UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Prototipo de CAD 3D en la nube

Memoria

Alumna: Shamsa Kanwal

Especialidad: Ingeniería de Software

Título: Grado en Ingeniería de informática

Director: Óscar Llorens Maldonado

Ponente: Antoni Urpí Tubella

Fecha Entrega: 21/10/2021

Índice general

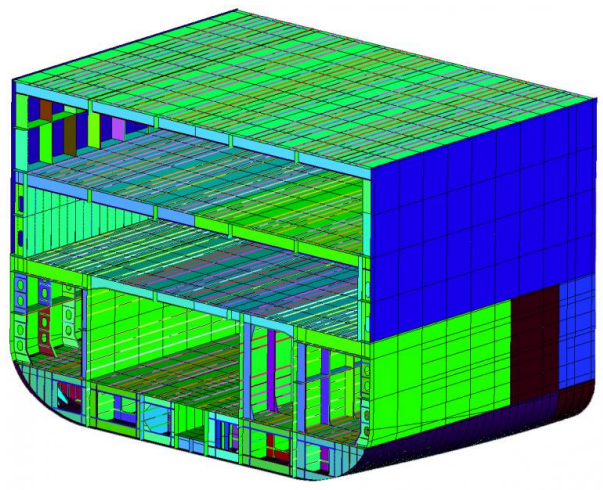
1. Contextualización y Alcance

1.1. Introducción y contextualización

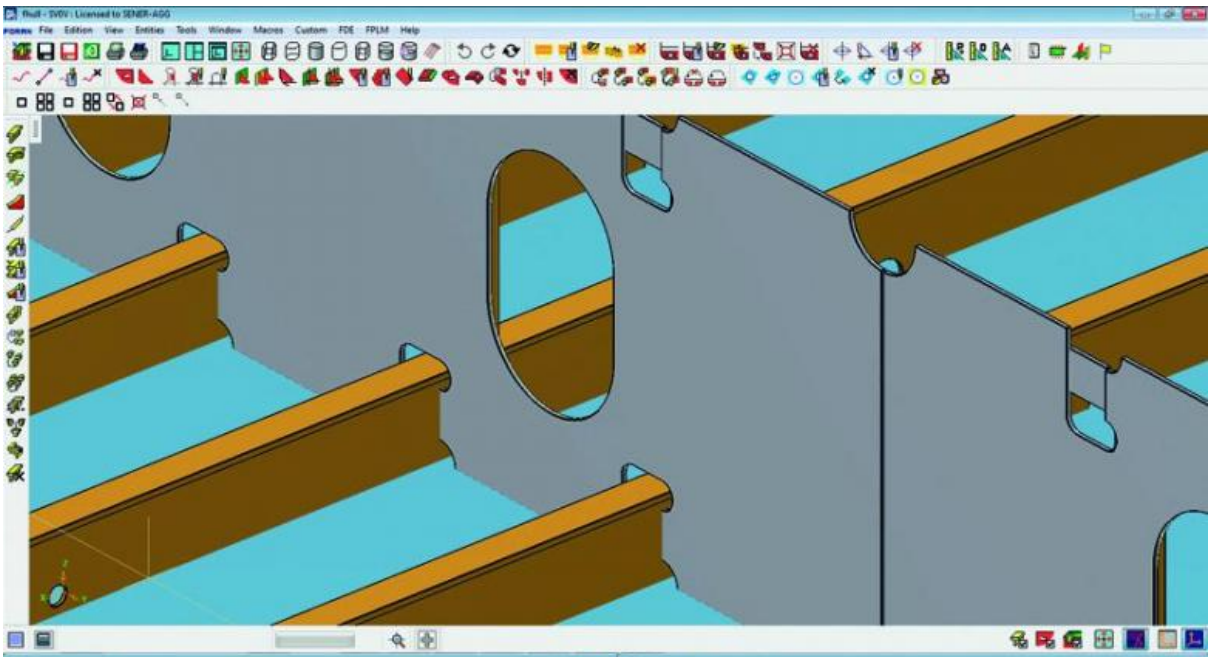
El proyecto “Prototipo de CAD 3D en la nube” se trataba de un trabajo fin de grado (TFG) para la Facultad de Informática de Barcelona (FIB), concretamente para el grado de ingeniería informática y la especialidad de ingeniería del software. Este trabajo se ha hecho dentro de la modalidad B, es decir, es un proyecto realizado en una empresa. El ponente por parte del profesorado del FIB es Antoni Urpí Tubella y el director del proyecto es Óscar Llorens Maldonado por parte de la empresa.

La empresa SENER, Ingeniería y Sistemas, S. A. es líder mundial en el desarrollo de proyectos con una elevada carga tecnológica, muchos de ellos entregados como llave en mano, en cuatro grandes sectores: Aeroespacial, Infraestructuras, Energía y Naval. [1]

FORAN es un sistema de diseño y construcción de buques y artefactos marinos desarrollado por SENER desde hace 50 años y actualmente licenciado en más de 150 astilleros y oficinas técnicas de 40 países. [2]. A continuación, podemos observar unas capturas de Foran.



Captura 1.1. Estructura interna. Fuente: https://www.marine.sener/subsystems



Captura 1.2. Estructura interna. Fuente: https://www.marine.sener/subsystems

1.1.1. Definiciones

Cloud Computing

De una forma sencilla podemos decir que la computación en la nube o Cloud Computing es una tecnología que deja acceso a software, almacenaje de ficheros y procesamiento de datos a través de Internet, siendo una opción alternativa a la ejecución en la computadora personal o bien servidor local.

CAD

El término CAD (Computer Aided Design o Diseño Asistido por Ordenador) hace referencia a una herramienta software que, mediante el uso del ordenador, permite crear, modificar, analizar y optimizar planos y modelos en dos y tres dimensiones, y manipular de una manera fácil elementos geométricos sencillos. Se trata de herramientas que van más allá del concepto de “dibujo” o representación gráfica. [3]

1.1.2. Problema a resolver

En pocas palabras, queríamos ser capaces de ofrecer al usuario otras formas de trabajar con FORAN.

Usar un equipo de menores prestaciones, y acceder a la información sin necesidad de grandes transferencias de datos y procesamiento local, por lo tanto, en un principio una conexión más rápida.

1.1.3. Stakeholders

A continuación, vemos los diferentes actores implicados (stakeholders), ya sean persones interesadas, que se beneficiarán, persones afectadas o participantes en el desarrollo.

- Desarrollador: El proyecto consta de una única desarrolladora, la Shamsa Kanwal, como parte del Trabajo del Fin de Grado de Ingeniería Informática. Ella se encargó de desarrollar el programa, hacer las pruebas como usuario, y de documentar el proyecto.
- Director del Trabajo del Fin de Grado: El director, Óscar Llorens Maldonado, es un miembro del programa Foran. Estuvo involucrado en la supervisión del proyecto y ayudó en que vaya en dirección correcta.
- Equipo Foran: Al final el proyecto es un prototipo que podríamos llegar a implementar en el programa FORAN.

1.2. Justificación

El sistema FORAN es rígido en cuanto a la configuración: se debe disponer de un equipo servidor con una base de datos, un servidor de licencias y un servidor de control de usuarios/permisos; además, cada usuario debe disponer de un equipo de alta prestación y preferiblemente con tarjeta gráfica dedicada. Esto conlleva además que todos los datos deben ser transferidos entre el servidor y el cliente, siendo muy lenta una conexión si se trata de VPN. Querríamos ser capaces de ofrecer al usuario otras formas de trabajar con FORAN, cómo sería trabajar en la nube con un

equipo de menores prestaciones, y acceder a la información sin necesidad de grandes transferencias de datos y procesamiento local.

1.3. Alcance

1.3.1. Objetivo

El principal objetivo de este proyecto fue poder demostrar si somos capaces de crear funcionalidades de FORAN usando Cloud Computing y cómo sería posible.
Sin embargo, nuestro objetivo estaba a alto nivel, si nos adentramos un poco más algunos de los objetivos principales de este proyecto eran:

- Comparar grandes proveedores de Cloud Computing y decidir por uno que nos convenía más.
- Crear un prototipo de CAD que contenía las mínimas funcionalidades del FORAN usando el proveedor seleccionado.

1.3.2. Requisitos

1.3.2.1. Requisitos funcionales

Crear un servicio en la nube que use Visor 3d (OpenCascade) que sea capaz de:

- Leer piezas de base de datos.
- Leer formas de casco.
- Definir alguna entidad y guardarla en base de datos.
- Sincronización de tal manera que podíamos leer en un FORAN algo definido en el visor web.

1.3.2.2. Requisitos no-funcionales

Para la primera parte del proyecto, comparativa de los proveedores del Cloud, eran:

- Seguridad
- Base de datos Oracle en la nube
- Computación

- Confidencialidad de datos en uso
- Nube híbrida: Posibilidad de tener base de datos en local
- Desarrollar usando lenguaje C++, a ser posible.

En cambio, para la segunda parte, la implementación, no tuvimos en cuenta ni eficiencia, ni apariencia, entre otras cosas. Ya que solo nos interesaba saber que éramos capaces de hacer usando esa tecnología.

1.3.3. Riesgos

A la hora de desarrollar un proyecto de software, aunque, aparentemente, parezca que no hay ningún riesgo visible, siempre nos podemos encontrar con unos obstáculos durante la ejecución del proyecto, como los que se mencionan a continuación.

- **Bugs**: Cuando estamos desarrollando algún software, uno de los problemas más comunes que nos hacen perder tiempo son los bugs, es decir, errores en el código. Podemos encontrarnos bugs de varios tipos, algunos son más fáciles de detectar que otros. Incluso podemos llegar a encontrar casos en que la librería que usamos contenga errores.
- **Desviaciones en el planning**: Podemos encontrarnos estas desviaciones por varios motivos, como pueden ser: nuevas funcionalidades que tienen más prioridad, cambios en el equipo, entre otras.
- **Plazo de entrega del proyecto**: Existe un plazo límite para la entrega del proyecto que hay que realizar. Esto obliga a tomar decisiones drásticas durante el desarrollo. Por lo tanto, tendrá que haber un buen plan que puede cambiar y cumplir con los plazos especificados para poder terminar el proyecto a tiempo.
- **Cosas imprevistas** como la actual pandemia, Coronavirus.
- **Visor 3D en navegador**: Podrá un navegador mover la escena.
- No ser capaz de **sincronizar base de datos**, o siquiera de importarlas.

1.4. Metodología

1.4.1. Metodología Scrum

Actualmente, existen diversas metodologías de trabajo, donde cada una está pensada para diferentes problemas y entornos de trabajo, por lo tanto, no hay una mejor que la otra. Se tienen que tener en cuenta varios factores a la hora de decidir qué metodología se adapta mejor a las necesidades. Se puede observar el *Framework* en la Figura 1.

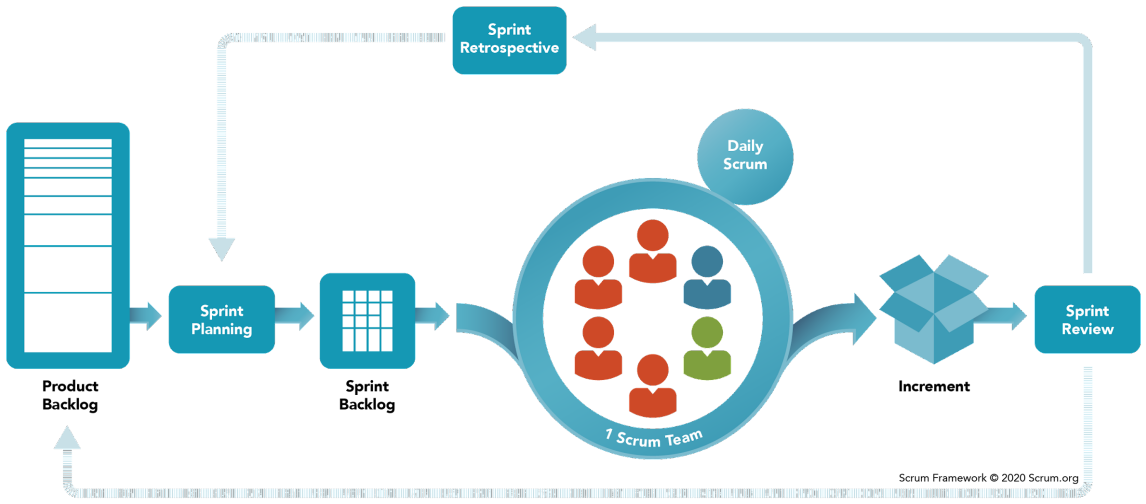


Figura 1. Framework de Scrum. Fuente: [4]

El Scrum tiene 4 aspectos a destacar. La primera es el *backlog*, que no es más que una lista de tareas pendientes. Esas tareas pendientes pueden ser de todo el proyecto o de las que se piensan hacer durante un *sprint.* El *sprint* es un periodo de tiempo determinado en el que el equipo Scrum trabaja para completar las tareas definidas para ese periodo de tiempo. Las tareas son *User stories* (historias de usuario) que no son más que una unidad pequeña con la que podemos trabajar, suelen tener un formato como en la Figura 2.

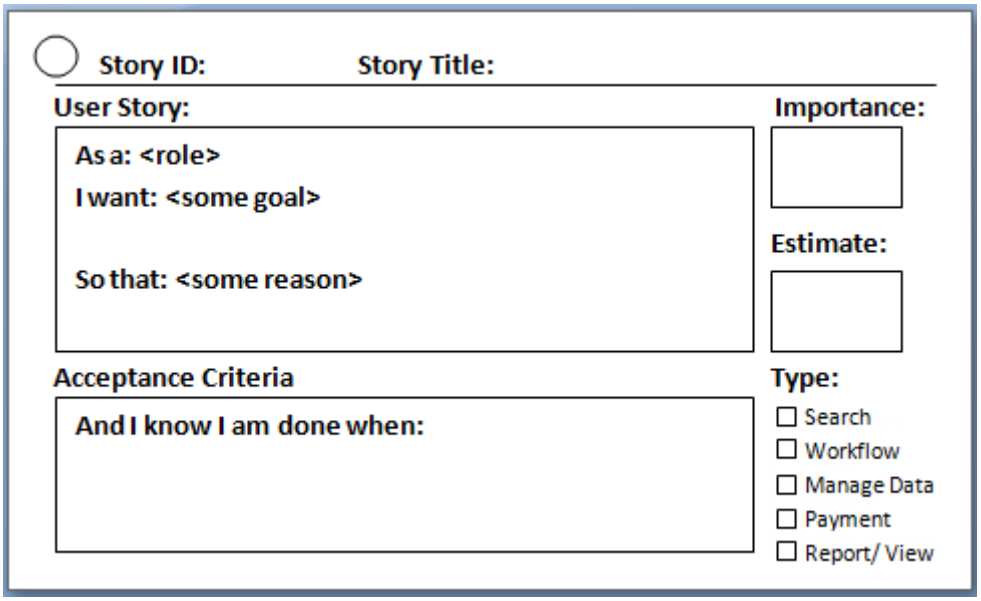


Figura 2. Formato de User Story. Fuente: [5]

El segundo aspecto a destacar de Scrum es que define tres responsabilidades, Product Owner, Scrum Master y Developer.

- Product Owner: Es la persona encargada de optimizar y maximizar el valor del producto, así como de priorizar este producto a través del Product Backlog (lista de tareas pendientes del proyecto).
- Scrum Master: Es la persona encargada de gestionar el proceso Scrum y de eliminar impedimentos para agilizar el desarrollo del producto.
- Equipo de Desarrollo: Equipo que suele estar formado entre 3 y 9 personas, en nuestro caso somos 2 personas (si contamos al director del proyecto) dedicadas al proyecto. Es el encargado de desarrollar el producto.

El tercer aspecto son los eventos que sirven para crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos están encuadrados en el tiempo. Los eventos son:

- Sprint Planning: Es una reunión que se hace al inicio de cada sprint y tiene como finalidad planear lo que se va hacer durante ese *sprint*. Suele durar cerca de una hora. Nosotros vamos a tener esta reunión cada dos semanas para iniciar un nuevo *sprint*.

- Daily Scrum: Es una reunión diaria que dura pocos minutos en la que cada miembro del equipo informa del estado de sus tareas al resto del equipo intentando resolver estas tres preguntas:
    - ¿Que complete ayer?
    - ¿En que trabajare yo?
    - ¿Estoy bloqueado con alguna cosa?
- Sprint Review: Es una reunión que se realiza al finalizar un *sprint*. El objetivo es revisar las iteraciones realizadas y exponer el trabajo realizado por el equipo.
- Sprint Retrospective: Es una reunión que se realiza al finalizar un *sprint.* El objetivo de esta reunión es analizar que ha ido bien y que ha ido mal para mejorar en el futuro.

Por último y no menos importante, está el aspecto de un incremento, que es un trampolín concreto hacia el objetivo del producto. Cada Incremento se suma a todos los Incrementos anteriores y se verifica minuciosamente, lo que garantiza que todos los Incrementos funcionen juntos. Para proporcionar valor, el incremento debe ser utilizable.

1.4.2. Metodología adaptada

El equipo estaba formado por Shamsa Kanwal y el director. Pero la única persona que desarrolló fue Shamsa Kanwal. Por lo tanto, tuvimos una metodología adaptada.

El proyecto, más bien era un estudio. Por lo tanto, lo más probable era que las tareas se vayan cambiando en cada avance. Por esta razón, en el backlog teníamos tareas de tamaño grande que incluso podrían llevar hasta 50 o más horas.

Para nosotros, consideraríamos un sprint acabado cuando habíamos acabado las tareas asignadas para ese Sprint. Por lo tanto, la duración del Sprint se fue cambiando, entre lo anteriormente comentado y también por no tener experiencia en

la tecnologías, uno de los motivos que nos puede llevar más tiempo de lo previsto. Una excepción para acabar el Sprint sin haber hecho las tareas correspondientes era, no saber continuar la tarea o quedarse atascado durante bastante tiempo sin ningún tipo de avance.

Otro cambio respecto Scrum es que no tendríamos User Stories para todas las tareas. Por ejemplo, tener user story para la comparativa de proveedores de la nube no tiene sentido, porque al usuario no le interesa que proveedor estamos usando.

Responsabilidades por este proyecto:

- Product Owner: Oscar Llorens Maldonado, el director.
- Scrum Master: Shamsa Kanwal.
- Equipo de Desarrollo: Shamsa Kanwal.

Sprint planning sigue igual que en Scrum. En cambio, Sprint Review & Retrospective fueron comentadas muy por encima, no hicimos los informes. Otro cambio importante fue que no había daily, un único desarrollador. Si el desarrollador tuviera alguna duda durante el Sprint, hablaría directamente con el director del proyecto.

Usamos un repositorio de GitHub como herramienta para el control de versiones, lo que nos permitió en todo momento la disponibilidad del código (estaba en la nube) y la recuperación de versiones anteriores en caso de fallos críticos. Todo el código que se ha desarrollado estará en la rama principal (*main branch*).

2. Planificación temporal

2.1. Introducción

El proyecto tendrá una duración aproximada de 568 horas, distribuidas en 142 días comenzando desde el 23 de febrero de 2021 hasta el 15 de septiembre de 2021. No

se había decidido la fecha para la defensa oral todavía. Por lo tanto, la fecha límite anterior era la más temprana que se puede tener. Estaba previsto trabajar 4 horas (en media, tres horas para desarrollo y una para documentar) aproximadamente todos los días laborales aunque los exámenes podían afectar debidamente este tiempo periódicamente.

"*Hace falta recordar que el TFG o TFM realizado en modalidad B se tiene que defender en la primera convocatoria de tribunal después de la expiración del convenio con la empresa en la que se ha realizado.*" [6]

El convenio con la empresa se acababa el 15 de septiembre del 2021, por lo tanto, tocaba defender en el turno de octubre del 2021.

2.2. Tareas

En general suele ser conveniente dividir el proyecto en paquetes de trabajo, ya que permite descomponerlo en partes claramente identificables. Cada una de estas partes puede descomponerse en actividades o tareas a realizar, interdependientes entre sí. Las actividades deben tener las siguientes características: [7]

- Ser mensurables en términos de tiempo, recursos, esfuerzo y coste
- Tener un producto final como resultado
- Tener un comienzo y un fin claro
- Ser responsabilidad de una sola persona

2.2.1. Tareas de gestión de proyecto

TG1 → Contextualización y alcance (25 horas): Documentación de la primera entrega de gestión del proyecto, donde había el contexto del proyecto, su justificación, el alcance y la metodología a que íbamos a usar.

TG2 → Planificación temporal (8 horas): Documentación de la segunda entrega de la gestión del proyecto, donde se encontraba la propuesta de planificación de tareas, su diagrama de Gantt y la gestión de los riesgos observados en la primera entrega. Dependencias: TG1.

TG3 → Presupuesto y sostenibilidad (9 horas): Documentación de la tercera entrega de la gestión del proyecto, donde se encontraban los presupuestos y el informe de sostenibilidad. Dependencias: TG2.

TG4 → Documentación de la fase inicial (19 horas): Documentación donde se recogieron las tres entregas anteriores y se les aplican los cambios recomendados en el feedback del tutor de GEP. Dependencias: TG3.

TG5 → Reunión con director (15 minutos): Reunión con el director, aproximadamente una vez a la semana, para ver el estado de las tareas. Como solo había un desarrollador, no hacía falta hacer daily scrum. Dependencias: Ninguna.

TG6 → Sprint Planning (2 horas): Cada dos semanas, nos reunimos para desgranar en tareas las historias de usuario que se pretenden realizar durante ese sprint. Dependencias: Ninguna.

TG7 → Documentación de la fase seguimiento (40 horas): Preparar la entrega y hacer la reunión de seguimiento con el ponente, Toni Urpí, para tener conocimiento del progreso y estado del proyecto. Dependencias: TG4.

TG8 → Documentación de la fase final (60 horas): Acabar de documentar todo el proyecto. Dependencias: TG7.

TG9 → Preparación de la lectura (40 horas): Preparación de la presentación del proyecto. Dependencias: TG8.

2.2.2. Tareas de desarrollo

TD1 → Comparación de proveedores (20 horas): Comparación de proveedores de Cloud Computing.

TD2 → ¿OCC usa alguna de estas plataformas? (3 horas): Recerca de si OCC había hecho algo en una de las plataformas de Cloud Computing.

TD3 → Reuniones con Marc Cardenas (3 horas): Marc Cardenas es un integrante del equipo de Sener que se encarga de llevar algunos servicios a la nube usando Azure. En caso de que íbamos a usar Azure, el objetivo era el de alinearme con él para configurar los servicios que íbamos usar.

TD4 → Base de datos en la nube o en Sener (5 horas): Estudiar cómo se podía tener una BD en la nube o en Sener. Estudiar la viabilidad de tener una BD con datos de FORAN (exportada en Oracle) en un servidor en la nube, o poder acceder desde el servidor de la nube a una BD local en Sener. Dependencias: TD1, TD2 y TD3.

TD5 → Pedir servicio y configurarlo (25 horas): Pedir servicio con el proveedor se había escogido y configurarlo. Dependencias: TD1, TD2 y TD3.

TD6 → Integrar base de datos de Foran (5 horas): Integrar base de datos existente del FORAN en la de nube. Dependencias: TD5.

TD7 → Web visor con OCC (15 horas): Integrar visor web 3D con OpenCascade. Dependencias: TD5.

TD8 → Formas de casco (15 horas): Leer datos de un fichero en la nube para mostrarlos en el visor. Específicamente, se leerán elementos representados en geometría exacta para poder ver las líneas del casco en el visor. Dependencias: TD7.

TD9 → Mostrar entidades 3D en el visor (30 horas): Leer entidades 3D de la base de datos y mostrarlo en el visor de OpenCascade. Dependencias: TD8.

TD10 → Crear entidades y guardarlas (50 horas): Ser capaz de crear entidades y guardarlas en base de datos. Si pasamos esas entidades en la base de datos de Foran, seríamos capaces de ver exactamente como las habíamos creado en la nube. Sincronizamos, seríamos capaces de utilizar la aplicación existente de Foran para leer entidades hechas desde el visor web. Dependencias: TD9.

TD11 → Sincronización de base de datos con el de Sener (30 horas): Estudiar las maneras en la que podríamos sincronizar BD con una de Sener o como tener directamente la BD en Sener. Dependencias: TD4.

TD12 → Añadir otras funcionalidades en visor (a determinar) (60 horas): Otras funcionalidades que nos podrían haber interesado añadirlas en el visor durante el proyecto.

2.3. Recursos

Recursos humanos: Shamsa Kanwal [SKW], sería la que estaría desarrollando el programa. Periódicamente, el director, Oscar Llorens [OLM] le iría dando feedback sobre los progresos con el trabajo. Toni Urpí [TU], el ponente, también me iría dando soporte y ayudando cuando fuera necesario. También tendría reuniones con Marc Cardenas, experto en pasar servicios a la nube, para orientarse.

Recursos materiales: Íbamos a necesitar el portátil con acceso al internet para poder desarrollar el programa. También un lugar para trabajar, que sería la casa de Shamsa.

En cuanto a software, necesitaríamos:

- Git: Herramientas para el control de versiones y repositorios.
- Herramientas de Microsoft [HM]: Teams, Outlook, Word, Excel, etc.
- Atenea: Fuente de donde sacar los recursos necesarios para la redacción y desarrollo de la parte del gestión del proyecto.
- Google Chrome [GC]: Navegador que usare tanto para buscar la información necesaria como para probar los desarrollos que lleve a cabo.
- Gantt Project [GP]: Herramienta con la que crear el diagrama de Gantt.
- Plataforma de la nube [X]: El proveedor que hayamos escogido después de las comparaciones.

2.4. Estimaciones y Gantt

En la Tabla 1 podemos observar las tareas que iba a llevar a cabo.

ID	Tarea	Horas	Dependencias	Recursos
TG1	Contextualización y alcance	25	-	SKW, OLM, TU, HM, Atenea, GC
TG2	Planificación temporal	8	TG1	SKW, OLM, TU, HM, Atenea, GC GP
TG3	Presupuesto y sostenibilidad	9	TG2	SKW, OLM, TU, HM, Atenea, GC
TG4	Documentación de la fase inicial	19	TG3	SKW, OLM, TU, HM, Atenea, GC
TG5	Reunión con director	0,25	-	SKW, OLM, HM

TG6	Sprint Planning	2	-	SKW, OLM, HM
TG7	Documentación de la fase seguimiento	40	TG4	SKW, OLM, TU, HM, Atenea, GC
TG8	Documentación de la fase final	60	TG7	SKW, OLM, TU, HM, Atenea, GC
TG9	Preparación de la lectura	40	TG8	SKW, HM, Atenea, GC
TD1	Comparación de proveedores	20	-	SKW, OLM, HM, GC
TD2	¿OCC usa alguna de estas plataformas?	3	-	SKW, OLM, HM, GC
TD3	Reuniones con Marc Cardenas	3	-	SKW, OLM, MC, HM, GC
TD4	Base de datos en la nube o en Sener	5	TD!, TD2, TD3	SKW, OLM, HM, GC
TD5	Pedir servicio y configurarlo	25	TD!, TD2, TD3	SKW, OLM, HM, GC, X
TD6	Integrar base de datos de Foran	5	TD5	SKW, OLM, HM, GC, X
TD7	Web visor con OCC	15	TD5	SKW, OLM, HM, GC, X, Git
TD8	Formas de cascos	15	TD7	SKW, OLM, HM, GC, X, Git
TD9	Mostrar entidades 3D en el visor	25	TD8	SKW, OLM, HM, GC, X, Git
TD10	Crear entidades y	50	TD9	SKW, OLM, HM, GC,

	guardarlas			X, Git
TD11	Sincronización de base de datos con el de Sener	40	TD4	SKW, OLM, HM, GC, X
TD12	Añadir otras funcionalidades en visor (a determinar)	60	TD9	SKW, OLM, HM, GC, X, Git

Tabla 1. Resumen de las tareas. Fuente: Elaboración propia

En la Figura 3 podemos observar el diagrama de Gantt para las tareas que habíamos propuestas sin tener en cuenta las reuniones.

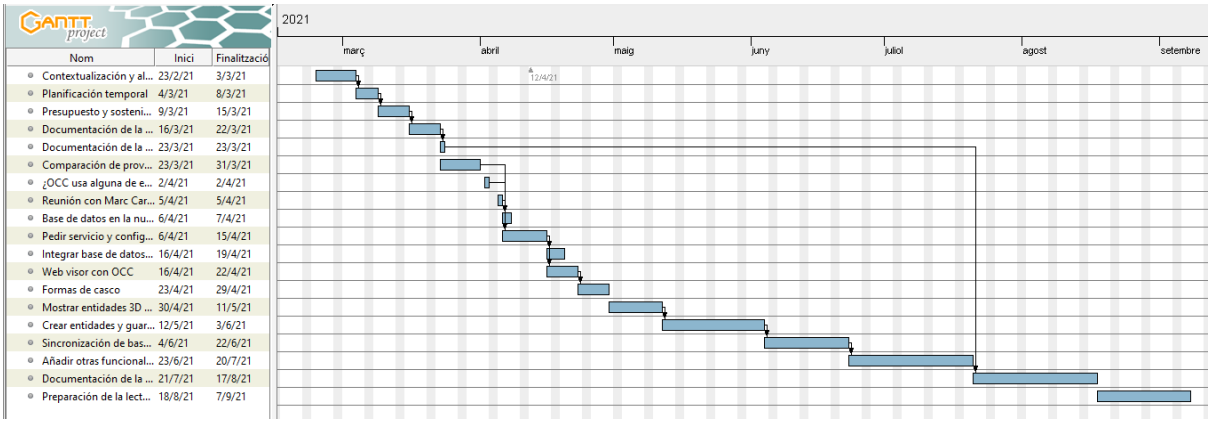


Figura 3. Diagrama de Gantt para las tareas propuestas sin tener en cuenta las reuniones. Fuente: Elaboración propia

2.5. Gestión del riesgo: Planes alternativos y obstáculos

Como ya habíamos visto anteriormente, el proyecto podía tener unos riesgos que podían retrasar la entrega prevista o, al menos, parte de ella. Sin embargo, teníamos que proponer posibles soluciones por si se produjera alguno de estos riesgos.

Para intentar minimizar los bugs haríamos muchas pruebas. Para no tener problema con la fecha de entrega, habíamos dejado margen de unas horas libres.

Para finalizar, estábamos viviendo en un momento de VUCA (Volátiles, inciertos, complejos y ambiguos), por lo tanto, debíamos tener en cuenta la posibilidad de que ciertas cosas pudieran suceder, ya que podría causar un impacto importante en el desarrollo del proyecto. El caso más probable era el posible confinamiento, pero aunque la reunión presencial con el ponente se podía ver afectada, ya se había demostrado en los últimos meses que podíamos adaptar adecuadamente y éramos capaces de sobrellevar cualquier incidencia desde casa, por lo que no supondría un impacto muy negativo y es poco probable que este riesgo supusiera retrasos en las entregas.

2.6. Cambios respecto la planificación inicial

En este apartado vamos a ver los cambios que hicimos respecto a la planificación inicial. Al final, tocó defender el proyecto el 28 de octubre del 2021. Así que la fecha límite también se extendió.

Algunas de las tareas se desgranaron, otras se quedaron igual y también surgieron nuevas por necesidad. En la Tabla 2, podemos observar el resumen de las tareas realmente desarrolladas con el tiempo dedicado.

ID	Tarea	Horas
TG1	Contextualización y alcance	25
TG2	Planificación temporal	8
TG3	Presupuesto y sostenibilidad	9
TG4	Documentación de la fase inicial	19
TG5	Reunión con el director	5
TG6	Sprint Planning	10
TG7	Documentación de la fase seguimiento	60
TG8	Documentación de la fase final	35
TG9	Preparación de la lectura	40
TD1	Comparación de proveedores	20

TD2	¿OCC usa alguna de estas plataformas?	3
TD3	Reuniones con Marc Cardenas	3
TD4	Base de datos en la nube o en Sener	7
TD5	Pedir servicio y configurarlo	35
TD6	Buscar Frameworks	5
TD7	Crear entorno de trabajo para Wt	25
TD8	Descargar 3rd party software para OCCT y compilar	20
TD9	Descargar código fuente de OCCT y compilar	10
TD10	Compilar WebGl que estan en OCCT/samples	10
TD11	Repetir de TD8-TD10 pero compilando con emscripten	20
TD12	Aprender Node.js	10
TD13	Usar Node.js en código por runtime stack en Azure	10
TD14	Create Oracle DB in VM on Azure	15
TD15	Recerca de como conectar base de datos en código	25
TD16	Descargar SQL Developer y conectar con DB creado en Azure	10
TD17	Descargar Oracle Client y configurarlo	10
TD18	Conectar Oracle DB que está en Azure en el código (integrar)	20
TD19	Integrar base de datos de Foran	10
TD20	Web visor con OCC (adaptado desde webgl)	10
TD21	Mejoras para objetos que son muy pequeños	10
TD22	Formas de cascos	15
Horas totales		514

Acabada Entremedia Descartada Pendiente

Tabla 2. Resumen de las tareas realmente desarrolladas. Fuente: Elaboración propia

Las tareas TD18 y TD19 que están en naranja porque no se han acabado del todo. El problema que tuve en TD18 es que el SQLAPI++ (API para conectar con base de datos) se tiene que pagar, hay unas librerías de esa API para pruebas. Pero en esas librerías no hay para emscripten. Habíamos creado otro mini proyecto que se compilaba con GNU. En este proyecto usamos las librerías de SQLAPI++ que

venían en prueba compiladas con GNU. Fuimos capaces de conectar, crear tablas y filas en esas tablas desde código.

En cuanto a la tarea TD19, me daban errores en el directorio donde quería hacer dump. Eso no me permitía hacer dump. Faltaba resolver esos errores.

2.6.1. Descripción de las tareas

En este apartado vamos a ver las tareas que son diferentes de la planificación inicial y donde surgió la necesidad de crearlas.

Teníamos claro que como back-end íbamos a usar C++ pero no habíamos decidido que Framework íbamos a usar y tampoco teníamos una tarea para la creación del entorno de desarrollo. Entonces, surgió la necesidad de crear unas nuevas tareas que eran TD6 y TD7.

TD6 Buscar Frameworks: Buscar que frameworks podríamos usar. Acabamos decido por Wt, C++ Web Toolkit. Está inspirada en Qt que usamos para FORAN desktop.

TD7 Crear entorno de trabajo para Wt: Crear proyecto con el Framework que se había decidido en la tarea anterior, es decir, Wt. Eso implica bajar y compilar las librerías necesarias para un proyecto de Wt, y también un entorno integrado de desarrollo.

Buscando información sobre unos errores que encontré al realizar la tarea TD7, acabé encontrando unos ejemplos de muestras en GitHub del Open CASCADE Technology. En ella había una muestra concreta, “webgl”, que era algo que nos interesaba ya que contenía un visor web. Así que, después de hablar con el director, habíamos quedado en usar esa muestra como punto de partida de nuestro proyecto.

De allí surgió la necesidad de crear las siguientes tareas.

TD8 Descargar 3rd party software para OCCT y compilar: Para poder compilar la muestra y OCCT, necesitábamos descargar unos softwares externos al código de OCCT. Entonces en esta tarea íbamos a descargar esos softwares y compilarlos.

TD9 Descargar código fuente de OCCT y compilar: Ya teníamos software externos, entonces, podíamos bajar el código de OCCT desde Git y compilarlo.

TD10 Compilar WebGL que están en OCCT/samples: Compilar la muestra que son interesaba, WebGL.

Al realizar la tarea TD10, nos habíamos encontrado con el problema de que tanto el código OCCT como software externo se tenía que compilar con emscripten. Por lo tanto, teníamos una nueva tarea, TD11.

TD11 Repetir de TD8-TD10 pero compilando con emscripten: Compilar software externo, código fuente OCCT y la muestra OpenGL con emscripten.

La muestra de OCCT, WebGL, no era un código que podíamos subir al Azure directamente. Para eso hicimos uso de Node.js. Node.js es uno de los runtime stack que Azure permite. Habíamos decido por Node.js porque al compilar con emscripten, el código del dominio y de base de datos se transpile en javascript. Se crearon las siguientes tareas.

TD12 Aprender Node.js: Nunca habíamos trabajado con Node.js, pues había que aprender.

TD13 Usar Node.js en código por runtime stack en Azure: Añadir código Node.js necesarios en la muestra WebGL para poder subirlo al Azure.

Azure no permite crear una base de datos directamente, en cambio, deja crearla en Virtual Machine. De allí la siguiente tarea.

TD14 Create Oracle DB in VM on Azure: Crear una base de datos Oracle en Virtual Machine de Azure.

Nunca habíamos conectado con una base de datos desde código. Así que tuvimos que hacer una recerca de cómo conectarlo. Por eso tenemos tarea TD15.

TD15 Recerca de como conectar base de datos en código: Recercar maneras de conectar con base de datos desde código, como usarlo, etc. ¿Si existe algún api?

TD16 Descargar SQL Developer y conectar con DB creado en Azure: Había que descargar SQL Developer. Eso nos permitía gestionar la base de datos y ver si se había creado bien en el Virtual Machine (conectividad).

Con la tarea TD15, habíamos encontrado que una manera de conectar con base de datos desde código es usar Oracle Client, concretamente, el fichero tnsnames.ora para conectar con la base de datos creada.

TD17 Descargar Oracle Client y configurarlo: Descargar Oracle Client y en tnsnames.ora añadir una nueva conexión con base de datos pasandole adresa IP, puerto y OID del base de datos creada en maquina virtual.

TD18 Conectar Oracle DB que está en Azure en el código (integrar): No conseguía conectar con base de datos con tnsnames.ora, así que habíamos buscado más información y al final, habíamos acabado encontrado una manera mucha directa de conector con base de datos.

TD20 Web visor con OCC (adaptado desde webgl): Hacer los cambios necesarios en WebGL para que quedé como queríamos, es decir, el viewer ocupe toda la pantalla, cámara ortogonal, entre otras cosas.

Cuando estábamos leyendo objetos para pintarlo en viewer, parecía como si no se estuviera leyendo o a lo mejor no estaba pintando en viewer 3D de OpenCascade. De allí la tarea TD21.

TD21 Mejoras para objetos que son muy pequeños: Intentando encontrar el error comentado anteriormente, habíamos encontrado que el problema era que los objetos eran tan pequeños que comparado con las formas del casco no se veían nada. Por lo tanto, había que hacer unas mejoras en ese aspecto.

3. Gestión económica y sostenibilidad

3.1. Introducción

En este apartado, se describieron los elementos a considerar al realizar la estimación presupuestaria, que incluye costos de personal por tarea, costos genéricos entre otros. Finalmente, responderemos algunas preguntas sobre el aspecto de la sostenibilidad en el proyecto.

3.2. Presupuesto

3.2.1. Presupuesto del personal

En esta sección se calcula el costo total de cada tarea definida en la segunda entrega. El costo de una tarea se calcula sumando el costo del trabajo del personal. El costo de cada trabajador se calcula multiplicando su costo por hora por la cantidad de tiempo que estuvo involucrado en la actividad.

En este proyecto existían 2 tipos de personal, cada uno con un costo diferente por hora. En primer lugar, el director del proyecto era responsable de la planificación y correcto desarrollo del proyecto. El tutor de GEP, el ponente, el director y Shamsa jugaron este papel. En segundo lugar, el desarrollador que programa el código.

Rol	Salario/hora (bruto)	Salario/hora (bruto) + Seg Social (30%)
Director del proyecto	22,4€	**29,12€**
Desarrollador becario	9€	**11,7€**

Tabla 3: Salario bruto del personal por hora. Fuente: [8]

ID	**Tarea**	Horas totales	Gestor del proyecto		Desarrollador becario	Coste (€)
			Tutor/ director	Desarrollador becario		
TG1	Contextualización y alcance	25	1	24	0	309,92
TG2	Planificación temporal	8	1	7	0	111,02
TG3	Presupuesto y sostenibilidad	9	1	8	0	122,72
TG4	Documentación de la	19	2	17	0	257,14

	fase inicial					
TG5	Weekly Scrum	14	7	7	7	367,64
TG6	Sprint Planning	56	28	28	28	1470,5
TG7	Documentación de la fase seguimiento	40	0	0	40	468
TG8	Documentación de la fase final	60	0	0	60	702
TG9	Preparación de la lectura	40	0	40	0	468
TD1	Comparación de proveedores	20	0	0	20	234
TD2	¿OCC usa alguna de estas plataformas?	3	0	0	3	35,1
TD3	Reuniones con Marc Cardenas	3	0	0	3	35,1
TD4	DB en la nube/Sener	5	0	0	5	58,5
TD5	Pedir servicio y configurarlo	25	0	0	25	292,5
TD6	Integrar DB Foran	5	0	0	5	58,5
TD7	Web visor con OCC	15	0	0	15	175,5
TD8	Formas de cascos	15	0	0	15	175,5
TD9	Mostrar entidades 3D en el visor	25	0	0	25	292,5
TD10	Crear entidades y	50	0	0	50	585

	guardarlas					
TD11	Sincronización de base de datos con el de Sener	40	0	0	40	468
TD12	Añadir otras funcionalidades en visor (a determinar)	60	0	0	60	702
Total						**7389,2**

Tabla 4: Coste del personal junto con sus horas. Fuente: Elaboración propia

3.2.2. Costos generales

3.2.2.1. Amortización

Uno de los aspectos a tener en cuenta es la amortización de los recursos materiales utilizados en el proyecto. Se consideraba una media de 4 horas de trabajo al día durante un total de 142 días. Se estimó que todo el proyecto se iba a llevar a cabo usando la computadora portátil HP que costó 500€ [9]. Hay que tener en cuenta que ese mismo portátil íbamos a usarlo para otras cosas, por lo tanto, en total fueron, aproximadamente, unas 10 horas al día. En la Figura 4 podemos observar la ecuación para calcular la amortización del recurso y la amortización del portátil que fueron **19,45€**.

$$\text{Amortización (euros)} = \text{Precio recurso} \times \frac{1}{4\ \text{Años}} \times \frac{1}{365\ \text{Días}} \times \frac{1}{10\ \text{Horas}} \times \text{Horas usadas}$$

$$\text{Amortización (euros)} = 500 \times \frac{1}{4\ \text{Años}} \times \frac{1}{365\ \text{Días}} \times \frac{1}{10\ \text{Horas}} \times 568\ \text{Horas} = 19{,}45\ \text{euros}$$

Figura 4: Fórmula para calcular la amortización del recurso. Fuente: Elaboración propia

3.2.2.2. Internet

La tarifa de Internet costaba 30€ al mes [10]. Teniendo en cuenta que el proyecto iba a durar 7 meses y eran 4 horas al día, el coste de internet iba a ser 30 € * 7 meses * (4 horas / 24 horas) = **35€**.

3.2.2.3. Coste de viaje

En principio, solo estaba previsto viajar una vez para la reunión con el ponente para el seguimiento del proyecto y otra para la lectura. Las reuniones con el director del proyecto iban a ser vía Microsoft Teams. Por lo tanto, la idea era comprar el T-Casual [11] de una zona, que costaba **11,35€** y eran 10 viajes.

3.2.2.4. Lugar del trabajo

El proyecto se desarrolló desde casa de Shamsa. Entre lo que costaba y como no ocupaba ella sola, costaba unos 250€ al mes (incluso la electricidad y agua). El proyecto iba a durar unos 7 meses, por lo tanto, **1750€** en total.

3.2.2.5. Resumen del coste general

A continuación podemos observar el resumen de los costes generales:

Amortización → 19,45€

Internet → 35€

Coste de viaje → 11,35€

Lugar de trabajo → 1750€

Total → **1815,8€**

3.2.3. Otros costos

3.2.3.1. Contingencias

Era muy probable que en el desarrollo del proyecto pudieran aparecer imprevistos, que formaban parte de nuestro presupuesto. Por este motivo, era necesario haber preparado un fondo de contingencia para estar preparados para enfrentar estos eventos. Habíamos agregado un 15% de margen de contingencia sobre el costo total. Por lo tanto, el fondo de contingencia era (7389,2 + 1815,8) * 0,15 = **1380,75€.**

3.2.3.2. Costo de incidentes

También teníamos que tener en cuenta el coste de aplicar planes alternativos en caso de que ocurrieran eventos inesperados durante el curso del proyecto. Podemos observar en la Tabla 5 los posibles incidentes que podíamos haber encontrado. El coste era el coste estimado por el riesgo.

Incidente	Costo estimado (€)	Riesgo (%)	Coste (€)
Bugs (25 horas)	225	100	225
Llevar más tiempo del estimado (30 horas)	270	50	135
Inexperiencia en la tecnología (20 horas)	180	50	90
Total			**450**

Tabla 5: Posibles incidentes con costes aproximados. Fuente: Elaboración propia

3.2.4. Coste Total

A continuación encontramos el resumen de los costes anteriormente calculados.

Coste Personal → 7389,2€

Coste general → 1815,8€

Contingencias → 1380,75€

Coste de incidentes → 450€

Total → 11035,75€

3.2.5. Control de gestión

Veamos como se había previsto llevar el control sobre las posibles desviaciones en el presupuesto que podíamos haber encontrado a lo largo del proyecto. Para ello, íbamos a intentar anotar el tiempo real que nos había llevado desarrollar cada tarea. De este modo, podríamos saber la desviación que teníamos en nuestro presupuesto y si estábamos dentro de lo que se ha presupuestado en un inicio. Íbamos a tener control sobre las siguientes métricas:

$$Desvío\ de\ mano\ de\ obra\ en\ precio\ =\ (Coste\ estimado\ -\ Coste\ real)\ *\ Consumo\ horas\ real$$

$$Desvío\ de\ mano\ de\ obra\ en\ consumo\ =\ (Consumo\ estimado\ -\ Consumo\ real)\ *\ Coste\ estimado$$

Al finalizar el periodo del proyecto estaba previsto realizar los cálculos para obtener los desvíos. Con esto podríamos hacer un análisis del presupuesto y determinar si, necesitaríamos hacer uso de la contingencia calculada al inicio del proyecto o no.

3.3. Sostenibilidad

3.3.1. Autoevaluación

Antes de entrar en la carrera, cada vez que escuchaba la palabra sustentabilidad pensaba que era sinónimo de no dañar el medio ambiente, que equivocada estaba. Después de entrar en la carrera me he dado cuenta de que existen 3 dimensiones: social, económica y medioambiental. Todas son igual de importantes.

Reflexionando sobre mis conocimientos sobre la sostenibilidad, después de hacer la encuesta y pensar en el aspecto de la sostenibilidad en un proyecto, he llegado a la conclusión de que tenía una idea sobre eso. Pero me falta profundizar bastante en ese tema. Finalmente, me ha hecho pensar en la importancia de tener en cuenta la sostenibilidad a la hora de planificar y desarrollar un proyecto desde el punto de vista de las tres dimensiones diferentes, si no tenemos en cuenta, el proyecto puede verse afectado negativamente.

3.3.2. Dimensión económica

PPP: ¿Habías estimado el coste de la realización del proyecto (recursos humanos y materiales)?

Como yo no formaba parte del equipo encargado del presupuesto, había realizado un presupuesto basándose en los salarios y costes medios que había encontrado en Internet, así como de las tareas que íbamos a realizar y el tiempo que nos llevaría desarrollarlas.

Vida Útil: ¿Cómo se resolvía el problema que querías abordar (estado del arte)? En que iba a mejorar económicamente tu solución a las existentes?

El programa FORAN solo se podía acceder de una sola manera determinada. Íbamos a mejorar económicamente ya que no tendríamos que tener tantos equipos como lo necesitábamos antes (servidor con una base de datos, un servidor de licencias y un servidor de control de usuarios/permisos; además, cada usuario debe

disponer de un equipo de alta prestación y preferiblemente con tarjeta gráfica dedicada).

3.3.3. Dimensión ambiental

PPP: ¿Habías estimado el impacto ambiental que tendría la realización del proyecto? ¿Te habías planteado minimizar el impacto, por ejemplo, reutilizando recursos?

Necesitaríamos menos equipo, como ya comenté anteriormente. Estaríamos compartiendo equipo con terceros ya que sería equipo del Cloud, por lo tanto, reutilizando recursos.

Vida Útil: ¿Cómo se resolvía el problema que querías abordar (estado del arte)? ¿En qué mejoraría ambientalmente tu solución a las existentes?

Antes para usar el programa FORAN necesitábamos mucho equipamiento, por lo tanto, cuando se les acababa la vida útil había que tirar estos y comprar nuevos. Con la nueva solución necesitaríamos menos equipo, por lo tanto, menos daño al ambiente.

3.3.4. Dimensión social

PPP: ¿Qué creías que te va a aportar a nivel personal la realización de este proyecto?

A nivel personal creía que este proyecto me podía enriquecer en muchos sentidos, no solamente con conocimientos sobre el desarrollo de software, sino también en aprender cómo funcionaba un equipo de desarrollo en una empresa y cómo aplicar metodologías ágiles.

Vida Útil: ¿Cómo se resolvía actualmente el problema que quieres abordar (estado del arte)? En qué mejoraría socialmente (calidad de vida) tu solución a las existentes?

Los usuarios tendrían distintas maneras de acceder al programa.

Vida Útil: ¿Existe una necesidad real del proyecto?

En su momento, no había una necesidad real ya que teníamos el programa en desktop. Pero en un futuro, por muchos muchos motivos, nos podría interesar tener el programa en la nube. Por eso íbamos a crear este prototipo.

4. Arquitectura del sistema

4.1. Visión global

El hosting de la web se hace en Azure. Teníamos un Web App en la Azure y una máquina virtual que contenía base de datos Oracle. En la figura 5 podemos observar la arquitectura del sistema.

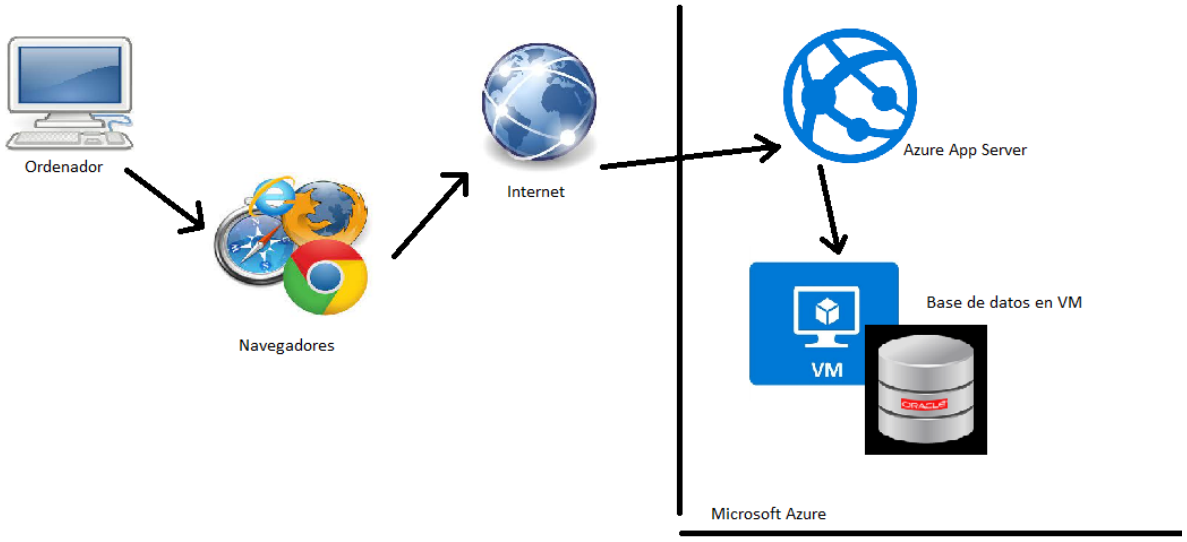


Figura 5: Arquitectura del sistema. Elaboración propia

4.2. Arquitectura de 3 capas

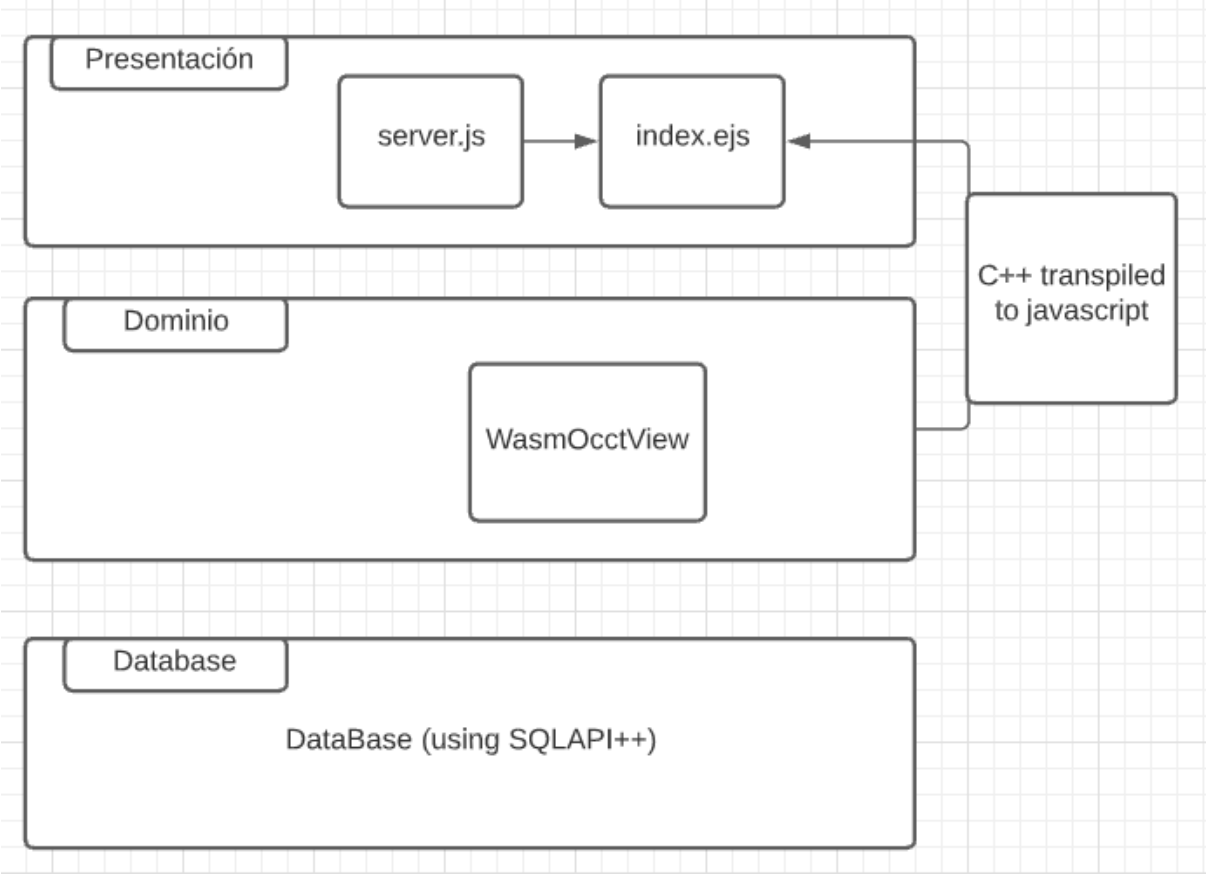


Figura 6: Arquitectura de 3 capas, con pequeños detalles. Elaboración propia

Había usado la arquitectura de 3 capas que es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente-servidor: capa de presentación, lógica de negocios y capa de datos. De esta forma, por ejemplo, es sencillo y mantenible crear diferentes interfaces sobre un mismo sistema sin requerir cambio alguno en la capa de datos o lógica.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo afectará al nivel requerido sin tener que revisar entre el código fuente de otros módulos, dado que se habrá reducido el Acoplamiento informático hasta una interfaz de paso de mensajes.

[12]

4.3. Diagrama de clases

El diagrama de clases sólo teníamos una clase aparte del Main. Como ya comenté anteriormente, había utilizado como base el código de la muestra de WebGL. Entonces solo he modificado clases que ya existían en la muestra.

Como había comentado en el apartado de requisitos no-funcionales, no nos interesaba tener un buen código ni escalable. Porque al final, solo nos interesaba saber que éramos capaces de hacer en la nube. Así que no puse mucho esfuerzo y modifique la existente.

En general, habíamos escrito más código en la capa de presentación que en otras.

4.4. Patrones usados

Igual que antes, no puse mucho esfuerzo y simplemente usé un único patrón que era Singleton. Lo usé para establecer una única conexión con base de datos.

Singleton

Es un patrón de diseño que se utiliza para restringir la instancia de una clase a un objeto. Esto es útil cuando se necesita exactamente un objeto para coordinar acciones a nivel de todo el sistema o programa. [13]

4.5. API usados

Opencascade Technology (OCCT)

La tecnología Open CASCADE (OCCT) es la única librería de geometría 3D de código abierto a gran escala. Esforzándose por ser uno de los mejores núcleos de software CAD gratuitos, OCCT se utiliza ampliamente para el desarrollo de programas especializados que se ocupan de los siguientes dominios mecánicos y

de ingeniería: modelado 3D (CAD), fabricación (CAM), simulación numérica (CAE), equipos de medición. (CMM) y control de calidad (CAQ). Desde su publicación en 1999 como un núcleo de software CAD de código abierto, OCCT se ha utilizado con éxito en numerosos proyectos que van desde la edificación y la construcción hasta la industria aeroespacial y automotriz. [14]

Es el API que nos permitirá el uso de OpenCascade Viewer y hacer cosas sobre ella.

Emscripten

Permite transformar el código en C++ a javascript.

SQLAPI++

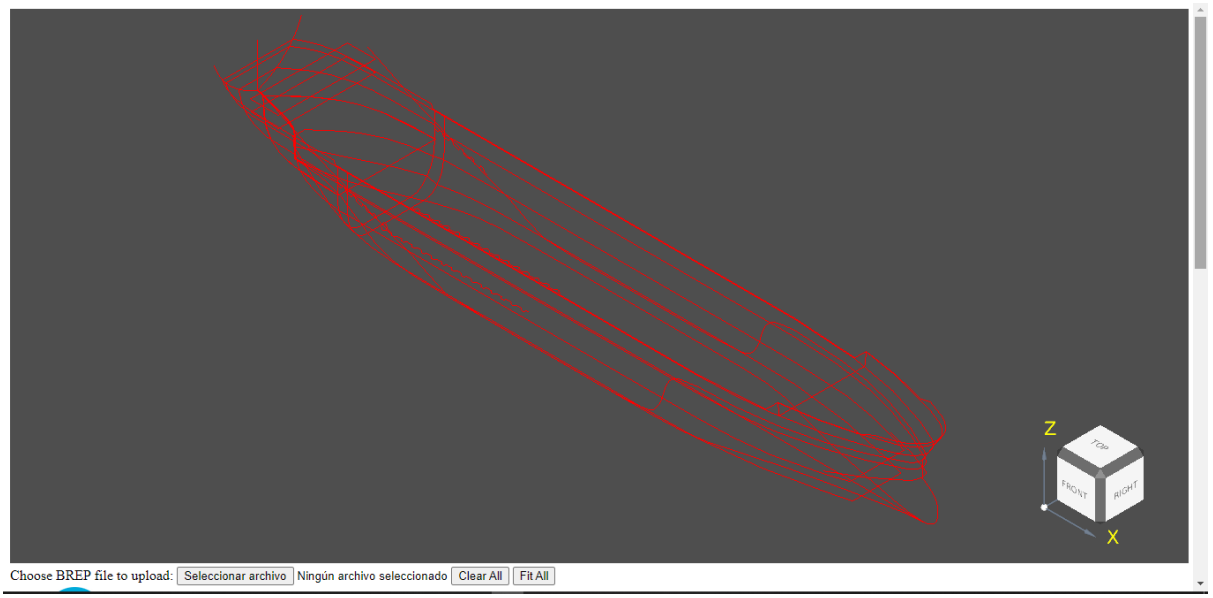
Es una API que nos permite establecer la conexión con base de datos. Por lo tanto, leer, modificar, eliminar, crear, etc en base de datos. No es gratis.

4.6. Diseños de base de datos

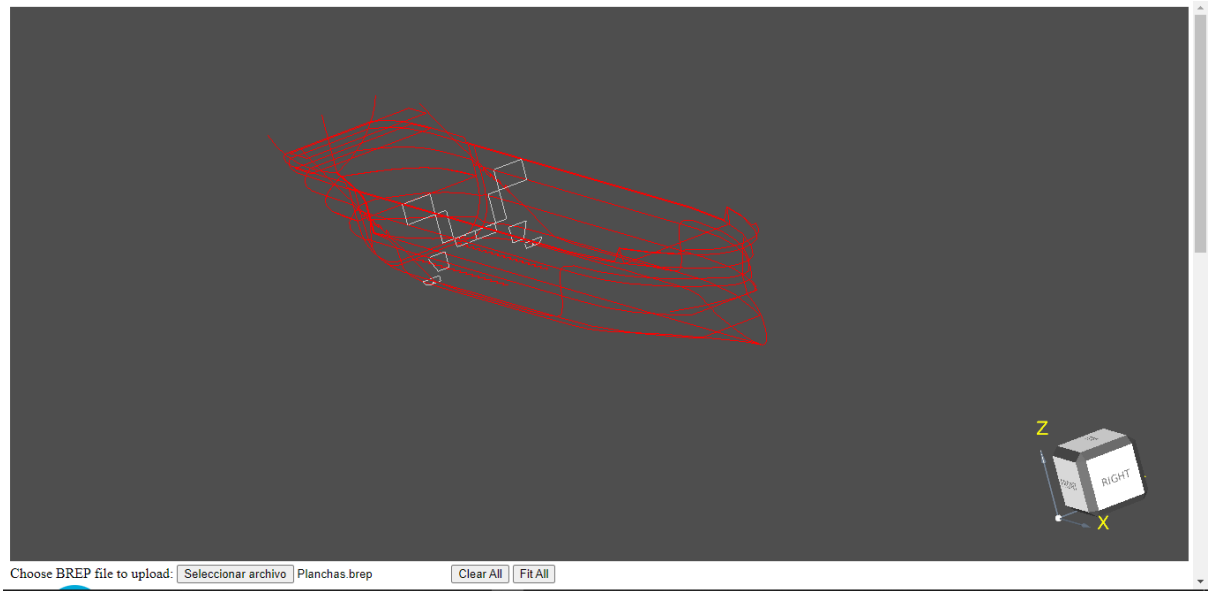
No había diseño de base de datos. Iba a hacer dump de la base de datos de Foran, si me daba tiempo. Como se había comentado anteriormente, no se ha podido completar por unos errores.

4.7. Diseño de interficie

Solo hay una pantalla que contiene los botones para subir los breps (es un formato), eliminar entidades de la escena (formas de casco no se eliminan) y, por último, reajustar la escena para que se vean todas las entidades. No hay menú porque no habíamos podido implementar la parte de creación de planchas.



Captura 4.1 Formas de casco en web viewer. Elaboración propia



Captura 4.2 Formas de casco con planchas (Wireframe) en web viewer. Elaboración propia

Captura 4.3 Una bola 3D sólida en web viewer. Elaboración propia

5. Implementación

5.1. Tecnologías y lenguajes empleados

A continuación vamos a ver software y lenguajes usados en el proyecto.

Git

Git es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia. [15]

Python

Python es un lenguaje de programación que te permite trabajar rápidamente e integrar los sistemas de forma más eficaz. [16]

Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. [17]

Emscripten

Emscripten es un completo compilador toolchain a WebAssembly, que utiliza LLVM, con un enfoque especial en la velocidad, el tamaño y la plataforma web. [18]

C++

C++ es un lenguaje de programación orientado a objetos muy potente que evolucionó de la extensión de lenguaje informático “C” y que hoy en día sigue usándose para realizar programación estructurada de alto nivel y rendimiento, como sistemas operativos, videojuegos y aplicaciones en la nube. [19]

5.2. Herramientas de desarrollo

Visual Studio Code

Visual Studio Code es un editor de código redefinido y optimizado para crear y depurar aplicaciones web y en la nube. [20]

Portal Azure

Portal Azure es una consola unificada basada en web que proporciona una alternativa a las herramientas de línea de comandos. Con Azure Portal, puede administrar su suscripción a Azure mediante una interfaz gráfica de usuario. [21]

SQL Developer

Oracle SQL Developer es un entorno de desarrollo integrado y gratuito que simplifica el desarrollo y la gestión de Oracle Database en implementaciones tradicionales y en la nube. SQL Developer ofrece un desarrollo completo de un extremo a otro de sus aplicaciones PL / SQL, una hoja de trabajo para ejecutar consultas y scripts, una consola DBA para administrar la base de datos, una interfaz de informes, una solución completa de modelado de datos y una plataforma de migración para mover su Bases de datos de terceros para Oracle. [21]

5.3. Despliegue

Habíamos desplegado en Azure Portal. Usando la opción de Web App para la página web.

6. Pruebas

No hice las pruebas automáticas. Lo que sí hice, fue hacer unas pruebas como si fuera el usuario final.

7. Logro de las competencias técnicas

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y / o críticos. [Bastante]: La idea era desarrollar un software de CAD. Había que usar unas cuantas APIs.

CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles. [En profundidad]: Habíamos dado una nueva solución que era tener software en nube.

CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que se pudieran presentar. [Algo]: Hay un apartado en la documentación que comenta los posibles riesgos que podíamos haber encontrado.

CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de red. [Bastante]: Estábamos usando computación en la nube que funciona sobre red.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. [Bastante]: Habíamos hecho pruebas como usuario final y unas directamente sobre el código para comprobar ciertas partes.

CES3.1: Desarrollar servicios y aplicaciones multimedia. [Bastante]: Al final, teníamos un Viewer del OpenCascade que cuenta como multimedia.

8. Proveedor de nube escogido

8.1. Comparativa

En este apartado vamos a ver la comparativa entre Microsoft Azure y AWS (Amazon Web Service). En la tabla 6 podemos observar la tabla comparativa.

	Microsoft Azure	**AWS**
Computación	Un servicio básico y fundamental en la nube es la capacidad de cálculo o proceso. Cualquiera de los dos ofrecen distintos tipos de instancias, tanto basadas en Windows como en Linux, con GPUs o con configuraciones de gran tamaño y alto rendimiento.	
	Computación confidencial de Azure: aseguran que los datos críticos de la empresa estén protegidos mientras están en uso, aprovechando la infraestructura, las herramientas y el SDK confidenciales líderes de Azure. [23] “Usted es el propietario de sus datos Siéntase seguro en cuanto a dónde se almacenan sus datos. Nuestro principio de privacidad	Usan un modelo de seguridad blindado, prohíben cualquier tipo de acceso administrativo, incluido el realizado por empleados de Amazon, lo que evita la posibilidad de que ocurran errores humanos y manipulaciones. [24]

	fundamental es que usted es el propietario de sus datos y nunca los usamos con fines publicitarios o de marketing.” [25]	
Tipo de conexión	ExpressRoute	Direct Connect
Suporte para nube híbrida	Con Hybrid Cloud, las organizaciones pueden integrar servidores en el sitio con instancias de Cloud.	No ofrecen lo mejor en soporte de nube híbrida.
Zonas de disponibilidad	60+ regiones 140 países [26]	25 regiones 81 zones [27]
Base de datos Oracle	Dejan crear base de datos en máquina virtual	Base de datos
Herramientas	Dejan elegir el lenguaje y las herramientas que prefiramos. “Compile, depure, implemente y administre aplicaciones con el lenguaje o la plataforma que prefiera. Y aproveche la compatibilidad integrada con Azure de los entornos de desarrollo integrado más populares en los que confían más de 20 millones de desarrolladores: Visual Studio y Visual Studio Code.” [28]	Las herramientas para desarrolladores de AWS están diseñadas para trabajar junto con AWS, lo que hace más fácil para su equipo realizar las configuraciones y ser productivo. [29]
Seguridad	Proporcionan seguridad al ofrecer permisos en toda la cuenta.	La seguridad se proporciona mediante roles definidos con función de control de permisos.

Tabla 6: Comparativa entre Azure y ASW. Fuente: Elaboración propia

8.2. ¿OCC usa alguna de estas plataformas?

No había encontrado ninguna información sobre opencascade usando alguna plataforma de nube. Pero había encontrado que CAD Exchanger usa Microsoft Azure.

CAD Exchanger es la solución de escritorio/móvil para visualización/conversión 3D destinada a abordar los desafíos de interoperabilidad de datos CAD. El software permite a los usuarios visualizar datos 3D, convertirlos en una amplia gama de formatos CAD, construir mallas computacionales para análisis de ingeniería, mostrar propiedades de modelos 3D y más. La lista de formatos compatibles crece constantemente e incluye tanto formatos neutros (IGES, STEP, STL, JT, VRML, X3D, OBJ) como modelos específicos del kernel (ACIS, Parasolid, Opencascade, Rhino / Open NURBS). [30]

8.3. Base de datos en la nube o en Sener

Azure permitía ambas opciones:

En nube: Creando base de datos de Oracle en máquina virtual de Azure. En el apartado del “Manual” está explicado como se puede crear.

En Sener (o propio): Conectar/Establecer la conexión en código usando el IP y puerto junto con service name.

	Implementación local	Implementación en Azure
Redes	LAN/WAN	SDN (Redes definidas por software)
Grupo de seguridad	Herramientas de restricción de IP y puerto	Grupo de seguridad de red (NSG)
Resistencia	MTBF (tiempo medio entre errores)	MTTR (tiempo medio para recuperación)
Mantenimiento planeado	Aplicación de revisiones/actualizaciones	Conjuntos de disponibilidad (aplicación de revisiones o actualizaciones administradas por Azure)
Recurso	Dedicado	Compartido con otros clientes
Regiones	Centros de datos	Pares de región
Storage	SAN/discos físicos	Almacenamiento administrado por Azure
Escala	Escalado vertical	Escalado horizontal

Figura 7: Comparativa entre usando Base de datos local o en Azure. Fuente [31]

8.4. Resultado

Mirando la comparativa del apartado anterior, habíamos visto que Azure y AWS estaban más o menos empatados. Lo único que les diferenciaba bastante, era mirando nuestros requisitos no-funcionales, es el hecho de que Azure era más bueno que AWS para las nube híbridas. Nos podía llegar a interesar tener base de datos en local, ya que este contiene datos delicados que no queríamos que salgan fuera. Aunque los proveedores ofrecían una buena seguridad. Lo más probable es que a los clientes de Foran no les haga gracias tener datos en nube.

Otro punto a tener en cuenta era que Sener ya estaba subiendo unos servicios en la Azure.

Por lo tanto, por la comparativa y los sub-apartados anteriores, habíamos decidido que íbamos a usar el proveedor Microsoft Azure.

9. Manual

Creamos una cuenta de Azure y una subscripción. En mi caso será Azure For Students, es gratis.

9.1. Software necesario para desarrollo

Git

Git es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia. [15]

Podemos descargarlo de la web oficial de Git, https://git-scm.com/downloads.

Python

Python es un lenguaje de programación que te permite trabajar rápidamente e integrar los sistemas de forma más eficaz. [16]

Vamos a necesitar 3.6 o más nuevo. Podemos descargarlo de la web oficial de Python, https://www.python.org/downloads/.

Emscripten

Emscripten es un completo compilador toolchain a WebAssembly, que utiliza LLVM, con un enfoque especial en la velocidad, el tamaño y la plataforma web. [18]

Podemos descargarlo, necesitamos tener descargado el Git y Python. Vamos a ejecutar las siguientes instrucciones en Terminal/CMD [32]:

$> \ git\ clone\ https://github.com/emscripten-core/emsdk.git$
$> \ cd\ emsdk$
$> \ git\ pull$
$> \ emsdk\ install\ latest$
$> \ emsdk\ activate\ latest\ --permanent$

Node

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. [17]

Emscripten SDK (emsdk) ya contiene Node.

Visual Studio Code

Visual Studio Code es un editor de código redefinido y optimizado para crear y depurar aplicaciones web y en la nube. [20]

Podemos descargarlo de la web oficial de Visual Studio Code, https://code.visualstudio.com/download.

Portal Azure

Portal Azure es una consola unificada basada en web que proporciona una alternativa a las herramientas de línea de comandos. Con Azure Portal, puede administrar su suscripción a Azure mediante una interfaz gráfica de usuario. [21]

Podemos descargarlo de la web oficial de Portal Azure o usar la versión Web, https://portal.azure.com/.

SQL Developer

Oracle SQL Developer es un entorno de desarrollo integrado y gratuito que simplifica el desarrollo y la gestión de Oracle Database en implementaciones tradicionales y en la nube. SQL Developer ofrece un desarrollo completo de un extremo a otro de sus aplicaciones PL / SQL, una hoja de trabajo para ejecutar consultas y scripts, una consola DBA para administrar la base de datos, una interfaz de informes, una solución completa de modelado de datos y una plataforma de migración para mover su Bases de datos de terceros para Oracle. [22]

Podemos descargarlo de la web oficial de Oracle, https://www.oracle.com/tools/downloads/sqldev-downloads.html.

Mingw-w64

Mingw-w64 es un avance del proyecto original mingw.org, creado para admitir el compilador GCC en sistemas Windows. Lo ha bifurcado en 2007 para proporcionar soporte para 64 bits y nuevas API. Desde entonces ha ganado un uso generalizado y distribuido. [33]

Podemos descargarlo desde http://www.mingw-w64.org/doku.php/download/mingw-builds. Ayuda para instalar https://www.eclipse.org/4diac/documentation/html/installation/minGW.html.

En la carpeta bin del Mingw-w64, copiamos mingw32-make.exe y cambiamos nombre a make.exe.

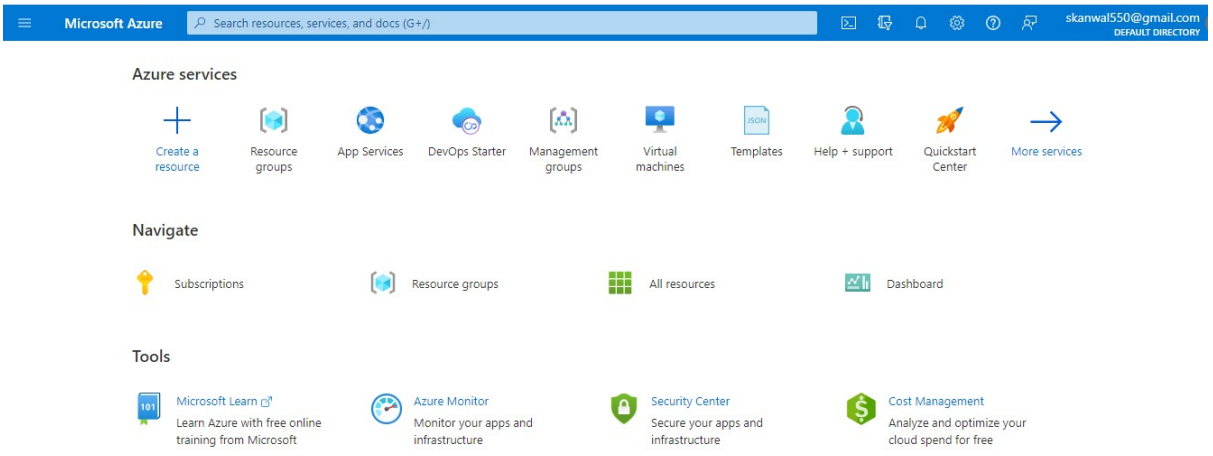
CMake

CMake es una familia de herramientas multiplataforma de código abierto diseñada para construir, probar y empaquetar software. CMake se utiliza para controlar el proceso de compilación de software mediante una plataforma simple y archivos de configuración independientes del compilador, y generar archivos MAKE nativos y espacios de trabajo que se pueden usar en el entorno de compilación de su elección. [34]

Podemos descargarlo de la web oficial de CMake , https://cmake.org/download/.

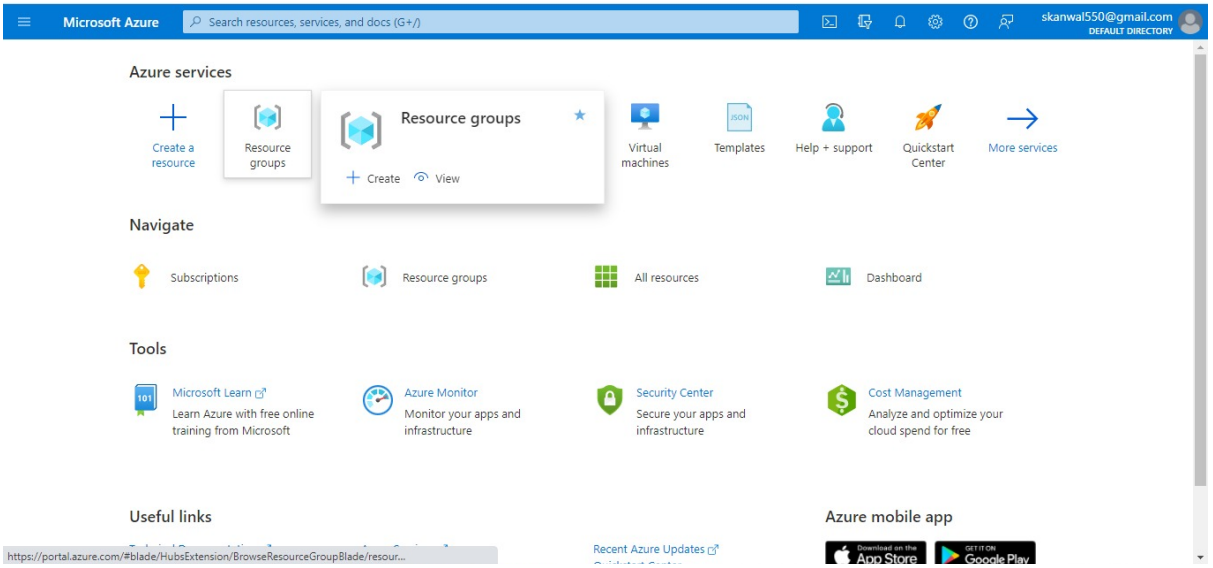
9.2. Crear Web App

Vamos a ir al Portal de Azure, https://portal.azure.com/. Entremos con nuestra cuenta de Azure.

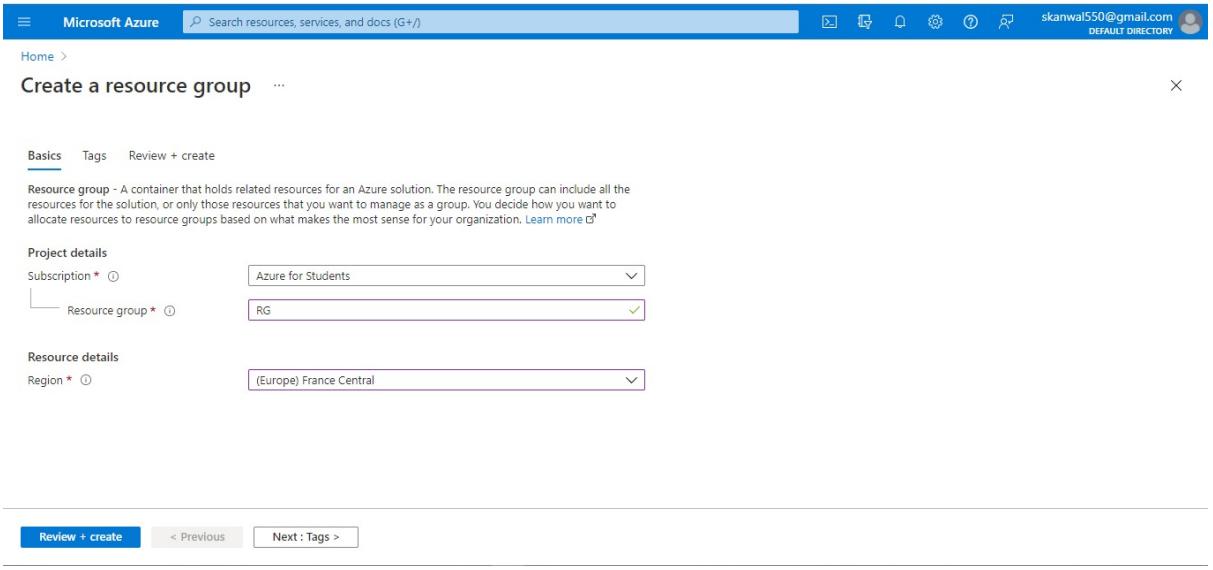


Captura 9.1: Portal de Azure.

Creamos Resources Group. Necesitamos especificar la suscripción, nombre del resources group y la región donde queremos que esté.

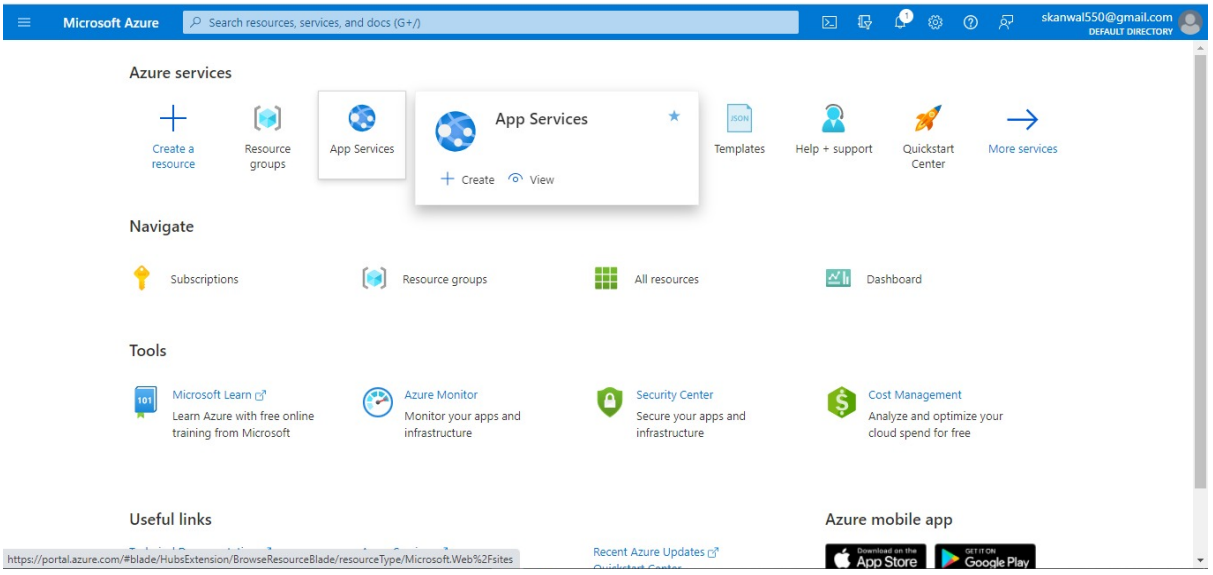


Captura 9.2: Opción de crear Resources Group

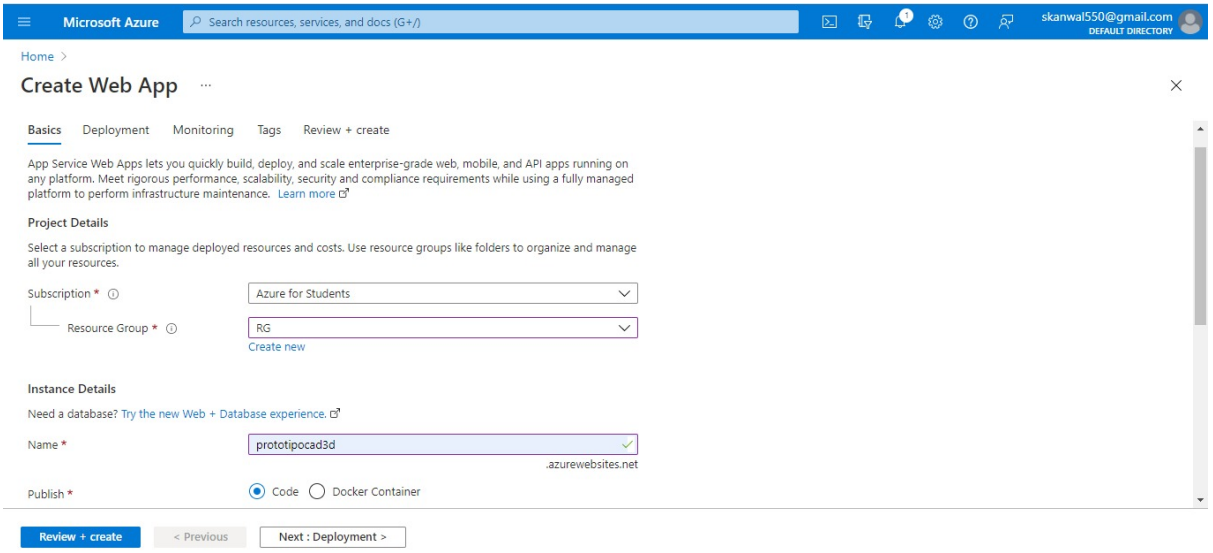


Captura 9.3: Crear Resources Group.

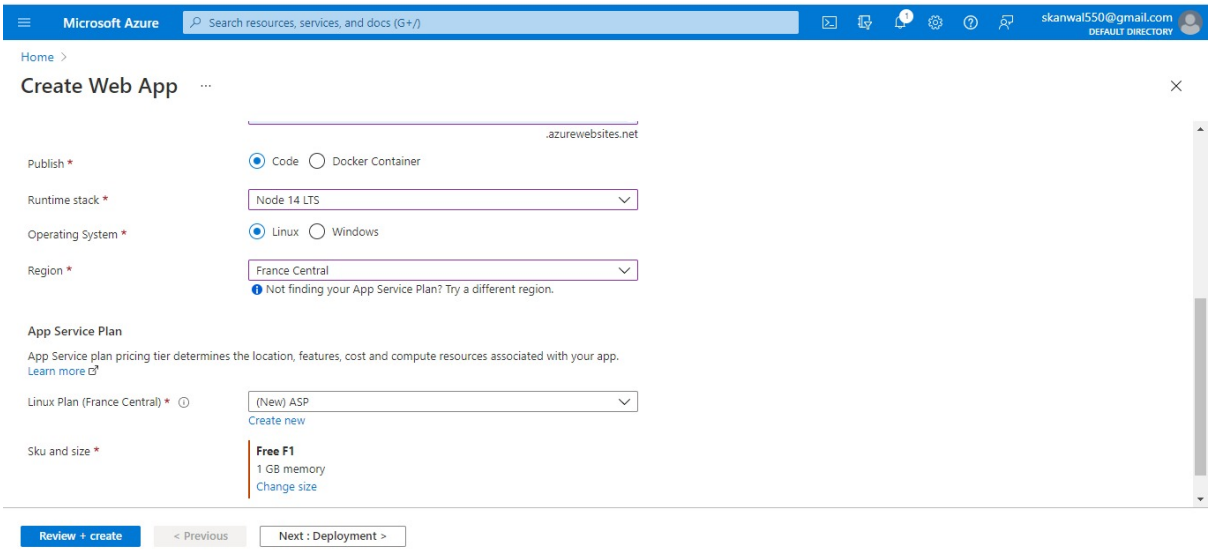
Creamos Web App. Vamos a necesitar la suscripción, nombre del resources group que acabamos de crear, nombre de la app que también estará en enlace (<AppName>.azurewebsites.net), vamos a subir código, escogemos Runtime Stack (Node 14 LTS), escogemos opción de Linux (Aunque da la opción de Windows, no funciona), región donde queremos que esté, y por último, App Service Plan, escogemos la opción que nos interese más. En Captura 9.5 y 9.6 podemos observar las opciones que escogí yo.



Captura 9.4: Opción para crear Web App.

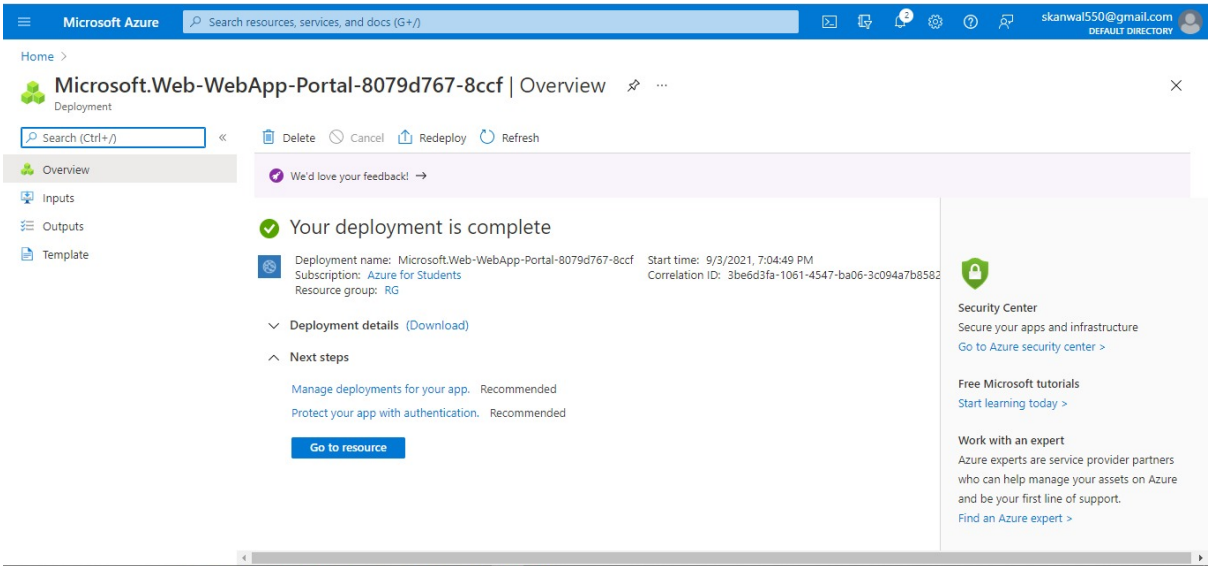


Captura 9.5: Crear Web App I.



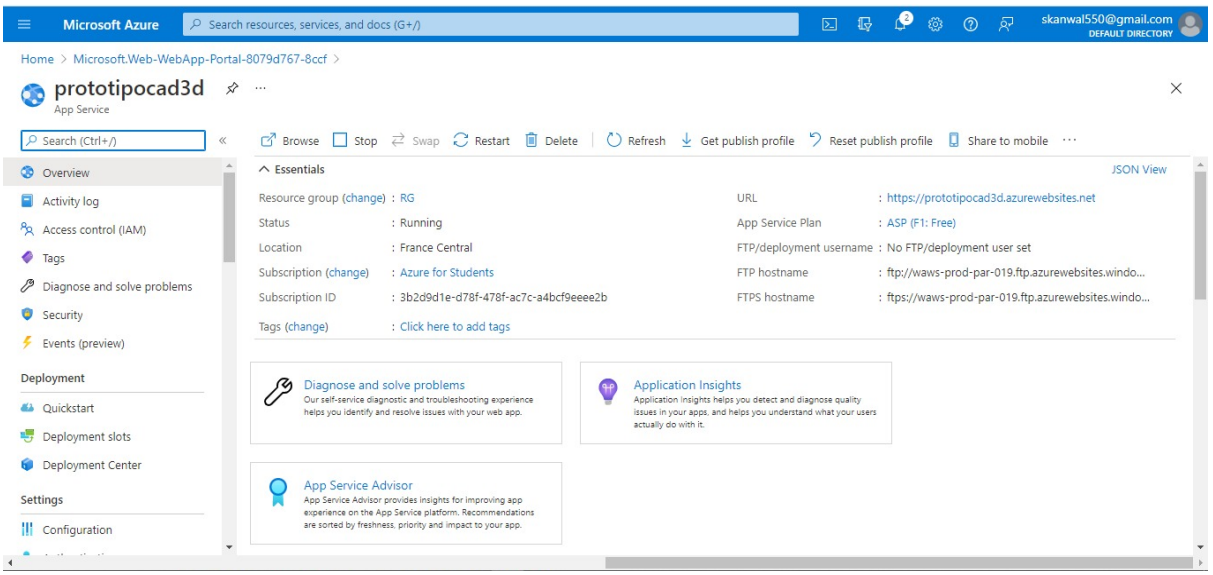
Captura 9.6: Crear Web App II.

Vamos a clicar en Review & create. Miramos el resumen, si os parece bien, le damos a Create. Esto nos llevará a la Captura 9.7 (tardará un poco en desplegar).



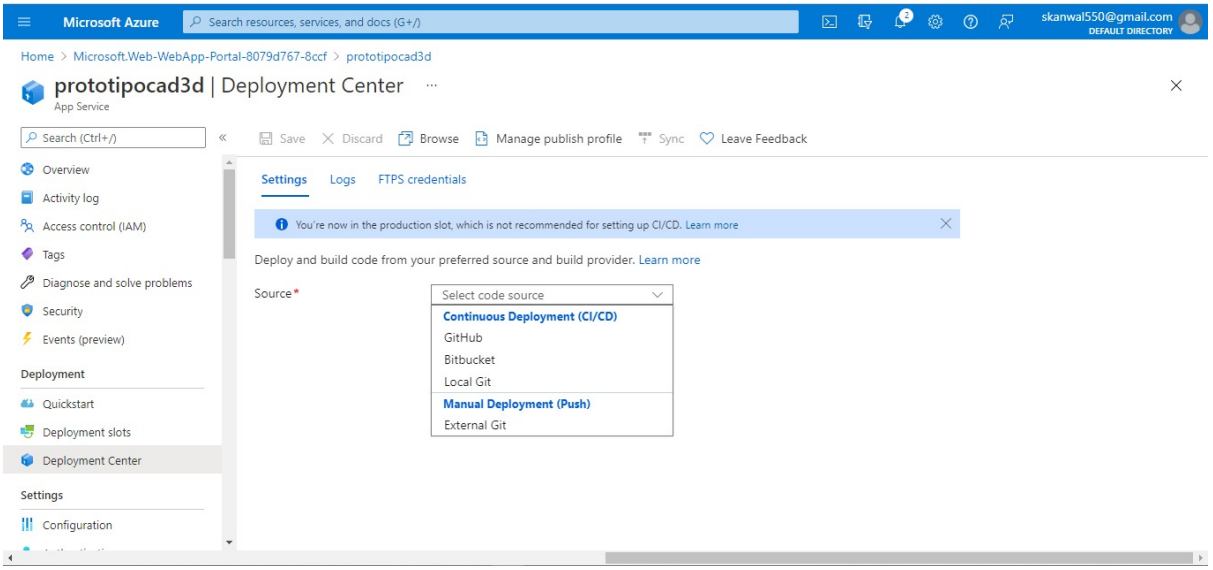
Captura 9.7: Se hace redirección a esta página después de crear Web APP.

Si clicas en Go to resource que está en Captura 9.7, llevará a la captura 9.8. Donde podemos observar la visión global del Web App.



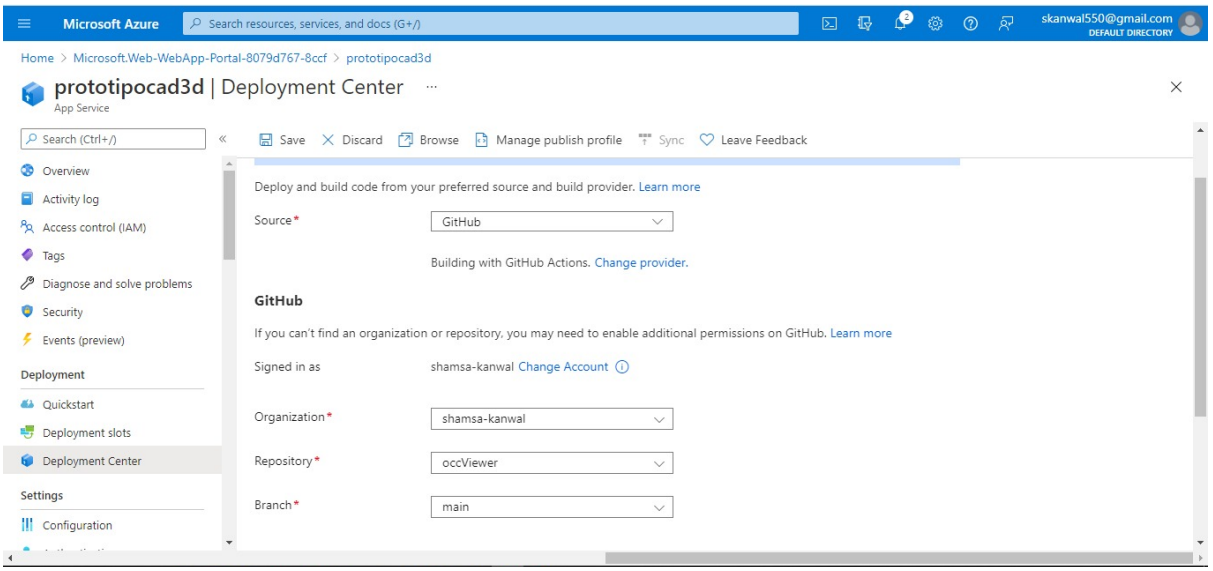
Captura 9.8: Visión global del Web App.

En la Captura 9.8, en la parte izquierda, podemos observar la opción Deployment Center. Clicamos en ella para dar fuentes del código.

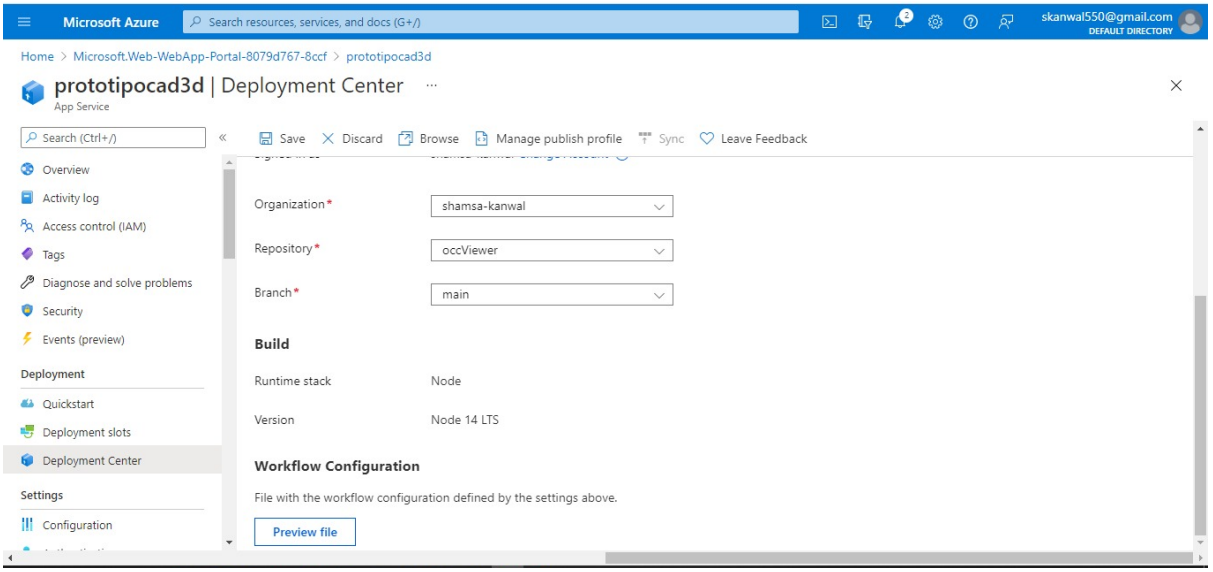


Captura 9.9: Opción para añadir fuentes del código.

En mi caso, tengo el código en Github. Así que seleccioné la opción de Github. Vamos a dar la información de la rama del código fuente. Podemos observar la información que di en la Captura 9.10 y 9.11. En Captura 9.11, está la opción de observar el preview file del workflow configuration.

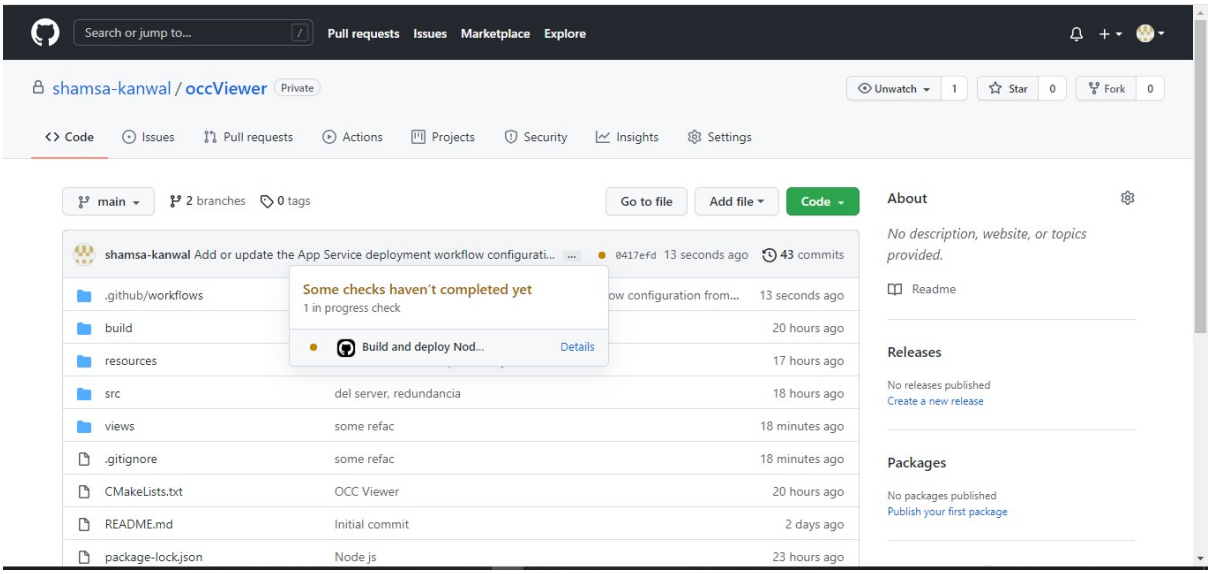


Captura 9.10: Información de la rama del código fuente.



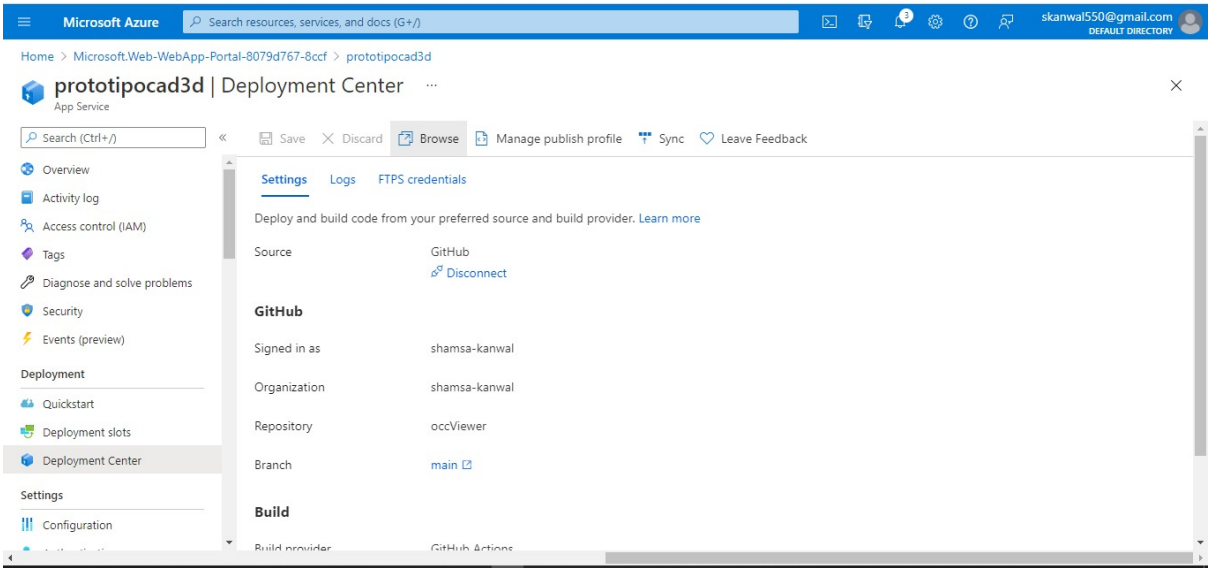
Captura 9.11: Información del build de la rama del código fuente.

El fichero de workflow configuration, básicamente lo que hará será que cada vez que haya algún cambio en la rama especificada, se hará el build del código y se desplegará en el azure. En la Captura 9.12 podemos observar que se está haciendo build y deploy. El punto marrón significa que se está haciendo build & deploy, el verde significa que se ha hecho bien (el web tiene los cambios) y por último, rojo que significa que ha habido error.



Captura 9.12: El punto marrón significa que se está haciendo el build y deploy.

En la Captura 9.13 tenemos la opción de Browse lo que nos permite abrir la Web App.



Captura 9.13: La opción de Browse, abre la web app.

Con todo esto ya tenemos creada la Web App.

9.3. Crear repositorio

En este apartado vamos a crear el repositorio. Yo voy a crear en GitHub que contendrá dos ramas. Una main/master y otra para desarrollo (develop).

Vamos al sitio web de GitHub, https://github.com/, y creamos un nuevo repositorio privado.

En ordenador vamos a clonar con el siguiente comando:

$> \ git\ clone\ < url >$

Entramos en la carpeta clonada que será con la que vamos a trabajar:

$> \ cd\ < folderCloned >$

Vamos a crear una nueva rama (branch) para desarrollo (develop)

$> \ git\ checkout\ -b\ develop$

Algunos de los comandos de Git que nos pueden interesar son:

$> \ git\ status$
$> \ git\ add\ .$
$> \ git\ commit\ -m\ "text"$
$> \ git\ push$
$> \ git\ pull$
$> \ git\ checkout\ <\ branchName\ >$
$> \ git\ branch$

Por último, cada vez que tengamos una versión estable, vamos a hacer commit en la rama main/master.

9.4. Descargar las librerías externas

En este apartado vamos a descargar y construir las librerías externas.

Vamos a crear una carpeta OCCT-Freetype y dentro de ella una más llamada OCCT-ext.

En OCCT-ext vamos a instalar los ejecutables y las librerías necesarias para construir OCCT (OpenCascade Technology).

Freetype

Vamos a necesitar instalar Freetype tanto para construir OCCT (<NombreLib>.dll & <NombreLib>.lib) como para el nuestro web (lib<NombreLib>.a). Necesitamos que sea la versión 2.5.3 y la vamos a descargar (.zip) desde https://sourceforge.net/projects/freetype/files/freetype2/2.5.3/.

Vamos a extraer en OCCT-ext y le cambiamos el nombre a freetype y entramos en ella.

Para freetype.lib: abrimos la solución freetype\builds\windows\vc2010\freetype.sln y la construimos.

Para freetype.dll:

-Dentro de la solución, abrimos menú Project → Properties → Configuration Properties → General y cambiamos la opción Configuration Type a Dynamic Library (.dll).

-Editamos el freetype\include\config\ftoption.h: en línea 285, descomentamos la definición de la macro FT_EXPORT y la cambiamos de la siguiente manera:

#define FT_EXPORT(x) __declspec(dllexport) x

-Construimos la solución

Para lib<NombreLib>.a que se construye con emsdk:

$> mkdir\ build$
$> cd\ build$
$> emcmake\ cmake\ ..$
$> emmake\ make$
$> emmake\ make\ install$

OCCT (OpenCascade Technology)

Antes de construir OCCT, necesitamos descargar y construir los ejecutables de terceros.

Descargar TCL 8.6 y TK 8.6 desde https://www.tcl.tk/software/tcltk/download.html. Vamos a extraer en OCCT-ext y siguiendo los pasos de https://dev.opencascade.org/doc/overview/html/build_upgrade_building_3rdparty.html vamos a construir esos ejecutables.

Ahora sí vamos a construir OCCT. Para empezar vamos a clonar desde Git.

$> git\ clone\ https://github.com/Open-Cascade-SAS/OCCT.git$
$> cd\ OCCT$
$> mkdir\ build$
$> cd\ build$
$> emcmake\ cmake\ ..\ -DBUILD\_LIBRARY\_TYPE="Static"$
$-DBUILD\_MODULE\_Draw='No'\ -DCAN\_USE\_GLES2='No'$
$> emmake\ make$

$> emmake\ make\ install$

Lo más probable es que no encuentre los variables como freetype_dir; tcltk_include, entre otros. Vamos a añadir esos PATHS manualmente dentro del CMakeLists.txt.

Note: Para tcl_lib_dir y tk_lib_dir hay que añadir path de las librerías (lib) del fuente y no la de compilada.

SQLAPI++

Vamos a descargar la librería SQLAPI++ para poder establecer la conexión con la base de datos.

Lo podemos descargar desde https://www.sqlapi.com/Download/.

Al extraer, nos interesa la carpeta “mingw-w64-x86_64-8.1.0-posix-seh-rt_v6-rev0”, concretamente, mingw-w64-x86_64-8.1.0-posix-seh-rt_v6-rev0/lib/libsqlapiddll.a. También nos interesa la cabecera include/SQLAPI.h

OCCI

OCCI es una interfaz definida por la empresa de bases de datos ORACLE que define una interfaz cómoda para que el programador de C ++ acceda a la base de datos Oracle con clases utilizando parámetros que recuerdan a las sentencias SQL.

Vamos a descargar Oracle Instant Client desde la web oficial, https://www.oracle.com/database/technologies/instant-client/downloads.html.
Extraemos oci.dll para luego generar liboci.a, para ello vamos a seguir las instrucciones que hay en siguiente enlace:

https://stackoverflow.com/questions/19849270/use-oracle-oci-dll-with-mingw64-64bit-compiler.

9.5. Refactorizar el código

En este apartado vamos a añadir las librerías externas y aprovechar el código de la muestra webgl (que se encuentra en OCCT).

En la carpeta donde clonamos el repositorio privado que habíamos creado:

- Crear una carpeta “ext” que contenga fuentes y librerías externas.
- Añadir CMakeLists.txt que se encuentra en webgl. Vamos a hacer los cambios necesarios para añadir las rutas de las librerías externas.
- Crear una carpeta “src” y añadir el código que hay en webgl.
- Crear una carpeta “bin” y otras que necesitemos para organizar bien el código.

9.6. Node.js

Como hemos visto en el apartado de Creación Web App, necesitamos un runtime stack, en mi caso he decidido por Node.js porque:

- Como hemos visto en el apartado de la arquitectura, al final el código del dominio es traspasado en javascript.
- Los programas están escritos en Javascript.

Vamos a implementar de modo que Node.js acabe usando el javascript que contiene código del dominio.

9.7. Crear una base de datos de Oracle en una máquina virtual de Azure

Para crear base de datos de Oracle vamos a seguir el tutorial que esta en https://docs.microsoft.com/en-us/azure/virtual-machines/workloads/oracle/oracle-database-quick-create#create-and-attach-a-new-disk-for-oracle-datafiles-and-fra además de hacer los cambios necesarios para luego poder conectar con base de datos.

Vamos a usar el mismo Resources Group que creamos para Web App.

> az vm create --resource-group RG --name vmoracle --image Oracle:oracle-database-19-3:oracle-database-19-0904:latest --size Standard_DS2_v2 --admin-username oracleuser --admin-password OracleDB2021 --public-ip-address-allocation static --public-ip-address-dns-name vmoracle

> az vm disk attach --name oradata01 --new --resource-group RG --size-gb 64 --sku StandardSSD_LRS --vm-name vmoracle

> az network nsg rule create --resource-group RG --nsg-name vmoracleNSG --name allow-oracle --protocol tcp --priority 1001 --destination-port-range 1521

> az network nsg rule create --resource-group RG --nsg-name vmoracleNSG --name allow-oracle-EM --protocol tcp --priority 1002 --destination-port-range 5502

> ssh oracleuser@<publicIpAddress>

> sudo su -

> parted /dev/sdc mklabel gpt

> parted -a optimal /dev/sdc mkpart primary 0GB 64GB

> mkfs -t ext4 /dev/sdc1

> mkdir /u02

> mount /dev/sdc1 /u02

> chmod 777 /u02

> echo "/dev/sdc1 /u02 ext4 defaults 0 0" >> /etc/fstab

> echo "<Public IP> <VMname>.eastus.cloudapp.azure.com <VMname>" >> /etc/hosts

> firewall-cmd --zone=public --add-port=1521/tcp --permanent

> firewall-cmd --zone=public --add-port=5502/tcp --permanent

> firewall-cmd --reload

> sudo su - oracle

> lsnrctl start

> mkdir /u02/oradata

> dbca -silent -createDatabase -templateName General_Purpose.dbc -gdbname vmoracledb -sid vmoracledb -responseFile NO_VALUE -characterSet AL32UTF8 -sysPassword OracleDB2021 -systemPassword OracleDB2021 -createAsContainerDatabase true -numberofPDBs 1 -pdbName vmoraclepdb -pdbAdminPassword OracleDB2021 -databaseType MULTIPURPOSE -automaticMemoryManagement false -storageType FS -datafileDestination "/u02/oradata/" -ignorePreReqs

> export ORACLE_SID=vmoracledb

> echo "export ORACLE_SID=vmoracledb" >> ~oracle/.bashrc

> sqlplus sys as sysdba

Con esto ya tenemos base de datos creada. Ya tenemos conectividad con publicIpAddress y puerto 1521. (tcping responde)

9.8. Conexión con base de datos mediante SQL Developer

Abrimos SQL Developer y clicamos en el símbolo +. Añadimos la siguiente información en la conexión:

- ☐ Name: oraclesys
- ☐ Username: sys
- ☐ Password: OracleDB2021
- ☐ Host: publicIpAddress
- ☐ Port: 1521
- ☐ Server Name; vmoraclepdb

Vamos a crear un usuario para después poder usarlo en el código para conectar con la base de datos con los privilegios necesarios.

9.8.1. Conectar con base de datos desde código

Vamos a usar SQLAPI++ para conectar con base de datos pasándole la adresa IP, puerto y nombre del servidor junto con usuario y contraseña que creamos en apartado anterior.

```
con.Connect(_TSA("51.103.65.107:1521/vmoraclepdb"),    _TSA("codeuser"),
_TSA("CodeUser"), SA_Oracle_Client);
```

A la función Connect, estamos pasando “@IP:Puerto/NombreServicio”, “NombreDeUsuario”, “Contraseña” y, por último, que es Oracle.

10. Conclusión y trabajo futuro

Podemos ver que se ha conseguido la mayor parte del objetivo principal. Comparando los proveedores de la nube, hemos visto que, en general, ofrecen algo parecido. Hay unos pequeños detalles que nos hicieron decidir por Azure que ya comenté en el apartado de resultado.

En cuanto, a la pregunta principal ¿Qué somos capaces de hacer en la nube? que planteamos en el inicio del proyecto. Podemos afirmar que podemos tener Viewer 3D del Opencascade en la Web App, añadir entidades en el viewer, conectar, leer y escribir en base de datos Oracle que tendríamos en máquina virtual de Azure. No tendría que haber problema al crear entidades, pintarlas en viewer y guardar en base de datos.

Por otro lado, no nos dió tiempo para la parte de la sincronización de base de datos entre la que está en Sener y la de nube.

Para los futuros trabajos, partiendo a partir de este proyecto podemos crear un prototipo CAD 3D con muchas más funcionalidades. Algo parecido al FORAN pero en la nube.

11. Bibliografia

[1] https://www.group.sener/es [En linea, 27/03/2021]

[2] https://www.marine.sener/es/foran [En linea, 27/03/2021]

[2.5] https://www.marine.sener/subsystems [En linea, 30/09/2021]

[3] https://www.bizkaia.eus/Home2/Archivos/DPTO8/Temas/Pdf/ca_GTcapitulo1.pdf?hash=fb75f1351c69c92a80d8fc0e7988e832 [En linea, 27/03/2021]

[4] https://www.scrum.org/resources/what-is-scrum [En linea, 28/03/2021]

[5] https://www.c-sharpcorner.com/article/what-is-user-story-in-agile-scrum/ [En linea, 28/03/2021]

[6] https://www.fib.upc.edu/es/empresa/practicas-en-empresa [En línea, 05/03/2021]

[7] https://sites.google.com/site/stigestionydesarrollo/recuperacion/recuperacion-gestion/tema-7/4 [En línea, 05/03/2021]

[8] https://www.glassdoor.es/Sueldos/jefe-de-proyecto-sueldo-SRCH_KO0,16.htm [En línea, 15/03/2021]

[9] Media Markt, Sant Cugat del Vallès

[10] https://fibra.jazztel.com/fibra-jazztel-com?utm_source=labellium&utm_medium=sem&utm_campaign=marca-generica&utm_term=pc&utm_content=marca-amplia&utoranw=labellium_sem_marca-generica_pc_marca-amplia&location=DI&gclid=Cj0KCQjw0caCBhCIARIsAGAfuMykVsTM8yWVPxo6o-QFKEuHErRlwdhtqXKOOOtg2KrPtddQE3I09LIaAtb8EALw_wcB&gclsrc=aw.ds [En línea, 13/03/2021]

[11] https://www.tmb.cat/ca/tarifes-metro-bus-barcelona/senzills-i-integrats/t-casual [En línea, 14/03/2021]

[12] https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas [En línea, 05/10/2021]

[13] https://ca.wikipedia.org/wiki/Patr%C3%B3_singleton [En línea, 06/10/2021]

[14] https://dev.opencascade.org/ [En línea, 08/10/2021]

[15]: https://git-scm.com/ [En línea, 15/09/2021]

[16]: https://www.python.org/ [En línea, 15/09/2021]

[17]: https://es.wikipedia.org/wiki/Node.js [En línea, 17/09/2021]

[18]: https://emscripten.org/ [En línea, 17/09/2021]

[19]: https://i.workana.com/glosario/que-es-c/ [En línea, 17/09/2021]

[20]: https://code.visualstudio.com/ [En línea, 19/09/2021]

[21]: https://docs.microsoft.com/es-es/azure/azure-portal/azure-portal-overview [En línea, 20/09/2021]

[22]: https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html [En línea, 15/10/2021]

[23] https://azuremarketplace.microsoft.com/es-es/marketplace/apps/microsoft-azure-compute.acc-virtual-machine-v2?tab=Overview [En línea, 08/07/2021]

[24] https://aws.amazon.com/es/products/compute/ [En línea, 09/07/2021]

[25] https://azure.microsoft.com/es-es/overview/why-azure/ [En línea, 09/07/2021]

[26] https://azure.microsoft.com/es-es/overview/azure-vs-aws/ [En línea, 12/07/2021]

[27] https://aws.amazon.com/es/products/?pg=WICC-N [En línea, 13/07/2021]

[28] https://azure.microsoft.com/es-es/overview/why-azure/ [En línea, 13/07/2021]

[29] https://aws.amazon.com/es/products/developer-tools/ [En línea, 14/07/2021]

[30] https://microsoft.github.io/techcasestudies/azure%20app%20service/2017/04/18/CAD.html [En línea, 16/10/2021]

[31] https://docs.microsoft.com/es-es/azure/virtual-machines/workloads/oracle/oracle-design [En línea, 17/10/2021]

[32]: https://emscripten.org/docs/getting_started/downloads.html [En línea, 12/09/2021]

[33]: https://www.mingw-w64.org/downloads/ [En línea, 08/10/2021]

[34]: https://cmake.org/ [En línea, 08/10/2021]