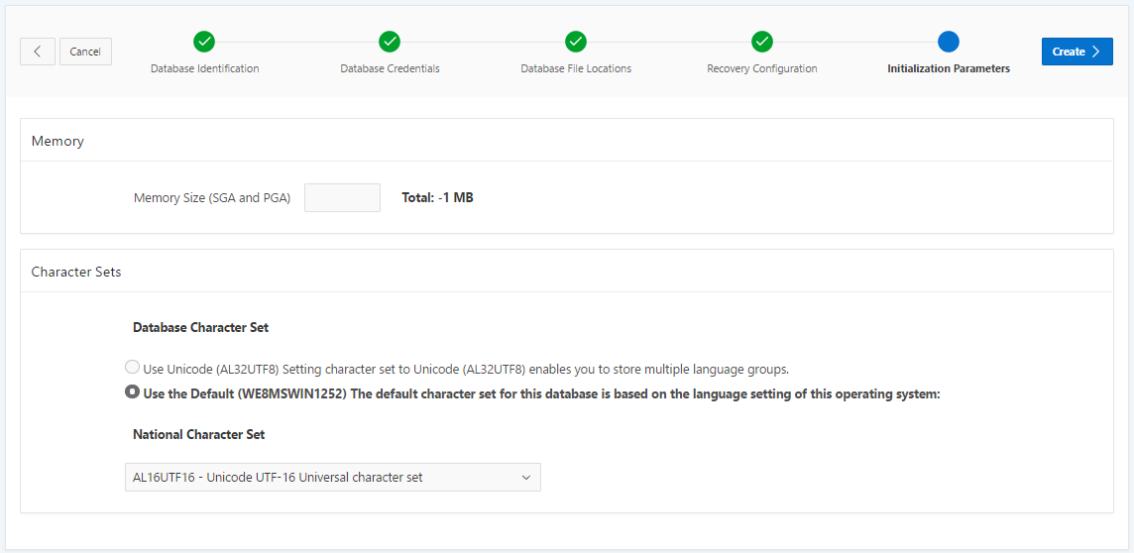


Ilustración 54: Asistente de creación de bases de datos - Initialization Parameters

Una vez que el usuario pulse el botón create se desencadenará una serie de acciones que se encargarán de crear la base de datos en el sistema final.

```
public static int getTotalMemory (String hostname)
{
  Process theProcess;
  BufferedReader inStream = null;

  try {
   theProcess = Runtime.getRuntime().exec(new String [] {
     "/usr/bin/ssh",
     "root@"+hostname,
     "/usr/bin/free  -m  |  /usr/bin/grep  Mem  |  /usr/bin/awk
'{print $2}'"});
   theProcess.waitFor();
   return Integer.parseInt(
     new BufferedReader(
          new InputStreamReader(
               theProcess.getInputStream())).readLine());


  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }

  return -1;
 }
```

Ilustración 55: Código Java - getTotalMemory()

```
create or replace FUNCTION getTotalMemory (hostname VARCHAR2)
RETURN NUMBER AS
LANGUAGE                        JAVA                            NAME
'avanttic.Oracle11Release2.getTotalMemory(java.lang.String)
return java.lang.Integer';
```

Ilustración 56: Código PL/SQL - getTotalMemory()

### 5.7.4. Scrum diario

Durante este *sprint* se ha realizado contacto diario entre miembros del equipo, en su mayoría de forma telemática para el planteamiento de dudas y su resolución para permitir un avance en el progreso del proyecto.

### 5.7.5. Revisión del sprint

A la finalización de este *sprint* se ha realizado una reunión entre los diferentes miembros del grupo y otros interesados para comprobar si la interfaz desarrollada es idónea para cumplir con el objetivo que se había propuesto y si es necesario añadir o eliminar elementos de la interfaz. No se han realizado modificaciones en el *Product Backlog*

### 5.7.6. Retrospectiva del sprint

Tras la reunión de revisión se ha realizado una nueva reunión con el objetivo de evaluar y probar la interfaz desarrollada y el sistema finalizado de instalación de software de bases de datos. Se ha determinado que el sistema cumple con las expectativas acordadas y que su funcionamiento es el esperado.

# 6. Conclusiones y propuestas

A continuación, se expone el análisis personal sobre el desarrollo de este proyecto, cuáles han sido los puntos positivos, que se podría mejorar para aportar más valor al proyecto y cuales han sido los problemas principales a los que nos hemos enfrentado a lo largo del desarrollo y finalmente una opinión personal.

## 6.5. Consecución de los objetivos parciales

Durante el desarrollo de este proyecto nos hemos enfrentando a una serie de dificultades que se han ido mostrando no de forma temprana si no a lo largo del proyecto y que han supuesto contratiempos en el desarrollo de éste. El mayor impedimento de este proyecto ha sido la dificultad para comprobar la compatibilidad entre los diferentes componentes de la arquitectura de la cual está formada el sistema. El alto acoplamiento entre los componentes no permite una depuración sencilla y esto ha lastrado enormemente el desarrollo del proyecto. No obstante, pese a las adversidades el proyecto ha salido adelante y ha permitido conseguir completar tanto los objetivos parciales como el objetivo principal en el tiempo esperado.

## 6.6. Posibles ampliaciones del proyecto

Inicialmente este proyecto se componía de 3 iteraciones, instalación del software de bases de datos, creación de bases de datos e implantación de un *Oracle Data Guard* [25]. Por contratiempos y motivos temporales, la realización de la tercera iteración ha sido posible, sumado al factor temporal se une que para la realización de esta iteración son necesarios conocimientos específicos sobre bases de datos los cuales no ha sido posible adquirir. Creo que una buena ampliación del proyecto sería terminar de desarrollar esta iteración y añadirla al sistema final que ya tenemos elaborado, junto con esto podríamos añadir características más avanzadas de bases de datos, como la instalación en RAC[37] o en *cluster*.

[37] *Real Application Cluster*

Por otro lado, también sería viable el plantear y analizar la forma de desarrollar la idea previa a este proyecto, es decir el desarrollo del *plugin* para el *Oracle Enterprise Manager*, con más tiempo para investigar cuales son las herramientas que pueden permitir esto sería una buena opción para el desarrollo de un nuevo proyecto, teniendo éste como referencia y viéndolo como una ampliación que como si empezáramos de 0.

Sería también interesante ver de qué otras formas se podría realizar el mismo trabajo que realiza este mismo sistema, que otras herramientas se pueden emplear en vez de las usadas aquí, que otros lenguajes, que otras tecnologías… Podría ampliarse también a otras bases de datos como MySQL, PostgreSQL, … Creo que este proyecto tiene muchas salidas posibles y ampliaciones tanto a lo ancho como a lo largo como añadiendo mayores características a las opciones actuales o directamente añadiendo más opciones, permite un crecimiento con tecnologías más abiertas o continuar con una visión más privativa, ambas son completamente viables. Siempre y cuando vayan encaminadas con la visión de la empresa.

## 6.7. Opinión personal

Este proyecto ha supuesto un gran reto para mí por diversas razones. Debido a mi tardía entrada en Avanttic S.L. el proyecto no ha alcanzado las dimensiones que esperaba ni ha tomado la forma que desearía. Es la primera vez que tengo que adaptarme completamente a la filosofía de trabajo de una empresa con unas herramientas determinadas y esto ha supuesto un gran reto para mí, acostumbrado al uso de herramientas de software libre y de solventar los problemas de diversas formas me he encontrado con la gran dificultad que supone el trabajar herramientas privativas o corporativas donde la documentación es escasa y en muchos casos insuficiente llevando a grandes frustraciones y bloqueos. Me ha costado mucho encajar en una forma de trabajar tan acotada, no así con la gente que trabaja de esta forma. Yo personalmente hubiera realizado este proyecto desde otros puntos de vista más "abiertos", pero es la experiencia que me llevo lo que más vale. Darle las gracias a Avanttic S.L. por la oportunidad que me ha brindado con este proyecto y por confiar en mí. Gracias a mis compañeros de trabajo, les deseo lo mejor.

# 7. Bibliografía y referencias


[1] «Oracle | Integrated Cloud Applications and Platform Services,» [En línea]. Available: https://www.oracle.com/index.html. [Último acceso: 15 Marzo 2017].

[2] «Escuela Superior de Informática (UCLM) » Empresas » Programa profESIonalízate,» [En línea]. Available: http://webpub.esi.uclm.es/spa/paginas/empresas-profesionalizate. [Último acceso: 29 Agosto 2017].

[3] «Platinum Partner - Oracle,» [En línea]. Available: http://www.oracle.com/partners/en/partner-with-oracle/get-started/levels-benefits/platinum/index.html. [Último acceso: 27 Marzo 2017].

[4] «Oracle Database Online Documentation 11g Release 2 (11.2),» [En línea]. Available: https://docs.oracle.com/cd/E11882_01/nav/portal_11.htm. [Último acceso: 12 Abril 2017].

[5] «Oracle APEX,» [En línea]. Available: https://apex.oracle.com/es/. [Último acceso: 12 Mayo 2017].

[6] S. J. C. E. F. Codd IBM Research Lab, «A relational model of data for large shared data banks,» June 1970. [En línea]. Available: http://dl.acm.org/citation.cfm?id=362685.

[7] «Oracle SQL Language,» [En línea]. Available: http://www.oracle.com/technetwork/database/database-technologies/sql/overview/index.html. [Último acceso: 27 Abril 2017].

[8] «Oracle VM VirtualBox,» [En línea]. Available: https://www.virtualbox.org/. [Último acceso: 13 Abril 2017].

[9] «XShell 5,» [En línea]. Available: https://www.netsarang.com/products/xsh_overview.html. [Último acceso: 12 Abril 2017].

[10] «Eclipse Neon,» [En línea]. Available: https://eclipse.org/. [Último acceso: 28 Marzo 2017].

[11] «Java Software | Oracle,» [En línea]. Available: https://www.oracle.com/java/index.html. [Último acceso: 28 Marzo 2017].

[12] «Vagrant by Hashicorp,» [En línea]. Available: https://www.vagrantup.com/. [Último acceso: 27 Abril 2017].

[13] «Oracle Linux OS and Support | Operating Systems | Oracle,» [En línea]. Available: https://www.oracle.com/linux/index.html. [Último acceso: 25 Agosto 2017].

[14] «Creating a Database with DBCA,» [En línea]. Available: https://docs.oracle.com/cd/B28359_01/server.111/b28310/create002.htm#ADMIN12479. [Último acceso: 20 Agosto 2017].

[15] «Introduction to Oracle Universall Installer,» [En línea]. Available: https://docs.oracle.com/cd/E11857_01/em.111/e12255/oui1_introduction.htm. [Último acceso: 15 Agosto 2017].

[16] «OpenSSH,» [En línea]. Available: https://www.openssh.com/. [Último acceso: 10 Agosto 2017].

[17] «SQL*Plus User's Guide and Reference,» [En línea]. Available: https://docs.oracle.com/cd/B19306_01/server.102/b14357/toc.htm. [Último acceso: 15 Junio 2017].

[18] «Oracle SQL Developer,» [En línea]. Available: http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html. [Último acceso: 25 Agosto 2017].

[19] «Bash - GNU Project - Free Software Foundation - GNU.org,» [En línea]. Available: https://www.gnu.org/software/bash/. [Último acceso: 20 Agosto 2017].

[20] «Oracle PL/SQL,» [En línea]. Available: http://www.oracle.com/technetwork/database/features/plsql/index.html. [Último acceso: 27 Mayo 2017].

[21] «Windows,» [En línea]. Available: https://www.microsoft.com/es-es/windows/. [Último acceso: 25 Marzo 2017].

[22] «draw.io,» [En línea]. Available: https://www.draw.io/. [Último acceso: 27 Agosto 2017].

[23] «Microsoft Word 2016, Software de procesamiento de texto y documentos,» [En línea]. Available: https://products.office.com/es-es/word. [Último acceso: 27 Marzo 2017].

[24] «Atom,» 25 Agosto 2017. [En línea]. Available: https://atom.io/.

[25] «Oracle Data Guard with Oracle Database 11g Release 2,» [En línea]. Available: http://www.oracle.com/technetwork/es/database/enterprise-edition/documentation/tutorial-oracle-data-guard-11gr2-1707492-esa.pdf. [Último acceso: 27 Mayo 2017].

[26] www.ieee.org, «IEEE Citation Reference,» [En línea]. Available: https://www.ieee.org/documents/ieeecitationref.pdf. [Último acceso: 29 Agosto 2017].

[27] «Installing Oracle Database and creating a Databas,» [En línea]. Available: https://docs.oracle.com/cd/E11882_01/server.112/e10897/install.htm#ADMQS002. [Último acceso: 22 Agosto 2017].

[28] «Oracle Enterprise Manager 11g,» [En línea]. Available: http://www.oracle.com/technetwork/es/oem/grid-control/overview/index.html. [Último acceso: 20 Mayo 2017].