Desarrollo de una interfaz gráfica para una herramienta de cálculo de estructuras

Federico García, Rafael Olivera
Dr. Ing. Franco Robledo, Dr. Ing. Jorge Pérez, Ing. Pablo Castrillo

14 de febrero de 2016

Índice

1. Introducción

1.1. Definición del problema y motivación

Desde las primeras casas construidas por el hombre, hasta el edificio más moderno y extravagante que exista en la actualidad, puede decirse que se buscó en el fondo el mismo objetivo: lograr una estructura segura, resistente y funcional. Hoy por hoy, la evolución del conocimiento humano y de la tecnología circundante ha permitido desarrollar a niveles altísimos el comprendimiento del problema y sus posibles soluciones.

El cálculo de estructuras, en ese sentido, es una rama fundamental dentro de la ingeniería civil. Se trata de una serie de complejos cálculos realizados con la finalidad de lograr estructuras óptimas con las condiciones descriptas anteriormente. A grandes rasgos, se busca que la estructura pueda soportar tanto su propio peso, como cualquier fuerza externa que pueda ser aplicada a la misma. Debido a estos factores, la estructura puede sufrir ciertas deformaciones antes de alcanzar su punto de equilibrio.

La Ingeniería en Computación no ha dejado este problema de lado, ya que existen diversos sistemas informáticos encargados de facilitar el diseño y cálculo de estructuras. Estos sistemas permiten, a grandes rasgos, dibujar una estructura mediante la definición de diferentes elementos estructurales, materiales, secciones, apoyos, fuerzas externas, etc. Finalmente, realizan los cálculos correspondientes, mostrando la estructura en un estado de equilibrio con las solicitaciones y deformaciones ocurridas en el proceso.

Así como existen estos sistemas reconocidos mundialmente, la Facultad de Ingeniería (FIng) cuenta también son su propio sistema de cálculo de estructuras. Su nombre es IETFEM, y fué desarrollado por los Ing. Pablo Castrillo y Jorge Pérez pertenecientes al Instituto de Estructuras y Transporte (IET). Se trata de un motor de cálculo de código abierto desarrollado en GNU-Octave[1] que recibe una estructura descripta en formato texto y genera gráficas, imágenes y tablas de resultados. Es un sistema de uso académico que actualmente se utiliza en diversos cursos dictados por el instituto en cuestión.

En este proyecto, se desarrolló una interfaz gráfica acorde para ser utilizada en conjunto con el motor de cálculo antes mencionado, logrando así un sistema completo de diseño y cálculo de estructuras con un mayor grado de amigabilidad. Se busca, en particular, agregar funciones de dibujado y visualización de resultados que pueden observarse en otros sistemas de la misma índole, acercando al IETFEM a los sistemas comerciales y logrando una mayor usabilidad y eficiencia para los estudiantes que lo utilizarán.

1.2. Desarrollo previo

Como mencionamos anteriormente, la FIng cuenta con un motor de cálculo de estructuras desarrollado por los ingenieros Pablo Castrillo y Jorge Pérez denominado IETFEM[2]. El mismo resuelve problemas de cálculo de estructuras utilizando el Método de los Elementos Finitos (MEF).

El MEF es, desde mediados del siglo XX, una de las principales herramientas utilizadas por los ingenieros para el análisis de sistemas estructurales, mecánicos, eléctricos, etc. El avance de la computación y la disponibilidad creciente de computadores potentes a bajo costo ha provocado que los programas comerciales de MEF para el cálculo estructural sean utilizados masivamente. De hecho, en los últimos cuarenta años el MEF ha transformado los procedimientos de trabajo de todas las áreas de ingeniería y constituye hoy una de las herramientas indispensables con las que un ingeniero debe contar en el ejercicio de su profesión. Por otra parte, el uso del MEF por parte de profesionales no debidamente capacitados podría eventualmente producir errores en el diseño de estructuras, y por tanto, riesgos para los usuarios.

En este contexto, la enseñanza del MEF en las carreras de ingeniería se transforma en un desafío docente, donde además de formar a los estudiantes en el uso de diferentes programas de cálculo estructural es necesario transmitirles los conocimientos y herramientas que les permitan realizar un análisis crítico de los resultados. Es importante destacar además, que la mayoría de los programas comerciales (ej: SAP2000[3] y AxisVM[4]) de MEF son de código cerrado, por lo que presentan como desventaja a nivel educativo, que no permiten a los estudiantes ver su funcionamiento interno, limitando la comprensión de los errores durante el aprendizaje.

De esta manera surge entre docentes del Instituto de Estructuras y Transporte (IET) la motivación de brindar una solución al problema a través del desarrollo de un *software* educativo y de código abierto: IETFEM.

IETFEM comenzó a desarrollarse en 2012 . El primer módulo desarrollado permitió resolver problemas de estructuras de barras articuladas ó aporticadas en el plano con cargas aplicadas en los nodos. Esta primera versión fue utilizada por estudiantes del curso de Elasticidad 2013; luego se incluyó la posibilidad de generar un informe de salida en formato LaTeX. Posteriormente, la herramienta contó con el aporte del docente del IET, Agustín Spalvier, desarrollando la capacidad de ingresar cargas distribuidas uniformes en elementos de pórtico y el análisis modal de vibraciones de pórticos. Finalmente, a principios de 2014, Castrillo desarrolló un módulo para la resolución de problemas con variaciones de temperatura y fuerza de volumen en barras articuladas.

Se buscó una herramienta que sin ser compleja para su aplicación en cursos de grado, permita al estudiante visualizar el funcionamiento interno del método de cálculo. Por ello se optó por la sintaxis de programación de GNU-Octave (herramienta libre de alta compatibilidad con Matlab[5]), ya conocida por los

estudiantes. Se considera que contar con un *software* abierto donde los estudiantes pueden entender e incluso programar nuevos cálculos de acuerdo a sus necesidades, enriquece el trabajo desde el punto de vista didáctico.

La forma de ingreso de datos se eligió de acuerdo a otros programas de cálculo de estructuras como SAP2000 donde se deben definir: materiales, secciones, estados de carga, geometrías, conectividades, etc. En el IETFEM se optó por una entrada de archivo de texto plano donde el estudiante debe ingresar esta información. La salida también es en texto plano (.txt y .tex) y gráfica, al igual que en los programas comerciales.

Sin embargo, la generación del archivo de entrada y la comunicación con el IETFEM pueden llegar a ser tediosas y complicadas para el estudiante. Debe tenerse en cuenta que debe especificarse la estructura nodo por nodo, barra por barra, describiendo los materiales, secciones y fuerzas aplicadas, entre otras cosas, respetando además un formato fijo de documento que puede derivar en diversos errores de sintaxis.

Por lo tanto, se desarrolló en este proyecto una interfaz gráfica de código abierto donde el estudiante pueda dibujar la estructura de una manera sencilla e intuitiva, y que genere la entrada al IETFEM de manera automática. De esta manera, se pretende mejorar tanto la facilidad de uso como la eficiencia del mismo.

1.3. Objetivos y resultados esperados

Como mencionamos antes, a pesar de la increíble potencia en la resolución del problema del cálculo de estructuras, IETFEM presenta ciertos puntos a mejorar para ser comparado con otros sistemas del mismo rubro.

A lo largo de este proyecto se persiguirton 2 grandes objetivos que se consideran esenciales para el enriquecimiento del sistema: Mejorar la eficiencia y mejorar la usabilidad.

Para mejorar la usabilidad, se desarrolló una interfaz que permite al usuario dibujar la estructura de manera fluida y amigable. Se trata de un espacio 3D donde el usuario puede moverse libremente utilizando el mouse para desplazarse y rotar la cámara. Permite dibujar la estructura de una manera continua e intuitiva. Además, facilita la comunicación con el motor de cálculo previamente desarrollado y la visualización de los resultados obtenidos.

Para mejorar la eficiencia, se redujo el tiempo de ejecución del motor de cálculo, eliminando el proceso de graficación y generación de imágenes, ya que ahora los resultados pueden verse en la nueva interfaz. Como regla básica, buscamos que el usuario pierda el menor tiempo posible en problemas tecnológicos o informáticos y que dirija sus esfuerzos al comprendimiento del problema y su método de resolución.

A modo de resumen, se busca un sistema ágil, de código abierto, que mejore

ambos aspectos lo suficiente como para poder ser utilizado sin problemas en el curso de Elasticidad dictado por el IET. Con el fin de verificar el cumplimiento de los objetivos por parte del sistema, una vez finalizado, será evaluado resolviendo ejercicios del curso antes mencionado, realizando comparaciones y análisis del tiempo de ejecución.

1.4. Desarrollo del proyecto

El proyecto comenzó con una fase fuerte de investigación. Inicialmente se realizaron reuniones ocasionales con los tutores, donde se reunió información valiosa sobre la teoría de cálculo de estructuras y el método de elementos finitos. Además se definió qué tipo de herramienta se quería, qué funcionalidades eran deseadas y qué objetivos se buscaban. Durante esta etapa se utilizó el motor de cálculo directamente para comprender su funcionamiento y compararlo con otras herramientas comerciales.

Una vez comprendido el problema, se procedió a buscar herramientas con las cuáles desarrollar la interfaz. Se investigaron librerías y lenguajes de progamación 3D, optando al final por utilizar tecnologías web por su simplicidad de uso, agilidad y portabilidad.

Posteriormente se comenzó a diseñar e implementar la herramienta, separando en diferentes módulos que serán descriptos en detalle en el Capítulo 4. Se ejecutaron reuniones quincenales con los tutores para definir detalles, corregir errores, evaluar resultados y tomar decisiones en conjunto. Esta fase ocupó la mayor parte del tiempo del proyecto, debido a la dificultad técnica del mismo.

Finalmente, una vez alcanzado un producto inicial que cumplía las espectativas planteadas, se procedió a realizar pruebas sobre el mismo, detectando ciertos errores de performance que fueron solucionados hasta un nivel considerablemente bueno (se hablará de estas medidas en el Capítulo 5).

1.5. Organización del documento

El resto del documento se organiza de la siguiente manera:

En el siguiente Capítulo, se comienza analizando el estado del arte, tanto del problema de cálculo de estructuras como de herramientas de programación 3D, y su posible uso en sistemas de este tipo. Se realiza un estudio de diferentes herramientas investigadas, el estado de las mismas y su posibilidad de ser utilizadas en este proyecto. También se investigan otros sistemas de cálculo de estructuras y otros proyectos académicos similares en América Latina y el mundo.

Posteriormente, en el Capítulo 3, se habla de la organización del trabajo a lo largo del proyecto. Se plantea el alcance del mismo, definiendo las funcionalidades y características específicas que se buscan en el producto final. Se describe

la metodología de trabajo utilizada y se realizan estimaciones para cada tarea comprendida, comparando finalmente con el esfuerzo efectivo.

A continuación, en el Capítulo 4, se procede a plantear la solución propuesta, detallando cada aspecto de la misma. Se describe con exactitud su proceso de diseño e implementación, la arquitectura definida, el funcionamiento de cada componente, las herramientas utilizadas y su uso en general.

En el Capítulo 5, se especifican los resultados obtenidos, analizando diferentes casos de prueba y comparando con resultados obtenidos desde IETFEM antes de la realización de este proyecto. Se analizan además los problemas obtenidos durante esta fase y cómo fueron resueltos.

Finalmente, el Capítulo 6, enumera las conclusiones obtenidas durante el proyecto, analizando el cumplimiento de objetivos y proponiendo posible trabajo a futuro a desarrollar sobre IETFEM.

2. Estado del Arte

2.1. Cálculo de estructuras

2.1.1. Problema y cálculos implicados

IETFEM fue creado en 2013 por docentes del IET para resolver problemas de pórticos y reticulados planos utilizando el MEF.

El MEF[6] es un método numérico general para la aproximación de soluciones de ecuaciones diferenciales parciales. Está pensado para ser usado en computadoras y permite resolver ecuaciones diferenciales asociadas a un problema físico sobre geometrías complicadas. La idea general de este método es la división de un continuo en un conjunto de pequeños elementos interconectados por una serie de puntos llamados nodos. Las ecuaciones que rigen el comportamiento del continuo regirán también el del elemento. De esta forma se consigue pasar de un sistema continuo (infinitos grados de libertad), que es regido por una ecuación diferencial o un sistema de ecuaciones diferenciales, a un sistema con un número de grados de libertad finito cuyo comportamiento se modela por un sistema de ecuaciones, lineales o no.

Este método se usa en el diseño y mejora de productos y aplicaciones industriales, así como en la simulación de sistemas físicos y biológicos complejos. La variedad de problemas a los que puede aplicarse ha crecido enormemente, siendo el requisito básico que las ecuaciones constitutivas y ecuaciones de evolución temporal del problema a considerar sean conocidas de antemano.

2.1.2. IETFEM

IETFEM nació como una herramienta académica de código abierto desarrollada íntegramente en la plataforma libre GNU-Octave que soluciona los problemas anteriormente descriptos de forma eficaz y eficiente teniendo como principal objetivo enriquecer el proceso de aprendizaje de los estudiantes.

Si bien se logro un producto final de calidad y a nivel de las herramientas comerciales (incluso superándolas) en cuanto a los resultados de los cálculos, la herramienta quedó con un debe en cuanto a la interacción con el usuario. El mismo debe generar un archivo de texto con los datos de la estructura en un formato específico con gran cantidad de información en forma de matrices. Este proceso puede rápidamente volverse engorroso y difícil de manejar. Dicha característica es considerada como la más importante fuente de motivación del presente proyecto, así también como mejorar potencialmente la visualización de los resultados finales.

2.2. Herramientas comerciales

Existen en el mercado diversos productos de *software* enfocados al análisis de estructuras, con gran cantidad de funcionalidades y utilizados por ingenieros de todo el mundo en problemas reales. En el marco de este proyecto se exploraron con mayor rigurosidad dos herramientas: SAP2000 y AxisVM, las cuales en etapas más avanzadas del desarrollo fueron tomadas como estándar para la implementación de ciertas funcionalidades, basados principalmente en la experiencia de los tutores con las mismas.

2.2.1. SAP2000

Es un *software* comercial desarrollado por la empresa Computers & Structures,Inc. [7] fundada en 1975 en California, siendo uno de los pioneros en herramientas de análisis de estructuras.

Actualmente en su versión 18, SAP2000 es una aplicación para computadoras que se ejecuta en ambientes Windows. Cuenta con un entorno gráfico 3D para el modelado y una interfaz de usuario muy completa que puede resultar demasiado compleja para modelar estructuras simples.

Entre las características más importantes se encuentran:

- Un motor de análisis que puede resolver varios tipos de problemas.
- Diversas características para el modelado como *templates*, sistema de grillas, distintas vistas y herramientas de meshing.
- Diversos componentes estructurales como articulaciones, barras, cables sólidos, resortes, etc.

- Posibilidad de aplicar distintos tipos de cargas.
- Varias posibilidades para ver la salida de los cálculos con diagramas, tablas y reportes.
- Importación y exportación de modelos en distintos formatos estándar.

Por todo esto, SAP2000, es uno de los productos comerciales líderes en el mercado siendo utilizado en más de 160 países en todo el mundo[8].

En cuanto al licenciamiento, es un software privativo el cual cuenta con la opción de solicitar un trial de una versión limitada de la herramienta. Para comprarla es necesario contactarse con un vendedor de Buenos Aires.

2.2.2. AxisVM

Es un *software* comercial desarrollado por la empresa InterCAD Kft. en 1991 y con sede en Hungría. Fue una de las primeras herramientas 3D basada en el método de los elementos finitos.

Actualmente en su versión 13, AxisVM requiere computadoras con el sistema operativo Windows. El conjunto de características es muy similar al descripto en la anterior herramienta.

Axis cuenta con una versión *light* gratuita con limitaciones en la cantidad de elementos que se pueden construir, ademas de la posibilidad de descargar una versión *trial* por tiempo limitado con el fin de evaluar más intensamente la herramienta.

2.2.3. Herramientas Web

El sector del *software* de análisis estructural en la web (o nube) es un paradigma poco explorado por los desarrolladores, existiendo un conjunto muy limitado de ofertas en este sentido.

De acuerdo a la investigación realizada es importante destacar las siguientes ofertas:

- Idea StatiCa [9] es un emprendimiento Checo que cuenta con calculadoras para 6 problemas principalmente en espacios 2D. Ha sido desarrollado en Silverlight y utilizando la nube de Microsoft Azure como plataforma de despliegue.

  Cuenta con ciertas características gratuitas y funciona con un sistema de créditos que se deben comprar para realizar ciertas funciones.
- CloudCalc [10] es un *software* en crecimiento enfocado al análisis de estructuras de acero en la nube proveniente de Houston EEUU. Ha sido desarrollado utilizando WebGL para las características gráficas 3D.

Actualmente es una aplicación gratuita, es necesario simplemente la creación de una cuenta para poder probarla.

- SkyCiv [11] es un emprendimiento reciente de origen Australiano y es la suite más desarrollada y con mayor calidad aparente de las vistas en este sector. Cuenta con calculadoras para distintos problemas en 2D y una versión pro que permite estructuras en 3D. Utiliza también WebGL para los gráficos.

  Cuenta con un *trial* por 30 días para probar todas sus funcionalidades y luego es un servicio que se paga de forma mensual.

Si bien existen algunas pocas ofertas, no logran niveles de calidad similares por ejemplo a herramientas de escritorio como SAP2000 o AxisVM, encontrándose así una ventana de oportunidad para el desarrollo de este tipo de herramientas con interfaces Web.

2.3. Desarrollo 3D

Dado el fuerte componente gráfico del proyecto fue necesario repasar un gran abanico de posibilidades a nivel tecnológico que permitan cumplir con los requerimientos 3D de la aplicación requerida. A continuación se muestran las principales opciones investigadas, que van desde especificaciones estándares de muy bajo nivel de abstracción, pasando por *wrappers* de las mismas, hasta completos y potentes motores gráficos.

Se priorizaron fuertemente herramientas gratuitas de código abierto dado que fue un requerimiento por parte de los tutores. Además se hizo especial foco en tecnologías conocidas por los desarrolladores tales como Java o tecnologías web.

2.3.1. OpenGL

Es una especificación estándar que define una *API* multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráfico 2D y 3D.

El funcionamiento básico consiste en aceptar primitivas como puntos, lineas y polígonos y convertirlas en píxeles. Es una *API* basada en procedimientos de bajo nivel que requiere que el programador dicte los pasos exactos para renderizar la escena. Esto la diferencia de otras *APIs* más descriptivas, donde el programador sólo debe describir la escena.

OpenGL [12] tiene dos propósitos esenciales:

- Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una *API* única y uniforme.

- Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL

En la actualidad en su versión 4.5 se utiliza ampliamente en CAD, realidad virtual, representación científica, visualización de información, simulación y desarrollo de videojuegos donde compite con Direct3d (en plataformas Microsoft).

En virtud de lo detallado en cuanto al bajo nivel de abstracción y por consecuente prolongada curva de aprendizaje y baja productividad esta opción fue descartada rápidamente al menos en su uso directo.

2.3.2. LWJGL y JOGL

JWJGL (Lightweight Java Game Library 3) [13] y JOGL (Java OpenGL) [14] son *wrappers* de OpenGL que proveen acceso de bajo nivel a sus funcionalidades que a menudo no se implementan de manera correcta. No son librerías con gran cantidad de funcionalidades ni proveen utilidades de alto nivel.

En la actualidad existen muchas herramientas y motores gráficos para desarrollar aplicaciones 3D con mayor cantidad de funcionalidades, menor curva de aprendizaje y mayor productividad que utilizan estas librerías como base.

2.3.3. JMonkeyEngine 3

Es un motor de código abierto con fuerte inclinación para el desarrollo de videojuegos [15], hecho especialmente para desarrolladores Java para la creación de aplicaciones 3D utilizando las más modernas tecnologías de una manera rápida y con una baja curva de aprendizaje.

Esta desarrollado en base a JWJGL y es la suite más popular en el mundo java para desarrollo de videojuegos en alto nivel, con una gran comunidad de desarrolladores y extensivamente documentado. Si bien el enfoque principal son los videojuegos es importante destacar que tiene todas las capacidades para poder construir otro tipo de aplicaciones.

2.4. Desarrollo 3D en la Web

La utilización de tecnologías web para el desarrollo de la aplicación probaba a priori ser una opción con mucho potencial aportando gran flexibilidad, una opción multiplataforma-multidispositivo además de ser innovadora para herramientas de este tipo.

Es por estas razones y la experiencia del equipo de desarrollo en estas tecnologías (HTML5, CSS, Bootstrap, JavaScript, AngularJS, etc.) que se investigó la factibilidad de una solución gráfica 3D en este contexto.

2.4.1. HTML5 - Canvas

El contexto 2D para el elemento de HTML «Canvas»[16] permite la creación de gráficos en páginas web. Es una tecnología que se usa principalmente para dibujar gráficos 2D en la web, aunque es posible realizar trabajos en 3D.

La relativa dificultad para realizar trabajos 3D y la gran diferencia de performance contra opciones como WebGL (Canvas corre en CPU) descartaron esta opción (en su uso directo) rápidamente.

2.4.2. WebGL

WebGL [17] es una especificación estándar que está siendo desarrollada actualmente para mostrar gráficos en 3D en navegadores web. Permite mostrar gráficos en 3D acelerados por *hardware* (GPU) en páginas web, sin la necesidad de emphplug-ins en cualquier plataforma que soporte OpenGL 2.0 u OpenGL ES 2.0. Técnicamente es una *API* para JavaScript que permite usar la implementación nativa de OpenGL ES 2.0.

Existe una gran gama de aplicaciones desarrolladas sobre esta tecnología, desde videojuegos 3D hasta aplicaciones científicas como visualizadores de estructuras moleculares, simulaciones del sistema solar o una aplicación de la NASA llamada «Experience Curiosity» [18] por el aniversario del aterrizaje del robot «Curiosity Rover» en Marte.

Es importante destacar que en la actualidad esta soportado por todos los principales navegadores web tanto en versiones de escritorio como de dispositivos móviles.

Con esta gran cantidad de demostraciones de calidad en WebGL y su amplio soporte se perfiló como una opción innovadora y altamente factible para la realización del proyecto.

2.4.3. Librerías para desarrollo 3D

Como WebGL es una tecnología diseñada para trabajar directamente con el GPU (unidad de procesamiento gráfico) es difícil de codificar en comparación con otros estándares web más accesibles, es por eso que muchas bibliotecas de JavaScript han surgido para resolver este problema.

Entre las mismas se privilegiaron aquellas con mayor cantidad de características implementadas, documentación y comunidad. La investigación entonces se simplifico a dos: ThreeJs [19] y BabylonJs [20] (Microsoft Open Source).

Ambas son librerías en JavaScript de alto nivel de abstracción sobre WebGL, tienen esencialmente el mismo conjunto de características tales como:

- Renderizacion con WebGL.

- Distintos efectos.
- Escenas, para añadir y eliminar objetos en tiempo de ejecución.
- Cámaras, perspectiva u ortográfica.
- Animaciones.
- Luces.
- Materiales.
- *Shaders*.
- Objetos y Geometrías
- Importación y exportación para texturas y otros assets.
- Gran comunidad de desarrolladores (bastante mayor en ThreeJS).

Ademas en sus paginas oficiales cuentan con cientos de códigos y aplicaciones de ejemplo que dejan ver el potencial de las librerías. Entre estos ejemplos se pueden observar editores estilo CAD que implementan varias funcionalidades similares a los requerimientos del proyecto lo cual asegura la factibilidad de la herramienta en este contexto.

2.5. Información complementaria

2.5.1. Investigación sobre proyectos similares en América Latina

No se encontraron, en una exploración por la web o por experiencia de los docentes de Ingeniería Civil, herramientas de *software* con componente gráfico 3D de análisis de estructuras desarrollados en Latinoamérica, aunque se encuentran ciertos esfuerzos de algunas empresas de las herramientas más importantes por llegar a este mercado mediante documentación, paginas web y/o re vendedores íntegramente de habla hispana.

2.5.2. Conclusión

Se investigaron las principales herramientas dentro del espectro de posibilidades para el desarrollo del *software* en cuestión.

Dentro de las herramientas de escritorio se destaca JMonkeyEngine, ofreciendo un ambiente de alto nivel para desarrollo 3D respaldado por una gran comunidad e interesantes funcionalidades. Por otra parte, dentro del mundo del desarrollo web, sobresale la calidad de las librerías de desarrollo implementadas sobre WebGL, como BabylonJs o ThreeJs. Estas librerías ofrecen el manejo de ciertos conceptos implicados en el desarrollo de la interfaz (movimiento de la cámara, agregado de objetos, etc.) de una manera intuitiva y manejable.

Aunque ambas opciones se perfilaron para ser utilizadas, fué clave en la decisión tomada el hecho de encontrar escasos sistemas de cálculos de estructuras desarrollados en la nube. El equipo se planteó como interrogante si es posible realizar un sistema con tales características, y si es así, «£por qué?» no ha sido explorado como posibilidad por las grandes empresas.

Este suceso, sumado a la experiencia del equipo de desarrollo en este tipo de tecnologías, centraron la atención en herramientas web para el desarrollo del proyecto.

3. Organización del trabajo

3.1. Alcance

Como se mencionó anteriormente, los objetivos planteados en este proyecto consisten en mejorar tanto la eficiencia como la usabilidad del IETFEM. En ese sentido, existen dentro de la rama del cálculo de estructuras una infinidad de funcionalidades y mejoras posibles que pueden resultar útiles en el sistema. Por lo tanto, se definió un conjunto acotado de funcionalidades y características deseables en el producto final, apuntando a alcanzar satisfactoriamente los objetivos planteados y lograr una herramienta de alto nivel.

Se consideró como prioridad apuntar a una herramienta académica, es decir, una herramienta libre, intuitiva para los estudiantes y aplicable en cursos dictados por el IET. En particular, se tomó como referencia el curso de Elasticidad, curso donde ya fué utilizado satisfactoriamente el IETFEM y donde será utilizado luego de la realización de este proyecto.

La principal y más grande funcionalidad que se desarrolló fué le presencia de un espacio 3D. El mismo es el elemento central de la aplicación, mediante la cual el usuario efectúa la mayor parte de las interacciones posibles. Se pretendió integrar dentro de este espacio 3D las siguientes funcionalidades:

- Rotación de la cámara de visualización.
- Movimiento de la misma por todo el espacio 3D.
- *Zoom In* y *Zoom Out*.
- Dibujado de nodos y barras.
- Dibujado de grillas auxiliares.
- Selección de nodos y barras: Para setear propiedades en las mismas.
- Eliminación de nodos y barras.
- Visualización y ocultación de propiedades: Fuerzas, puntos de apoyo y resortes.

- Visualización de estructura resultante: Observar la deformada y comparar con estructura original.
- Escalamiento de la estructura deformada: «Exagerar» la deformación, para apreciar los pequeños desplazamientos.
- Visualización de las propiedades de la estructura deformada utilizando escalas de colores: Deformación, Fuerzas, Tensiones, etc.

Más allá de que se pretende que el usuario tenga una experiencia interactiva mediante el dibujado de la estructura, es necesario definir ciertas funcionalidades fuera del espacio 3D. Ya sea tanto por comodidad como por intuición, estas opciones se encuentran en diferentes menús que rodean el espacio, similar a los demás programas comerciales dentro del rubro que se investigaron en el Capítulo anterior.

- Abrir y Guardar Estructuras.
- Definición de Materiales.
- Definición de Secciones.
- Asignar propiedades a barras: Material y sección.
- Asignar propiedades a nodos: Fuerzas, Apoyos y Resortes.
- Selección de nodos y barras: Para setear propiedades en las mismas.
- Eliminación de nodos y barras.
- Prendido y apagado de elementos auxiliares
- Seteo de Factor de escalamiento para la estructura deformada

Si bien estos elementos nos permiten estimar una interfaz gráfica completa e intuitiva, resta definir aún la funcionalidad más importante del proyecto: la comunicación con el motor de cálculo. La salida de la interfaz debe ser un archivo reconocible por el IETFEM, del cuál pueda obtener todos los datos de la estructura. Así mismo, el motor debe ofrecer como salida otro archivo, el cuál será recibido por la interfaz con el fin de mostrar los resultados obtenidos. Dicha comunicación puede observarse en la Figura 1, aunque se hará hincapié en cómo se resolvió esta comunicación en el siguiente Capítulo.

3.2. Metodología de trabajo

En las primeras etapas del proyecto se focalizó el trabajo en comprender el problema que se quiere resolver. Se tuvieron reuniones quincenales con los tutores dónde se habló del problema del cálculo de estructuras y cómo lo resuelve IETFEM. Dichas reuniones se apoyaron además en una permanente comunicación por e-Mail y una vasta investigación del problema por parte del equipo de

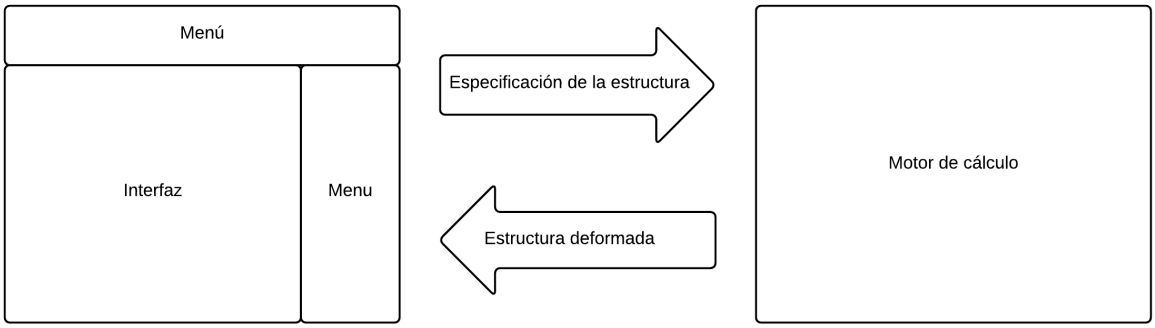

Figura 1: Ciclo de vida de IETFEM

desarrollo. Para esto no sólo se investigó sobre el problema, sino que además se utilizaron productos similares e incluso el propio IETFEM con ejemplos simples.

Una vez comprendido el problema, se buscó una solución al mismo. Dentro de esta etapa se pueden incluir la búsqueda de herramientas, el análisis y el diseño de la aplicación. Se mantuvieron las reuniones con los tutores, evaluando herramientas y enseñando prototipos realizados a modo de prueba. Se investigaron lenguajes y librerías de programación 3D, tanto de escritorio como web, decidiendo en última instancia utilizar WebGL (se hablará más en detalle en el siguiente Capítulo).

Conforme transcurría el tiempo, las reuniones se fueron enfoncando cada vez más en el producto final, comenzando a definir las funcionalidades y características del mismo. Mientras se mantenía contacto con los tutores, se realizó la definición de casos de uso, al tiempo que se definió la arquitectura del sistema en base a los requerimientos obtenidos y las prestaciones de las herramientas definidas.

Finalmente, para las etapas de implementación y testeo, se creó un repositorio en Github con el esqueleto de la aplicación y todo código reusable proveniente de la etapa de prototipación. Como metodología de trabajo se utilizó la metodología ágil Kanban[21]. Kanban es un método para gestionar el trabajo intelectual, con énfasis en la entrega justo a tiempo, mientras no se sobrecargan a los miembros del equipo. En este enfoque, el proceso, desde la definición de una tarea hasta su entrega al cliente, se muestra para que los participantes lo vean y los miembros del equipo tomen el trabajo de una cola. Se basa en la idea de visualizar lo que se está haciendo ahora, lo que se está terminando y lo que hay que hacer a continuación.

Existen diversas herramientas *online* de planificación y gestión de proyectos, tales como Jira[22] o TFS[23]. Sin embargo, debido a la poca cantidad de personas involucradas en el proyecto (2 desarrolladores y 2 tutores) y a que las tareas a realizar estaban bien definidas, se optó por utilizar una herramienta simple y natural: una planilla *online*. La misma se encontró en todo momento de libre

Descripción	Estado	Prioridad
Agregar la posibilidad de definir un material por defecto al dibujar barras	Resuelta	1
Agregar sprites y flechas a los nodos cuando se definen fuerzas o condiciones de desplazamiento	Resuelta	1
Agregar funcionalidad para "prender" y "apagar" grillas y fuerzas	Resuelta	2
Agregar funcionalidad Abrir Modelo / Guardar Modelo	En proceso	2
Agregar funcionalidad Nuevo Modelo, que limpia la escena y permite definir las nuevas unidades de medida	Para hacer	3

Figura 2: Planilla utilizando metodología Kanban

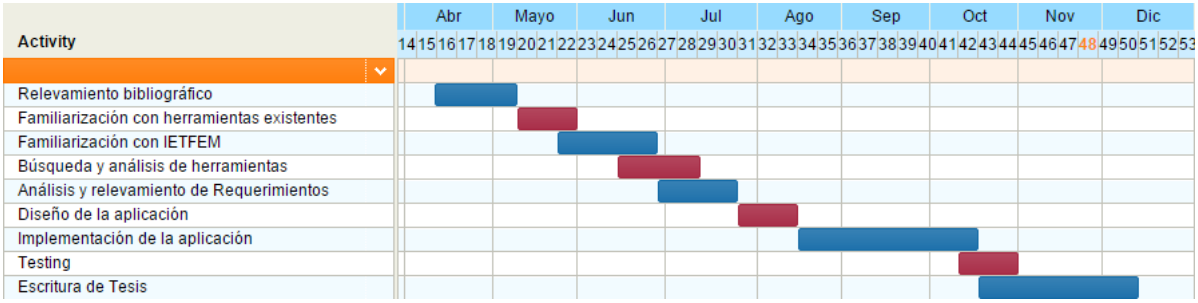

Figura 3: Diagrama de Gantt con la planificación del proyecto

acceso y modificación para los 4 participantes, y cada tarea tenía asignada una descripción, un estado, y una prioridad.

En un principio, se agregaron todas las tareas a realizar, y ambos desarrolladores tomaban cada una de ellas marcando su estado como «En proceso». Una vez finalizada, se marcaba la tarea como terminada y se subían los cambios al repositorio, marcando cada subida con la funcionalidad correspondiente.

A su vez, los tutores, los cuáles también tenían acceso a la última versión del IETFEM, iban relevando en la misma planilla problemas o cosas a mejorar que se encontraban en el produto, los cuáles pasaban a ser parte de la «pizarra de Kanban» y seguían el mismo flujo que las demás tareas.

Se puede ver la planilla mencionada en un instante de tiempo en la Figura 2

3.3. Estimación y esfuerzo efectivo

La planificación del proyecto se realizó tomando en cuenta el desconocimiento inicial del problema de cálculo de estructuras y la dificultad de la programación gráfica en 3D. En la Figura 3 se pueden ver las estimaciones realizadas calculando 15 horas de trabajo semanal por desarrollador. Se puede apreciar que el período de trabajo se calculó entre los meses deabril y diciembre, logrando un total de 34 semanas de trabajo, que se traducen en un total de 1020 horas de trabajo. Se considera el tiempo estimado dentro del rango de lo aceptable para el desarrollo de un sistema de mediano-gran porte como el se desea.

Podemos ver también que ciertas etapas se planificaron en simultáneo por ciertos períodos de tiempo, especialmente en las etapas tempranas del proyecto donde se comenzó utilizando y comprendiendo tanto el IETFEM como otras herramientas, mientras se iba definiendo al mismo tiempo cómo realizar la interfaz. Se planificó de esta manera debido a que se consideró que sería bueno evaluar varias herramientas en simultáneo, a modo de comparar y definir qué funcionalidades y características nos gustaría que estén presentes en el sistema. También evaluar cómo llevarlas a cabo utilizando las herramientas que existen en el mercado y el contexto académico en el cuál se quiere insertar la aplicación.

Se observa en la Figura 3 la concurrencia de tareas en los instantes finales del desarrollo, donde se planificó al mismo tiempo el *testing* y la escritura de la Tesis. Debido a la metodología ágil elegida y al tiempo estipulado, resulta conveniente que el testeo de la aplicación comience cuanto antes, ya que corregir un error pasará a ser parte de la cola de tareas, y dependiendo de la prioridad de la misma podría ser resuelta antes que otras tareas menos prioritarias definidas anteriormente. La escritura de la Tesis se planificó en simultáneo simplemente para intentar reducir el tiempo total del proyecto.

El cronograma estimado se realizó de manera exitosa, siguiendo cada etapa en el orden estipulado sin demoras excesivas. Como agregado, durante la implementación se descubrieron nuevas funcionalidades que serían útiles en el sistema, las cuales fueron evaluadas con los tutores y algunas de ellas se llevaron a cabo sin problemas, debido a que la metodología de trabajo lo permitía.

También es necesario destacar el tiempo invertido en la Ingeniería de Muestra a fines del mes de Octubre, el cuál contempló el diseño de carteles, presentación del proyecto y la propia presencia en el evento. Esto redujo unos días el tiempo estipulado para la escritura de la Tesis, el cuál se intenta recuperar durante el mes de noviembre aumentando la cantidad de horas a un promedio de 20 semanales por desrrollador dedicadas a dicha tarea.

4. Presentación de la solución

En esta sección se describe la solución propuesta para el problema planteado, describiendo cada aspecto de la misma y cómo fue realizada cada una de sus funcionalidades. Se detallan además las decisiones que fueron tomadas durante el proceso de análisis y diseño de la aplicación.

4.1. Análisis y relevamiento de requerimientos

Desde un principio se supo que IETFEM era una herramienta robusta, ofreciendo una solución para diferentes problemas posibles. En este sentido, el relevamiento de requerimientos se convirtió en una tarea delicada en dónde debía

definirse un número acotado de funcionalidades, para un número acotado de la totalidad de problemas que IETFEM podía resolver.

Luego de concretar varias reuniones con los tutores, se decidió que la interfaz permita resolver problemas de estructuras reticuladas, es decir, estructuras formadas por una serie de barras entrecruzadas y conectadas entre sí por medio de nodos articulados. Esto implica que para dibujar una estructura desde la interfaz, el usuario sólo tenga que colocar nodos y barras.

El estudiante coloca los nodos en el espacio 3D, y luego define barras entre 2 nodos ya dibujados, asignando para cada barra un material que la conforma y el área de su corte transversal, al que llamaremos sección, ambos previamente definidos. También pueden definirse ciertas propiedades para cada nodo, en particular, pueden definirse fuerzas aplicadas al mismo, condiciones de desplazamiento y resortes.

Una vez finalizado el proceso de dibujado, se extrae la estructura en un formato reconocible por el motor, se ejecuta el mismo, y se analizan los resultados obtenidos.

Destacamos además como funcionalidades secundarias la posibilidad de definir grillas auxiliares con motivo de facilitar el ingreso de datos y la posibilidad de ocultar elementos adicionales, como por ejemplo, los vectores indicadores de fuerzas aplicadas.

Basándonos en esta realidad, se definieron los siguientes casos de uso:

- **Alta, Baja y Modificación de Materiales:** Los materiales se definen en base a 5 propiedades: Nombre, Modulo de Young, Gamma(γ), Alpha(α) y Nu(ν).
- **Alta, Baja y Modificación de Secciones:** La sección es el corte transversal de una barra, y para este tipo de problemas solo interesa conocer su área.
- **Alta, Baja y Modificación de Nodos:** Cada nodo tiene asignado un conjunto de coordenadas espaciales (x, y, z). Además es posible asignar al mismo una fuerza aplicada, así tambien como condiciones de desplazamiento y resortes en cada coordenada.
- **Alta, Baja y Modificación de Barras:** Cada barra tiene asignado un nodo inicial, un nodo final, un material y una sección.
- **Crear grilla:** Son «cuadrículas» auxiliares que facilitan el proceso de dibujado. Para cada coordenada se define la cantidad de líneas auxiliares y la separación entre ellas.
- **Modificar Visualización de Propiedad:** Los nodos con propiedades definidas, como por ejemplo fuerzas aplicadas o resortes, son marcados en la pantalla con vectores o geometrías básicas para ser diferenciados

del resto. Esta funcionalidad permite ocultar, mostrar y escalar dichos elementos a gusto del usuario.

- **Nueva Estructura:** Permite limpiar la pantalla para comenzar una nueva estructura.
- **Abrir y Guardar Estructura:** Se busca la posibilidad de obtener un archivo con la estructura dibujada, de manera de poder seguir con el trabajo realizado en otro momento. También es deseable la carga de dicho archivo en la interfaz, obteniéndo la misma estructura en la que se estaba trabajando al momento de guardar.
- **Generar Especifiación:** A partir del dibujo realizado, se genera un archivo reconocible por el motor de cálculo con la especificación de la estructura
- **Procesar Resultado:** Se trata de procesar el archivo resultante del motor y actualizar la pantalla con la estructura deformada.
- **Escalar Deformada:** Debido a que en algunos casos los desplazamientos pueden ser tan pequeñas que pueden parecer imperceptibles a la vista, se incluye este caso de uso con el fin de «exagerar» los desplazamientos y poder apreciar mejor los resultados obtenidos.
- **Colorear Estructura:** Al igual que la funcionalidad anterior, este caso de uso aplica a los resultados obtenidos del motor. Se trata de colorear la estructura en base a los datos obtenidos(por ejemplo, pintar de un color las barras que se comprimen y de otro las que se estiran). También se busca transparentar la estructura original o la deformada, para poder apreciar mejor los cambios entre una y otra.

En el anexo A se incluye la especificación de cada caso de uso descripto.

Puede apreciarse el Modelo de Dominio definido en la Figura 4. Como observaciones, se destaca la presencia de la entidad «Deformada», la cuál puede existir o no de acuerdo a si ya se procesaron los resultados obtenidos del motor o si se encuentra en el proceso de dibujado. De esta acotación se desprende el «por qué» de la relación 0..1 - 1 entre las entidades «Deformada» y «Estructura»: mientras se dibuja la estructura todavía no se tiene una deformada definida.

El resto del modelo se encuentra considerablemente intuitivo y adecuado a la realidad planteada.

4.2. Diseño de la solución

Finalizado el relevamiento de requerimientos y correspondiente análisis, se prosiguió con la etapa de diseño, donde se tomaron decisiones importantes tanto a nivel de diseño tecnológico como en la estructura propia de la aplicación.

Desde las primeras reuniones que se tuvieron con los tutores, el objetivo principal fué lograr una aplicación académica. De esta manera, se tuvo como prioridad

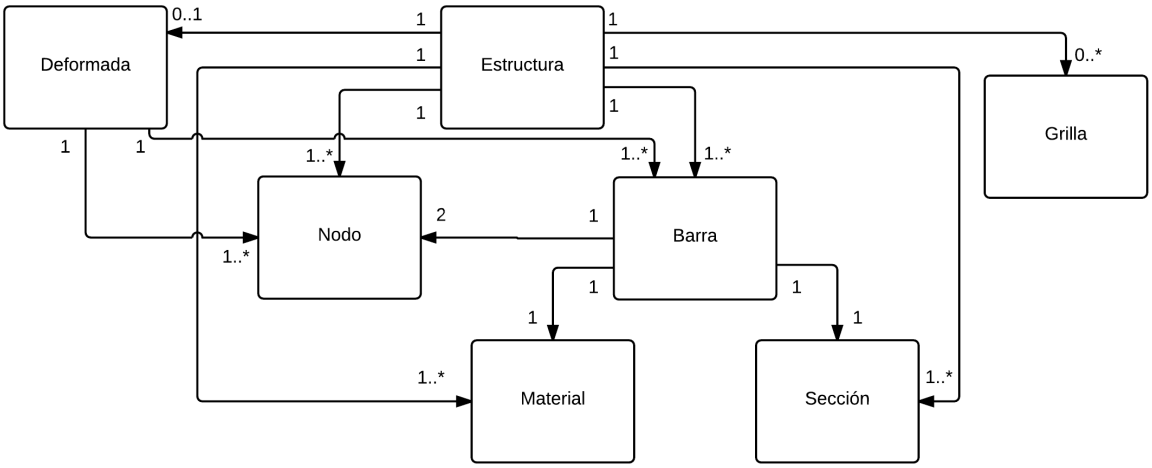

Figura 4: Modelo de dominio de IETFEM.

mantener la simplicidad y la eficiencia de la herramienta por sobre acomplejizar la misma con funcionalidades potentes que serían útiles en un programa profesional. Por ejemplo, se toma en cuenta que en un ámbito académico, el estudiante no ingresará en el sistema estructuras gigantescas (veáse la seción 2.3 del Capítulo 5), y sólo utilizará el mismo para los temas comprendidos dentro del curso que desarrolla. Cabe destacar que además, se busca en un segundo plano, lograr el mayor porcentaje de reusabilidad de código posible, ya que en un futuro, IETFEM puede crecer gradualmente para convertirse en un sistema profesional.

Teniendo en cuenta estos aspectos, sumado a las prestaciones destacadas en las herramientas de desarrollo 3D en la web, y la poca cantidad de sistemas de cálculo de estructuras en la nube, se decidió en conjunto con los tutores, realizar la interfaz en un ambiente web.

Sin embargo, realizar la interfaz en la nube implica ciertas situaciones preocupantes por parte de los tutores, por ejemplo, mantener un servidor donde se aloje la misma una vez finalizado el proyecto. Es necesario entonces destacar ciertas consideraciones sobre la solución elegida:

- Los tutores se sienten a gusto al considerar una versión final en la nube, debido a que la mayoría de este tipo de sistemas son de escritorio y no para todos los sistemas operativos.
- Existe preocupación con respecto al servidor donde se aloje la aplicación. En particular, preocupa justamente encontrar un servidor gratuito donde alojarse y cómo mantener la aplicación una vez finalizado el proyecto.
- Una de las prestaciones actuales del motor de cálculo existente es que al

estar desarrollado en GNU-Octave, permite al estudiante ver el funcionamiento interno del código (o incluso programar nuevos cálculos de acuerdo a sus necesidades), enriqueciendo el proceso de aprendizaje. Dichas características se quieren mantener en la nueva solución.

- Se busca reusabilidad en el código de la aplicación, ya que en un futuro se pretende evolucionar la herramienta a un nivel profesional, donde se pretende que la misma posea diferentes características (por ejemplo, no sería deseable en un sistema profesional que se pueda acceder al código del motor de cálculo directamente).

Tomando en cuenta las mencionadas premisas, la solución propuesta consistió en desarrollar la interfaz como una herramienta web. Sin embargo, no se desplegará la misma en un servidor, sino que se encapsulará la misma en un *framework* que permita ejecutar la misma como una aplicación de escritorio. A ojos del estudiante, la aplicación parecerá ser de escritorio.

Una vez finalizado el dibujado de la estructura, el estudiante podrá generar un archivo con al especificación de la estructura, el cual podrá ingresar en el motor de cálculo de manera manual. Luego, puede desde la interfaz procesar la salida del motor para observar sus resultados.

De esta manera se logran las siguientes características:

- Para la versión inicial, es decir, la versión académica, se ahorra la utilización de un servidor, ya que cada sistema ejecutará en la máquina de cada estudiante. Esto implica que el mantenimiento a posteriori sea nulo por parte de los tutores una vez finalizado el proyecto.
- Se desacoplan el motor y la interfaz, o sea, el estudiante puede visualizar los cálculos realizados en el motor, o incluso programar nuevos, sin necesidad de tocar el código de la interfaz. Es más, el archivo generado por la interfaz será en un formato legible, lo que hace que el estudiante pueda editar el archivo en caso de agregar cálculos nuevos.
- A su vez, esta solución no solo permite agregar cálculos nuevos a los estudiantes, sino que permite que el desarrollo del motor siga avanzando sin entorpecer la interfaz.
- Se obtiene un código totalmente reusable, ya que si en el futuro se quiere evolucionar la herramienta como un producto profesional en la nube, sólo basta con desplegar el código de la aplicación en un servidor.

De esta manera, la herramienta cumple con todas las especificaciones deseadas por los tutores, manteniendo las características positivas de la misma, y a su vez, potenciando la misma en vista de conseguir los objetivos planteados sobre mejorar la eficiencia y la usabilidad.

Finalmente, de acuerdo a las pautas establecidas en la sub-sección anterior, el flujo de la aplicación queda establecido como se muestra en la Figura 5. Los pasos 2 y 3 se anotan como opcionales debido a que el usuario puede cargar una

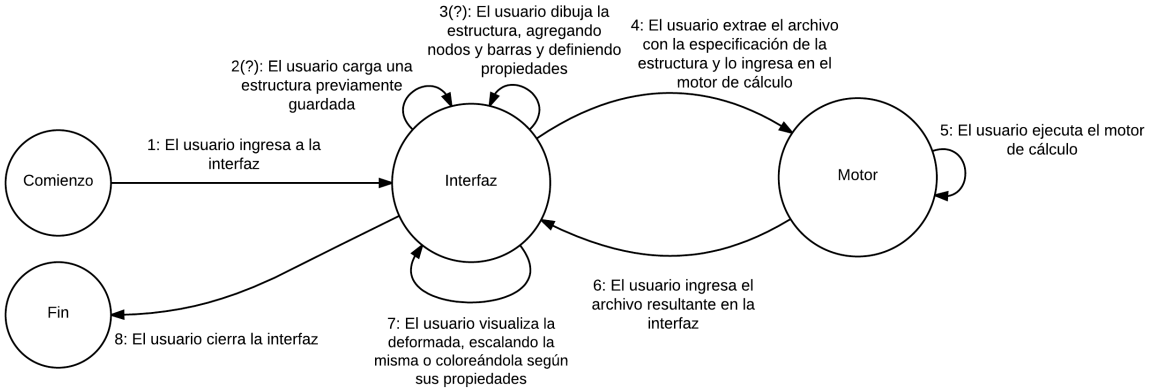

Figura 5: Flujo principal de la aplicación

estructura guardada como dibujar una nueva. Incluso podría realizar ambas, editando una estructura guardada antes de ejecutar el motor

De acuerdo a los casos de uso relevados, se distribuyeron las funcionalidades requeridas en diferentes módulos. Cada uno de estos módulos o subsistemas encapsula operaciones que se relacionan de alguna manera, logrando un nivel bajo de acoplamiento entre cada uno de ellos. Se hablará en detalle de cada subsistema en la siguiente sección.

4.3. Arquitectura

La arquitectura de la aplicación sigue el clásico patrón MVC (Modelo-Vista-Controlador), donde el usuario se encuentra permanentemente interactuando con el sistema, modificando el Modelo (en este caso, la estructura) y visualizando el mismo en el espacio 3D, al que llamaremos Escena.

Como se puede ver en la Figura 6, se agruparon los casos de uso relacionados con el fin de crear diferentes subsistemas encargados de realizar cierto tipo de funcionalidades. Cada una de estas componentes, ofrece al usuario diferentes operaciones que afectan tanto el modelo de la estructura que se mantiene almacenado en la aplicación como lo que se está viendo en el espacio 3D. Por tal motivo, se crearon las componentes «Modelo» y «Escena», las cuáles uniformizan todas las operaciones básicas que se hacen en el modelo de la estructura, y en el dibujo en la escena, respectivamente. Además se destaca el subsistema «Cámara», el cual se relaciona directamente con la escena, encargado de los movimientos del usuario dentro del espacio 3D (Rotación y Desplazamiento) y el subsistema «Deformada», el cual maneja las operaciones básicas que se hacen en la estructura deformada.

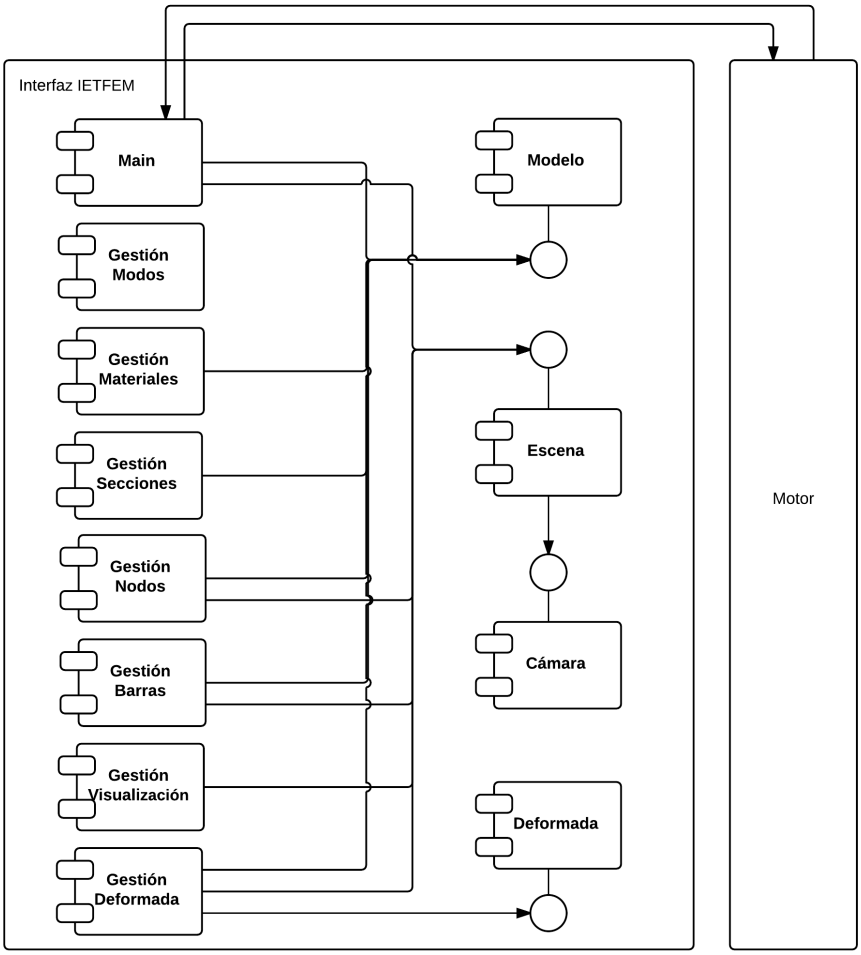

Figura 6: Diagrama de componentes de IETFEM UI

Finalmente, las operaciones que puede realizar el usuario se dividen en 8 subsistemas que se describen a continuación:

Main: Subsistema encargado de realizar la inicialización correcta del sistema. Contiene las operaciones relativas a todo el contexto de la aplicación: Cargar o guardar una nueva estructura, extraer la especificación de la estructura para el motor, y procesar el archivo con los resultados obtenidos. Por último, se encarga de la creación de las grillas auxiliares, ya que se considera una operación relativamente pequeña y poco relevante en el modelo de la estructura como para separarse en un módulo propio.

Gestión Modos: Debido a la necesidad de incluir diferentes características en interacción directa con la escena, se decidió mantener la aplicación en diferentes estados o modos. De esta manera, por ejemplo, un click en la escena realizará diferentes acciones dependiendo de en qué modo se encuentre el usuario. Este pequeño módulo se encarga de gestionar adecuadamente el estado actual y el pasaje entre diferentes estados.

Gestión Materiales: Este subsistema mantiene la creación, modificación y eliminación de Materiales. Debido a la naturaleza de la característica, se accede al mismo mediante un formulario en un menú superior, donde se definen las propiedades de cada material.

Gestión Secciones: Este subsistema mantiene la creación, modificación y eliminación de Secciones. Al igual que en con los materiales, la interacción con dicho módulo se lleva a cabo mediante un formulario.

Gestión Nodos: Este subsistema mantiene la creación, modificación y eliminación de nodos. Los nodos pueden agregarse haciendo click en la escena o ingresando sus coordenadas manualmente. También se ofrece un formulario en donde se pueden agregar propiedades a los nodos: Fuerzas, Condiciones de desplazamiento y Resortes. Además, cada una de estas propiedades agrega a la escena diferentes elementos que indican el valor de cada una de ellas:

- Si se define en el nodo una fuerza aplicada, se dibuja su correspondiente vector apuntando a ese nodo.
- Si se define en el nodo una condicion de desplazamiento en alguna de sus coordenadas, se dibuja una pequeña pirámide de color rojo donde su eje principal tiene la dirección de la propia coordenda en que se define.
- Si se define en el nodo un resorte en alguna de sus coordenadas, se dibuja una pequeña pirámide de color gris donde su eje principal tiene la dirección de la propia coordenda en que se define.

Gestión Barras: Este subsistema mantiene la creación, modificación y eliminación de barras. Las barras se agregan directamente en la escena, seleccionando un nodo inicial y un nodo final. También debe tener asignado un material y una sección, los cuales deben estar previamente definidos y pueden ser seteados una vez dibujada la barra. Se ofrece además la opción de definir propiedades «por

defecto», es decir, se elije un material y una sección, y todas las proximas barras que se dibujen tendrán seteadas dichas propiedades.

Gestión Visualización: Módulo encargado de gestionar la visualización de elementos indicativos en la escena, es decir, muestra u oculta los vectores, resortes y condiciones de desplazamiento definidos en la estructura. También ofrece la posibilidad de escalar los vectores presentes en la escena, de manera de no entorpecer la imagen cuando las fuerzas aplicadas son muy grandes.

Gestión Deformada: Subsistema encargado de gestionar la deformada obtenida del procesamiento de resultados. Ofrece operaciones para visualizar la estructura deformada, escalar deformaciones y colorear la estructura en base a los resultados.

Cada uno de estos subsistemas interactúa con los módulos «Escena», «Modelo» y «Deformada» de acuerdo a sus necesidades:

Modelo: Expone operaciones básicas para modificar el modelado de la estructura que se está ingresando. Cada vez que otro módulo necesite ingresar, modificar o eliminar un nodo, barra, material o sección, llamará a funciones contenidas en este módulo.

Deformada: Expone operaciones básicas para interactuar con la estructura deformada obtenida. Cada vez que el módulo de «Gestión Deformada» deba mover, colorear o transparentar un nodo o barra, se utilizarán funciones expuestas en este módulo

Escena: Expone operaciones básicas para interactuar con el espacio 3D. Cada vez que otro módulo necesite agregar o eliminar cualquier tipo de elemento del espacio 3D, se invocarán funciones expuestas en este módulo

Cámara: Mantiene el manejo de la cámara en la escena. Ejecuta funciones de desplazamiento, rotación y zoom.

De esta manera se define el sistema «IETFEM UI», el cuál interactúa con el sistema «IETFEM Core», que contiene el motor de cálculo, para totalizar lo que sería el IETFEM.

En la Figura 7 se puede preciar la distribución física de la versión estudiantil, donde todo se ejecuta en la máquina del usuario. Se observa la interfaz corriendo sobre un framework que simula ejecutar una aplicación de usuario, y el motor de cálculo en GNU-Octave, comunicándose manualmente mediante la acción del usuario.

En la Figura 8, se aprecia una posible distribución física para una posterior versión, en donde se propone separar físicamente la interfaz del motor. De esta manera se logra una mayor escalabilidad al poder replicar N servidores Web utilizando el mismo motor de cálculo Se propone desplegar la interfaz en un servidor Web, mediante la cuál el usuario accede utilizando el protocolo estándar HTTP. El usuario dibuja la estructura de la misma manera que se realiza en la versión académica, pero al momento de ejecutar el motor, se consume un servicio

REST expuesto por un servidor de aplicación en donde se encuentra corriendo el motor de cálculo, el cual provee a la interfaz con la estructura deformada. Esto implica que el usuario sólo tenga que presionar un botón para deformar la estructura, evitando el proceso de comunicación manual.

4.4. Tecnologías y herramientas utilizadas

4.4.1. HTML5 - JavaScript - CSS3

Existen ciertas tecnologías estándar e ineludibles al momento de desarrollar una aplicación web. Éstas son HTML5[24], JavaScript[25] y CSS3[26]. A continuación se describen brevemente las mismas.

HTML es un lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para su elaboración, definiendo una estructura básica y un código para la definición de contenido como texto, imágenes, videos, entre otros. En la actualidad se encuentra en su versión 5 la cual establece una serie de funcionalidades, elementos y atributos que reflejan el uso típico de los sitios web modernos.

JavaScript , abreviado comúnmente «JS», es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente del lado del cliente implementado como parte de los navegadores web, permitiendo mejoras en la interfaz de usuario y aplicaciones web dinámicas.Es el estándar de facto para scripting en la web y es interpretado por todos los navegadores web.

CSS , actualmente en la versión 3, es un lenguaje de estilos que define la presentación de los documentos HTML. Esto abarca cuestiones relativas a fuentes, colores, margenes, altura, ancho, posicionamiento, etc. Una hoja de estilos fue utilizada en el proyecto con el fin de definir algunos estilos de menús y formularios, esto es en aquellos no definidos o para personalizar los definidos en BootStrap.

4.4.2. BootStrap

Bootstrap [27] es un *«front-end framework» open source*, es la más popular de las librerías HTML,CSS y JS para el desarrollo de páginas web *responsive* y *mobile first*, diseñado para ayudar a construir componentes de la interfaz de usuario.

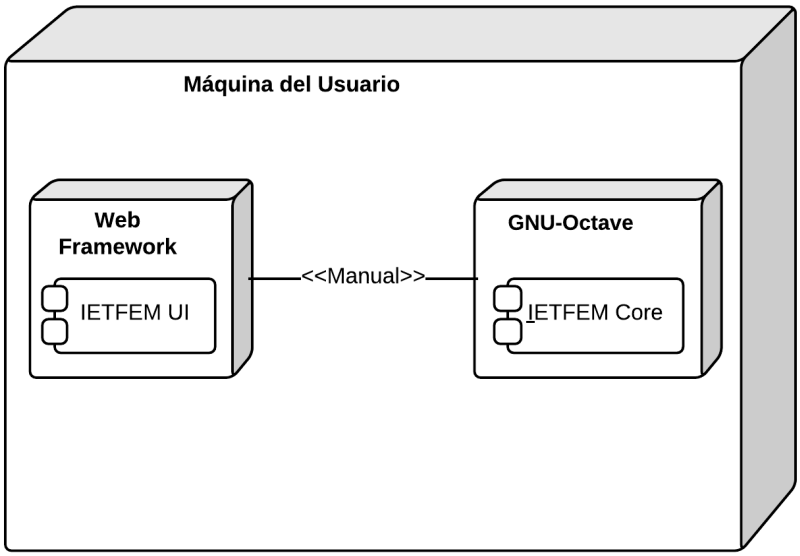

Figura 7: Diagrama de distribución física: IETFEM Estudiantil

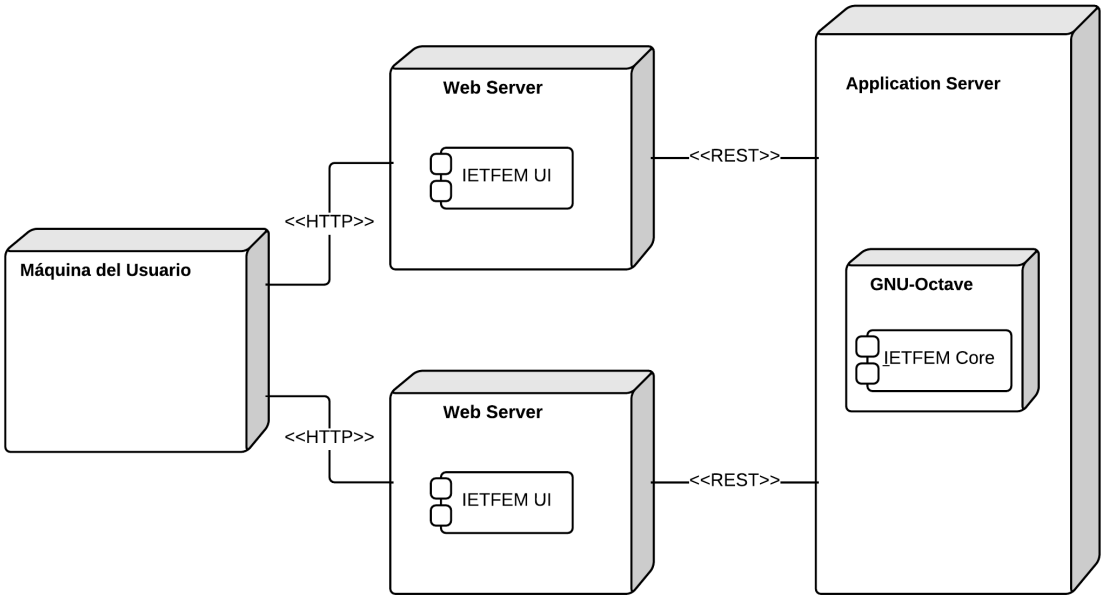

Figura 8: Diagrama de distribución física: IETFEM en la nube

Las principales características del *framework* - las cuales fueron ampliamente utilizadas en el proyecto - son:

- Sistema de grillas *responsive* para posicionar todos los elementos de la página de una manera sencilla.
- Estilos para controles de HTML.
- Componentes personalizados.
- Componentes JavaScript (Ej. Modals).

4.4.3. AngularJS

Es un *framework open source* mantenido por Google que tiene como objetivo solucionar los principales problemas encontrados en el desarrollo de aplicaciones web dinámicas.

AngularJs [28] propone la utilización de programación declarativa para las interfaces de usuario y programación imperativa para lógica de negocio. Implementa el patrón MVC para separar la presentación, datos y lógica. Todo esto da como resultado una aplicación prolija y testeable a nivel de código.

Principales características utilizadas:

- emphTwo way data-binding: esto permite mantener el modelo y la vista (DOM) sincronizados sin necesidad de escribir código especial para mantener dicha sincronización.
- MVC.
- Directivas: sirven para agregar funcionalidad a HTML mediante tags HTML, tanto *built-in* como personalizados.

4.4.4. ThreeJs

Dada la elección de realizar una solución web, la clara superioridad de WebGL para las características gráficas, sumado a la baja productividad y dificultad de desarrollo directamente sobre él, resultó necesario elegir un *framework* que lo abstraiga.

En este sentido se decidió por ThreeJs por ser un proyecto activo, con la mayor cantidad de funcionalidades, buena documentación y una gran comunidad.

En la sección 2.4.3 se describieron las características principales de este tipo de librerías.

4.4.5. Electron

Es un *framework* que permite escribir aplicaciones de escritorio multiplataforma usando HTML, JavaScript y CSS.

Esta herramienta permitió distribuir la aplicación de una manera más elegante que en una carpeta con código y un archivo HTML para ejecutarlo en un navegador local. A los ojos de los usuarios el producto final es una aplicación nativa corriendo transparentemente una implementación mínima del navegador Chromium, solucionando así también posibles problemas de compatibilidad con algunos navegadores.

4.5. Manejo del espacio 3D

4.5.1. Eventos de usuario

El espacio 3D ocupa la mayor parte de la pantalla, y es donde se espera que el usuario realice la mayor parte de interacciones posibles. Se busca que el usuario pueda manejar la escena mediante el uso del mouse, por lo tanto, al inicializar la aplicación se setean *EventListeners* a la ventana para cada tipo de acción con el mouse. Esto significa que la aplicación estará pendiente de todos los movimientos del mouse dentro de la escena. En particular, se tiene en cuenta el movimiento de la cámara y en qué modo se encuentra la aplicación

Los eventos definidos son los siguientes:

- Para el click izquierdo:
  - Si el mismo se presiona y suelta en el mismo lugar:
    - Si se encuentra en modo agregar nodos, y se hace el click encima de un punto de una grilla definida, se agrega el nodo.
    - Si se encuentra en modo agregar barras, y se hace el click encima de un nodo, se selecciona el mismo para agregar una barra desde o hacia él.
    - Si se encuentra en modo seleccionar, y se hace el click sobre un nodo o barra, se selecciona el elemento para modificar sus propiedades.
    - Si se encuentra en otro modo, no realiza ninguna acción.
  - Si el mismo se presiona y se arrastra, se rota la cámara, siempre apuntando al centro de la escena (sin importar el modo).
- Para el click derecho, se desplaza la cámara en dirección a donde se arrastre (sin importar el modo).

- Para el scroll del mouse, se hace zoom in o zoom out dependiendo de la dirección del movimiento (sin importar el modo).
- Para el movimiento del mouse, cuando no se presiona nada, se resalta en color celeste los nodos o barras a los cuáles se les hace Hover (sin importar el modo).

4.5.2. Adición, sustracción y transformación de objetos

El espacio 3D desarrollado se implementó como un objeto de ThreeJs denominado *Scene*, el cual provee operaciones *add()* y *remove()* para agregar y eliminar objetos del mismo. Internamente, una *Scene* contiene una gran cantidad de atributos, entre los que se encuentran la posición, rotación, escalamiento por coordenada, y una lista de objetos en donde se almacenan los elementos que conforman la escena.

Una vez definida la escena se agregan los primeros elementos: se utiliza el objeto *GridHelper* provisto por ThreeJS para definir una grilla auxiliar y se agregan 3 vectores definiendo los ejes x, y y z.

El resto de los objetos que se agregan a la escena se definen como objetos *Mesh* de ThreeJS, definidos por una geometría y un material:

- La geometría de un objeto define su figura geométrica, define si se trata de un cilindro, una esfera, etc.
- El material de un objeto define cómo se ve el mismo en pantalla, es decir, su color, transparencia, escalamiento, etc.

Para representar los nodos se decidió utilizar esferas y para representar las barras se decidió utilizar cilindros. Esta decisión se basa en que son figuras geométricas fácilmente escalables, es decir, si se quiere cambiar el tamaño de un nodo o barra, solo basta con cambiar el radio de su geometría. Por ejemplo, cuando se selecciona un nodo para modificar sus propiedades, el mismo aumenta su tamaño y cambia su color. Esto se logra siguiendo el siguiente proceso:

1. Se obtiene el objeto que se quiere modificar de la lista de objetos de la Escena.
2. Se genera una nueva geometría, en este caso, una esfera más grande.
3. Se genera un nuevo material, en este caso, el mismo que existía pero con un color diferente.
4. Se asignan el material y la geometria nuevos al objeto obtenido en el paso 1.
5. Se renderiza la imagen.

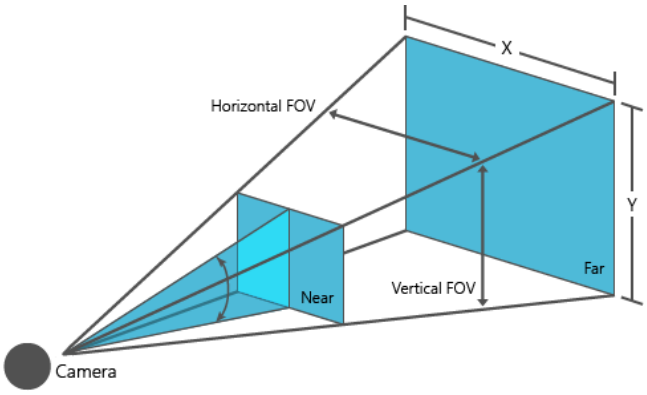

Figura 9: Definición de la cámara en ThreeJS

El renderizado se ejecuta cada vez que se realiza una acción en la escena. Esto significa que cada vez que un usuario modifica la estructura, los cambios quedan inmediatamente reflejados en el espacio 3D. De esta manera logramos una experiencia fluida y totalmente interactiva de dibujado donde la manipulación de la estructura se vuelve una tarea sencilla e intuitiva.

4.5.3. Manejo de la cámara

Una de las prestaciones más grandes que fue percibida en ThreeJS al momento de la investigación fue el sencillo manejo de la cámara. Three provee de un objeto denominado *Camera*, el cuál se define asignando el tamaño del *viewport*, hacia dónde apunta, los planos *«near»* y *«far»* que determinan qué objetos se ven dependiendo de su distancia a la cámara, etc (véase la Figura 9). Incluso permite definir el tipo de perspectiva en que se visualizará el resto del espacio. Una vez definida se setea la misma a la escena, logrando de esta sencilla manera obtener la visualización del espacio 3D.

Hasta el momento, sólo se colocó la cámara en el espacio, sin resolver aún el problema del movimiento de la misma. Aquí es donde entran en juego los llamados *«Controls»*. ThreeJS ofrece en su página web y de manera libre diferentes tipos de controles para la cámara, es decir, movimientos que pueden ser asignados a la misma. En particular, el control *OrbitControls* cumple con todas las características que se buscan para la interfaz: permite rotar la cámara y desplazarla por todo el espacio. Por lo tanto se agregó dicha clase al proyecto y se seteó a la cámara previamente definida.

Al observar la simplicidad del manejo de la cámara utilizando esta herramienta web, se decidió agregar otro juego de cámara-escena en la esquina inferior, la cual siempre apunta a los ejes e imita los movimientos de rotación de la principal, de manera de mantener una referencia al usuario en caso que deba alejarse

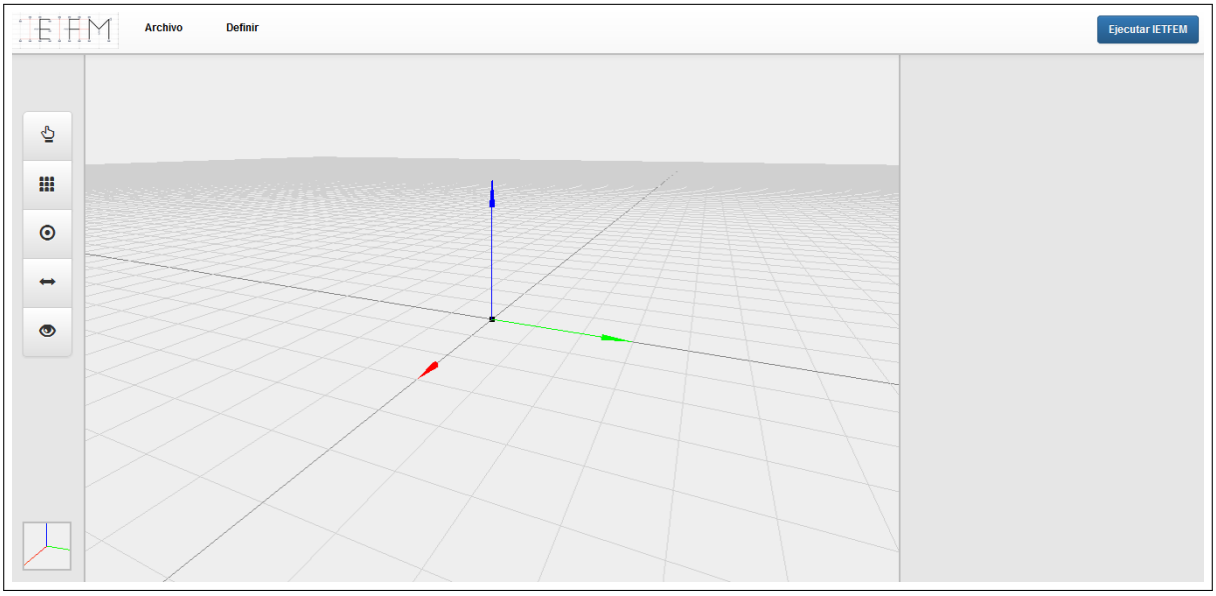

Figura 10: Visualización del Espacio desde la cámara en Perspectiva. También puede observarse la segunda cámara con los ejes en la esquina inferior izquierda.

963 demasiado del origen.

964 El resultado puede observarse en la Figura 10, se utilizó una cámara en pers-
965 pectiva en conjunto con los controles mencionados.

966 4.5.4. Trazado de rayos e intersecciones con objetos

967 El problema de la traza de rayos en la interfaz tomó una significativa importan-
968 cia, debido a que se quiere interactuar con un espacio 3D mediante una superficie
969 bidimensional como lo es la pantalla de una PC. Esto implica por ejemplo que
970 cuando uno hace click en la pantalla, en realidad no se está marcando un punto,
971 sino que se está trazando un rayo entre la cámara y la posición del click, por
972 lo tanto, no se sabe con exactitud en que coordenadas exactas quiere el usuario
973 agregar el nodo.

974 En este sentido, la definición de grillas auxiliares se establece como una solu-
975 ción excelente a este problema. Mediante esta funcionalidad, el cliente define
976 cuadrículas auxiliares que se dibujan en la pantalla, marcando líneas y nodos
977 levemente transparentados.

978 De esta manera, cuando el usuario hace clicks en la pantalla, se traza el corres-
979 pondiente rayo entre la cámara y la posición del click, y si ese rayo intersecta un
980 nodo de alguna grilla, entonces se agrega dicho nodo a la estructura. Dicha es-
981 trategia se utiliza también cuando el sistema se encuentra en modo seleccionar:

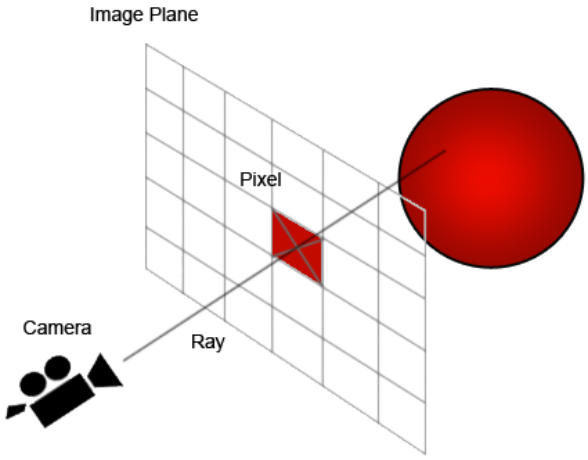

Figura 11: Ejemplo de traza de un rayo entre la cámara y el click del usuario

Se traza un rayo, y si intersecta con un nodo o barra, se destaca el elemento y se cargan sus propiedades para ser modificadas. En la figura 11 puede apreciarse el método descripto anteriormente

ThreeJS provee un objeto denominado *RayCaster* que traza un rayo dado un origen y un vector de dirección. Este objeto expone el método *intersects()*, que dada una lista de objetos, retorna la intersección del rayo con los mismos, obteniendo de esta manera el objeto al cual el usuario intenta referirse.

4.6. Manejo de datos

A pesar de que la estructura se visualiza de manera muy simple en el espacio 3D, se mantiene en segundo plano un complejo modelo que encapsula la información dibujada por el usuario. Es necesario mantener todos los nodos, barras, materiales, secciones, grillas y opciones que el usuario ingresó, ya que en un momento deberá extraer toda la información de la estructura para ingresar en el motor.

4.6.1. Entrada y mantenimiento de información

Como se ha mencionado en reiteradas ocasiones, el usuario puede ingresar la estructura ya sea dibujando una nueva como abriendo una previamente guardada. Cada vez que el usuario modifica elementos en la escena, los mismos cambios se realizan en el modelo que se almacena, manteniendo constante sincronización.

En las Figuras 4.6.1 y 4.6.1 se puede observar la correspondencia entre el dibujo

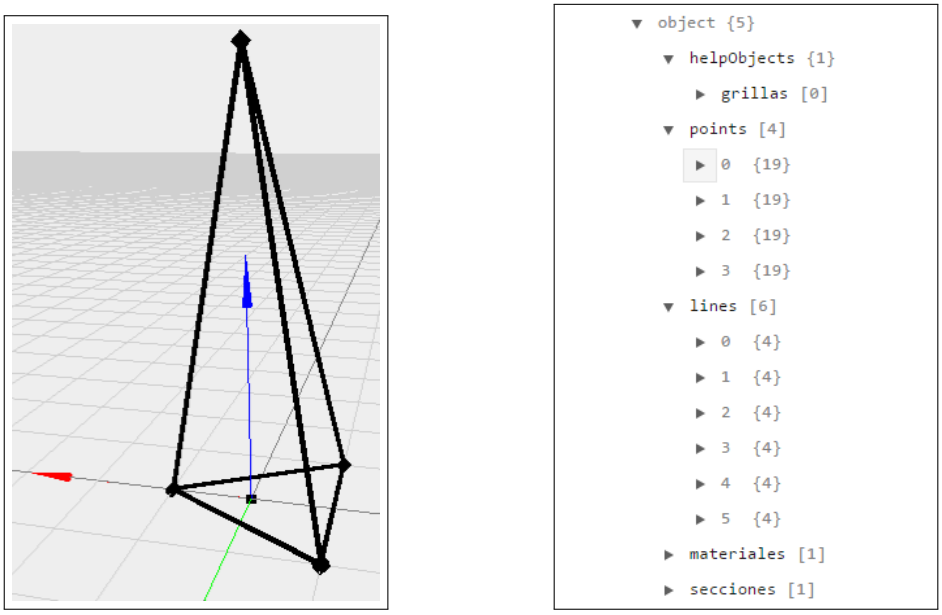

Figura 12: Correspondencia entre dibujo en la escena y el modelo mantenido en segundo plano

presentado al usuario y el modelo que se mantiene en segundo plano. Se observan los nodos, barras, materiales, secciones y grillas definidas en el dibujo. Se hablará más sobre cómo se almacena la estructura en la siguiente sección.

La entrada de datos mediante el dibujo se realiza elemento por elemento. El usuario siempre ingresa un nodo o barra a la vez, lo que hace que la sincronización entre la escena y el modelo sea realmente sencilla. Sin embargo, cuando se carga una estructura previamente guardada, la misma se vuelve realmente complicada, ya que debe limpiarse tanto la escena como el modelo y cargar ambos con la nueva información, asegurándose de mantener la consistencia entre los mismos. Esto se logra mediante una fuerte relación entre el módulo que manipula el modelo y el que manipula la escena. Cada vez que se modifica cualquier aspecto de la estructura, se invocan operaciones de dichos módulos que no sólo realizan el propio cambio, sino que aseguran mantener la consistencia una vez finalizado. Esto significa, por ejemplo, que cargar una nueva estructura, se traduce internamente en una secuencia finita de llamados a operaciones más sencillas como «Agregar nodo» o «Agregar barra» las cuales mantienen el sistema en un estado de sincronía.

4.6.2. Almacenamiento de la estructura

Si bien en la escena provista por Three se almacenan todos los objetos contenidos en la escena, se consideró que debía mantenerse otro tipo de objeto en donde se almacenaran los datos ingresados por el usuario. Esta decisión se basa en que el objeto Escena puede ser difícil de manejar debido a la gran cantidad de atributos que posee, además de que sería necesario sobrecargar cada objeto de Three para agregar los atributos nuevos (propiedades de las barras y nodos).

Básicamente, cada vez que se agrega un elemento a la escena, el mismo se agrega también al modelo. En particular se almacenan los siguientes aspectos:

- **Nodos:**
  - Coordenadas x,y y z.
  - Valores de condiciones de desplazamiento para x,y,z.
  - Valores de resortes para x,y,z.
  - Fuerza aplicada en el punto (coordenadas x,y,z).
  - Id de la esfera dibujada en la escena correspondiente al nodo.
  - Id del vector dibujado en la escena correspondiente a la fuerza aplicada en el nodo.
  - Id de cada pirámide dibujada en la escena que indica una condición de desplazamiento en el nodo.
  - Id de cada pirámide dibujada en la escena que indica un resorte en el nodo
- **Barras:**
  - Id del nodo inicial.
  - Id del nodo final.
  - Id del cilindro dibujado en la escena correspondiente a la barra.
  - Material de la barra.
  - Sección de la barra.
- **Materiales:**
  - Nombre.
  - Modulo de Young.
  - Gamma(γ).
  - Alpha(α).
  - Nu(ν).

- **Secciones:**
  - Área(μ).
- **Grillas:**
  - Id de cada objeto de la grilla dibujado en la escena (puntos y líneas).

A modo de aclaración, se destaca la presencia en el modelo de los ids generados por Three para cada objeto generado en la escena. Esto se debe a que debe mantenerse una referencia entre cada elemento del modelo y su correspondiente objeto en la escena, ya que cualquier modificación que se realice debe aplicar a ambos. Por ejemplo, cuando se elimina un nodo, debe eliminarse tanto del modelo como de la imagen en el espacio 3D. En este caso el proceso sería:

- El usuario hace click en un nodo.
- Se traza un rayo entre la cámara y el pixel, el cuál intersecta con el nodo deseado.
- Se obtiene el id del objeto en la escena.
- Con ese id, se obtiene el objeto del modelo.
- Se elimina el objeto del modelo.
- Se elimina el objeto de la escena.
- Se renderiza.

4.6.3. Salida de Datos

Una vez finalizado el proceso de dibujo, el usuario debe obtener un documento con la especificación de la estructura que pueda ser ingresado en el motor. Para comprender la solución propuesta, debe recapitularse hacia el uso del IETFEM antes de la realización de este proyecto.

Los estudiantes ingresaban la estructura mediante un archivo de especificación en la que debían escribir la posición de cada nodo, la conectividad de cada uno de ellos, los materiales implicados, secciones, y otros aspectos de la estructura y de configuración. Esto debía hacerse además respetando un formato predefinido el cuál derivaba en un fallo del motor en caso de ser omitido, incluyendo saltos de línea y espacios para separar valores.

De esta manera, resultaba sumamente tedioso para el estudiante lograr finalmente el texto correcto con la estructura deseada. Sin embargo esta propiedad del antiguo IETFEM resultó ser de suma utilidad, ya que, intentando reducir al máximo el trabajo realizado en el motor, se decidió utilizar el mismo archivo de entrada al mismo.

Esto quiere decir que la interfaz toma la responsabilidad de generar el archivo de texto que antes debía escribirse a mano, a partir del dibujo realizado por

el estudiante. La interfaz recorre cada aspecto del modelo definido y va confeccionando un archivo .txt manteniendo el formato predefinido por el Motor. Además, mantiene un formato legible para el usuario, con títulos y nombres para cada propiedad a definir.

4.7. Análisis de resultados del Core

Hasta ahora se ha descripto cómo el usuario interactúa con el sistema para generar una estructura. Sin embargo, el sistema incluye además la funcionalidad de visualizar los resultados generados por el motor de cálculo, logrando un nivel aún más alto de amigabilidad.

4.7.1. Generación de resultados

IETFEM generaba, hasta antes de la presencia de la interfaz, diferentes imágenes y gráficas correspondientes a la deformación resultante de la estructura especificada. La adición de la interfaz en el sistema implicó deprecar el sistema de generación de imágenes, ya que ahora, la estructura deformada se vería en la interfaz. Por lo tanto, debía definirse un nuevo modelo de salida del motor para lograr ser interpretado por la interfaz.

La solución que se implementó consta simplemente de agregar al archivo .txt recibido, dos matrices con los resultados obtenidos de los cálculos realizados:

- Una matriz que contiene el desplazamiento de cada nodo en cada coordenada.
- Una matriz que contiene los valores de deformación, fuerza y tensión de cada barra.

Se decidió que en la salida del motor debía incluirse el mismo texto que se recibió, de modo de poder corroborar en la interfaz que se están intentando procesar los resultados de la misma estructura que se generó, evitando así problemas de consistencia y compatibilidad.

4.7.2. Introducción de datos en la UI

Luego de generar el texto con la especificación de la estructura, la interfaz queda en un estado de espera, mientras el usuario obtiene dicho archivo y lo ingresa en el motor. Una vez finalizado el proceso, el cliente carga el archivo resultante en la interfaz y comienza el procesamiento del mismo.

El primer caso consta de separar las 2 matrices resultantes del resto del archivo. De esta manera, se compara el texto generado con el que se recibió, con el objetivo de verificar si se trata de la misma estructura. En caso afirmativo, se

define un nuevo objeto:«Deformada», donde se replican todos los nodos existentes en el modelo, aplicándo a cada uno los desplazamientos obtenidos en la primer matriz. Es importante destacar que la conectividad de la estructura no se ve alterada, por lo tanto no es necesario recibir dicha información en el procesamiento, simplemente se replican todas las barras especificadas en el modelo original, agregando a cada una la información de deformación, fuerza y tensión aplicada a cada una de ellas.

Finalmente, una vez que se logra modelar la estructura deformada, se dibuja la misma en la escena, superpuesta con la estructura original, de modo de apreciar las diferencias entre una y otra.

4.7.3. Visualización

Finalmente se tiene la estructura deformada dibujada en la escena. A partir de este momento, se ofrecen al usuario ciertas opciones para comprender mejor el resultado que se está observando:

- **Transparencia:** El usuario puede elegir transparentar levemente la estructura original o la deformada, de forma de resaltar una u otra de acuerdo a qué se esté analizando.
- **Escalamiento:** Algunas estructuras sufren de deformaciones realmente pequeñas, poco apreciables a la vista. Por lo tanto, se incluye una funcionalidad para escalar interactivamente la deformada. Esto significa que puede multiplicarse los desplazamientos por un número positivo con el fin de observar hacia dónde se realiza el desplazamiento de la estructura. Para hacer esto posible, simplemente se multiplica el desplazamiento de cada nodo por coordenada por el factor de escalamiento ingresado por el usuario, manteniendo la conectividad de la misma.
- **Colorizado:** Esta funcionalidad permite al usuario cambiar la colorización de la estructura dependiendo de los valores recibidos de las propiedades de las barras asignados en la deformada. Es decir, que el usuario puede elegir colorear cada barra de la estructura tomando en cuenta su deformación, fuerza o tensión. Se utilizan escalas de rojo para valores negativos y escalas de verde para valores positivos. Esto se logró recorriendo el modelo de la estructura deformada y cambiando el material que conforma cada barra por el color corespondiente a su valor.

5. Resultados obtenidos

5.1. Comparación IETFEM con y sin UI

5.1.1. Análisis del impacto en la usabilidad

El nacimiento de la propuesta del proyecto fue motivada casi en un 100 % en mejorar la usabilidad de una herramienta que ya resolvía de forma altamente satisfactoria los problemas para la que fue diseñada. En este sentido, se logro solucionar varios problemas que atentaban contra la amigabilidad de la herramienta tales como:

- Generación automática de la entrada para el motor gráfico en GNU-Octave, evitando trabajo manual por parte del usuario e archivo altamente estructurado en el que se pueden introducir muchos errores. Además, dicho archivo crece rápidamente, haciéndolo muy difícil de manejar para estructuras de un tamaño mediano.
- Manipulación visual y en tiempo real de la estructura que se esta generando.
- Experiencia de modelado similar a la utilizada por herramientas comerciales del rubro, lo cual aporta no solo a la usabilidad, sino que también prepara a los estudiantes para la utilización de las mismas en ambientes profesionales.
- Reducción del tiempo de procesado de la salida en comparación al tiempo de generación de gráficos de IETFEM sin interfaz gráfica.
- Guía y simplifica al usuario el flujo de trabajo para integrar su estructura dibujada con el motor y llevar a cabo la ejecución de los cálculos.

La mejora en la usabilidad es percibida, no solo por la opinión del equipo de desarrollo y los docentes tutores, sino por la opinión de alumnos de años anteriores que utilizaron la herramienta sin la interfaz gráfica e interactuaron con la misma en la muestra de ingeniería del presente año.

5.1.2. Análisis del impacto en el tiempo de ejecución

El segundo objetivo planteado en este proyecto trata sobre mejorar la eficiencia del sistema, lo que sin duda representó un desafío tomando en cuenta la potencia con la cual cuenta el motor de cálculo para resolver problemas estructurales.

En consecuencia, se buscó reducir en particular el tiempo de ejecución, logrado una significativa mejora que se puede apreciar en los casos de prueba en la siguiente sección. Debido a la presencia de la interfaz, el motor cuenta con ciertas funcionalidades que ya no son requeridas. Para ilustrar este caso, se observa el funcionamiento del antiguo motor y las mejoras que se realizaron:

Cuando uno ejecuta el IETFEM antiguo, se le plantean una serie de preguntas iniciales, las cuáles incluyen:

- Idioma a manejar(Español e Inglés).
- Tipo de problema (Reticulados, pórticos o arcos).
- Dimensiones del problema de problema(1D, 2D o 3D).
- Pequeñas o grandes deformaciones.
- Nombre del archivo donde está especificada la estructura.

Debido a que se decidió que la interfaz resuelva problemas de pequeñas deformaciones en reticulados, y que todos los problemas se ingresan en 3 dimensiones, se redujo este grupo de opciones solamente al idioma y el nombre del problema. Se decidió que el idioma esté seteado por defecto, debido a que la interacción directa con el motor pasó a ser casi nula. Por lo tanto, cuando uno ejecuta el motor sólo debe especificar el archivo con la estructura, eliminando una significativa cantidad de tiempo muerto al comienzo de cada ejecución.

Respecto al archivo con la estructura mencionado, se redujo el tiempo de confección del mismo, ya que el dibujado se realiza de manera más ágil y sencilla que ingresando cada coordenada y propiedad manualmente. Poder ver la estructura en el proceso, disminuye los errores de conectividad y asignación de propiedades, apoyos, etc, los cuales significaban errores usuales. Además, también se evita al usuario de problemas de sintaxis al escribir el archivo, ya que la interfaz se aseguró de generar el archivo con el formato exacto que necesita el motor.

Otro aspecto a destacar, es la eliminación de funciones de graficado del motor. Se observó que eliminando la generación de imágenes desde el mismo se obtiene una gran mejora en tiempos de ejecución. Se verá el impacto de esta medida en los casos estudiados en la siguiente sección.

Por último, apreciamos la mejora del escalamiento de la deformada. En el antiguo IETFEM, el escalamiento de la deformada era otro atributo que se especificaba en el archivo de entrada, por lo tanto, si uno quería modificar dicho valor debía editar el archivo y ejecutar el IETFEM nuevamente, repitiendo el proceso hasta encontrar el valor deseado. En la nueva versión, con la interfaz incluída, este valor es modificado directamente en la interfaz, luego de ingresar el resultado del motor, evitando al usuario tener que ejecuar nuevamente el motor.

5.2. Casos de prueba

5.2.1. Estudio de casos de pequeño porte (Torre pequeña)

El primer caso a probar consta de una estructura representando una pequeña torre. La misma se asemeja a las estructuras que se realizan en el curso de

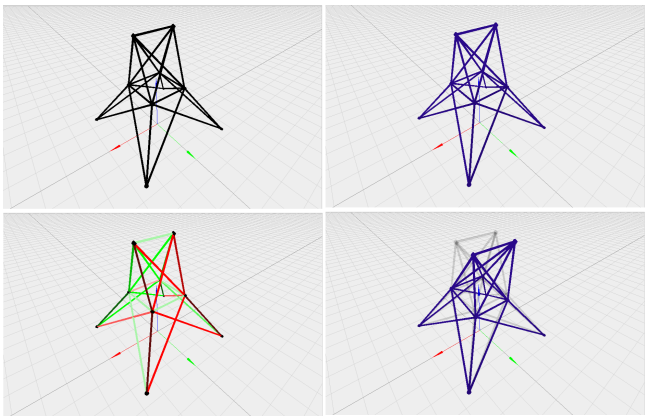

Figura 13: Caso de Estudio: Torre pequeña

Elasticidad en el que se utilizará la herramienta. Cuenta con 10 nodos y 25 barras.

La ejecución del mismo se llevó a cabo en etapas tempranas de desarrollo e integración con el motor, ayudando en el proceso de descubrimiento de *bugs* y resolución de los mismos.

Una vez finalizada la primer versión, se probaron todos los casos de uso especificados, respondiendo a los mismos con resultados positivos y ejecutando con excelente fluidez. En la Figura 13 se puede ver la estructura y los resultados obtenidos. Se aprecia la estructura indeformada y la deformada en color púrpura. También se puede ver la torre sometida a un factor de escala de 200, con el fin de apreciar mejor los desplazamientos de la misma. Por último, se coloreó la estructura de acuerdo a la deformación de cada barra, de manera que puede observarse en rojo las barras que se comprimen y en verde las que se traccionan.

En cuanto a tiempos de ejecución, se observa una significativa mejora, consecuencia de eliminar el módulo de generación de imágenes del motor. Si se ejecuta la torre en cuestión en el antiguo IETFEM, los resultados son los siguientes:

- **Lectura de datos:** 0.227 segundos
- **Cálculo de parámetros internos:** 0.031 segundos
- **Resolución del problema:** 0.017 segundos
- **Generación de imágenes: 370.803 segundos**
- **Generación de TXTs:** 0.281 segundos
- **Generación de TEXs:** 0.499 segundos
- **TIEMPO TOTAL: 371.849 segundos**

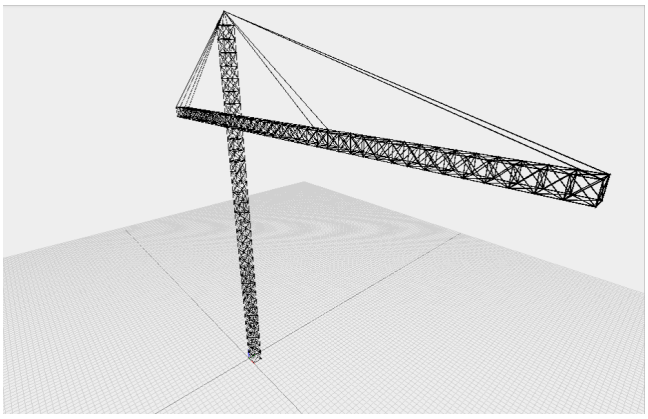

Figura 14: Caso de Estudio: Grúa

Se observa que el tiempo de generación de imágenes supera el 99 % del tiempo total de ejecución, mientras que la resolución del problema en sí se logra en menos de 1 segundo.

En el nuevo IETFEM, la ejecución del mismo archivo muestra los siguientes números:

- **Lectura de datos:** 1.222 segundos
- **Cálculo de parámetros internos:** 0.024 segundos
- **Resolución del problema:** 0.013 segundos
- **TIEMPO TOTAL: 1.259 segundos**

De esta manera, eliminando el proceso de generación de imágenes se ganan más de 5 minutos de eficiencia. Sin embargo, es necesario tomar en cuenta también el tiempo que tarda la interfaz en generar el archivo y el tiempo que tarda en procesar los resultados:

- **Generación del archivo:** 0.102 segundos
- **Carga de resultados:** 0.621 segundos
- **TIEMPO TOTAL: 0.723 segundos**

Se logra así un tiempo total de procesamiento 1.982 segundos entre la descarga/carga de archivos y procesamiento del motor.

5.2.2. Estudio de casos de mediano porte (Grúa)

El siguiente paso en la prueba del sistema fue utilizar un modelo de estructura más grande. El modelo elegido fue una grúa, que puede observarse en la Figura

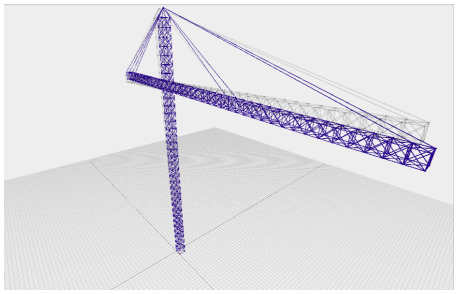

Figura 15: Deformación de la grúa

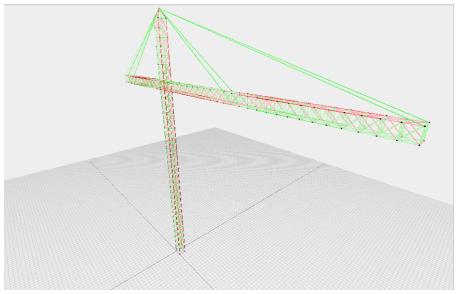

Figura 16: Coloración de la grúa

14. Dicha estructura cuenta con 289 nodos y 1294 barras, lo que se considera un alto nivel de datos considerando el objetivo académico de la estructura. Esto quiere decir que en el curso donde se pondrá en producción el nuevo IETFEM, el estudinte nuncá deberá dibujar una estructura de semejante tamaño. Por lo tanto, consideramos en un principio este caso como una prueba de stress.

No se detectaron bugs de integración con el motor en la ejecución de este caso. El mismo fué ejecutado y perfeccionado poco antes de la muestra de Ingeniería de este año, con el fin de demostrar la potencia y versatilidad de la aplicación. Los casos de uso se ejecutaron sin problemas, notando un nivel alto de fluidez al moverse por el espacio.

En las figuras 15 y 16 se ven los resultados de su ejecución. Se nota la deformacón obtenida sin necesidad de realizar escalamiento alguno. La colorización de la estructura se logró de manera óptima y es coherente con la defoormación de la misma.

En cuanto a tiempos de ejecución, en el IETFEM antiguo se obtuvieron los siguientes números:

- **Lectura de datos:** 5.196 segundos
- **Cálculo de parámetros internos:** 0.766 segundos
- **Resolución del problema:** 0.502 segundos
- **Generación de imágenes: 1598.457 segundos**
- **Generación de TXTs:** 1.543 segundos
- **Generación de TEXs:** 7.555 segundos
- **TIEMPO TOTAL: 1613.851 segundos**

Mientras que en el nuevo IETFEM, los resultados fueron los siguientes:

- **Lectura de datos:** 7.347 segundos
- **Cálculo de parámetros internos:** 0.508 segundos

- **Resolución del problema:** 0.339 segundos
- **TIEMPO TOTAL: 8.194 segundos**

Sumamos el tiempo de descarga/carga del archivo desde la interfaz como en el caso anterior:

- **Generación del archivo:** 0.122 segundos
- **Carga de resultados:** 1.334 segundos
- **TIEMPO TOTAL: 1.456 segundos**

Lo que hace que el tiempo total de procesamiento sea de 9.650 segundos, logrando una ejecución 167 veces más rápida que sin la interfaz

5.2.3. Estudio de casos de gran porte y performance (Torre Eiffel)

Luego de lograr un caso exitoso con la grúa, se estableció que la herramienta cumplía los requisitos necesarios para poder ser utilizado en el curso de Elasticidad. Sin embargo, se decidió intentar un caso aún más masivo para observar su comportamiento. El modelo elegido fue la Torre Eiffel, la cual cuenta con 9746 barras y 2077 nodos.

Lo primero que se observó al cargar el modelo en la interfaz, fue una notoria baja calidad en el movimiento en la cámara. Debido a la masividad de objetos ingresados en la escena, el renderizado consumía demasiado tiempo y le quitaba fluidez a la aplicación.

Motivados por la experiencia de renderizar una estructura enorme como la Torre Eiffel, se comenzó una investigación por parte del equipo de desarrollo de estrategias para atacar dicho problema y mejorar así el rendimiento de la aplicación. Si bien la mayoría de los fixes no serían realizados debido a que la aplicación ya soporta con creces los casos de uso para los que fue diseñada, dicha investigación resulta útil como base para trabajos futuros y por la adquisición de valioso conocimiento académico.

Derivado de la investigación se presentan algunas medidas que se podrían tomar:

- Utilizar threex.rendererstats[30], un monitor que nos da información útil sobre la escena como *draw calls* o cantidad de geometrías lo cual nos puede ayudar a tomar medidas correctivas.
- Ahorrar en la cantidad de geometrías y materiales reutilizando en la medida de lo posible. Existen experiencias en la web que dicen que la velocidad de renderizado se incrementa 4.5 veces para 2000 cubos iguales. (Esta es una medida implementada)
- En el caso de utilizar la geometría esfera, utilizarla con parámetros que deriven en una figura con menor cantidad de caras. Según la investigación la mejora de performance es muy baja.

- En lugar de añadir las geometrías de forma individual en la escena, utilizar la técnica de mergear o unir las mismas en un solo gran objeto. Esta técnica resulta en una mejora substancial en la cantidad de FPS (Frame per second) en la aplicación. De acuerdo a la investigación, se obtiene que se pueden renderizar 120000 cubos a 30FPS mientras que sin esta técnica solo 2000. Aunque la mejora es muy importante, esta forma de trabajo deriva en una complicación a nivel de código importante, debido a que resulta más difícil manipular los objetos individualmente en la escena como en los casos de selección o borrado de objetos.
- Utilizar estructuras de datos más performantes como el octree para tareas como raycasting.
- Utilizar *web workers*[29] con el fin de paralelizar operaciones de calculo que requieran gran procesamiento con el fin de no interferir con el thread de la UI.
- Utilizar *custom shaders*[29] en caso de encontrar un caso factible.
- Eliminar la funcionalidad que implica resaltar barras y nodos al hacer *hover* sobre ellas. Dicha funcionalidad implica un constante *raycasting* de manera de saber que objetos intersectan con la posición del mouse. Esta medida se implementó, logrando una mejora mínima, aunque luego se decidió descartar esta medida debido a que se considera que no vale la pena eliminar esta funcionalidad para mejorar en poca medida casos inalcanzables.

Con respecto a la ejecución del caso en la interfaz, se debe tener en cuenta que además de mantener la escena, el navegador debe mantener por detrás el modelo de la estructura con todas sus características. Además, una vez que se procesan los resultados del motor, se agregan todos los datos de la deformada, lo que hace que el navegador falle en algunos casos por problemas de memoria.

En una versión futura del sistema, este problema podría arreglarse, por ejemplo, manteniendo la estructura persistida en una base de datos, eliminando carga del navegador. Sin embargo, para la versión académica se conoce que el alcance es mucho menor y que nunca se utilizará una estructura de semejante porte. Por lo tanto, consideramos que el caso estudiado aportó al proyecto al mejorar la performance del mismo, pero se expone como un límite conocido el número de elementos de la estructura, sabiendo que para 1500 elementos(grúa), el sistema funciona perfectamente.

6. Conclusiones y trabajo futuro

6.1. Conclusiones

Habiendo finalizado con la implementación de la interfaz «IETFEM UI» se puede afirmar que el proyecto concluyó de forma exitosa.

Como primera conclusión, respondiendo a la pregunta realizada al comienzo del proyecto, se desprende que es posible realizar un sistema de cálculo de estructuras en la nube con un conjunto de funcionalidades acorde. Sin duda es una innovación dentro de este campo, debido a que los sistemas más populares del rubro ejecutan en un ambiente de escritorio. El equipo de desarrollo supone que estos sistemas no han migrado a una plataforma web simplemente por la cantidad de trabajo necesario para recrear la inmensa cantidad de funcionlidades que componen estos programas.

Las herramientas utilizadas se ajustaron a las necesidades. En particular, la herramienta ThreeJs resultó ser clave en el proyecto, demostrando que pueden realizarse aplicaciones 3D de una manera sencilla sobre un navegador Web. Se destaca, entre otros cosas, el amigable uso de la cámara, la cantidad de objetos y funciones previstas, y la documentación publidada, donde se ofrecen ejemplos productivos de cada aspecto de la librería.

En cuanto a las planificaciones y estimaciones realizadas, se transcurrió con muy pocas diferencias con lo planificado, logrando realizar las entregas de iteraciones del producto en tiempo y forma con un buen grado de aceptación por parte de los tutores.

La metodología de trabajo, como se esperaba, logró organizar la gran cantidad de tareas a realizar manteniendo un esquema de proiridades. Permitió además un alto nivel de transparencia con los tutores, dónde se podía observar las tareas pendientes y realizadas, además de agregar cosas a mejorar o posibles *bugs*.

En lo concerniente al producto, se logró un resultado final que supera las expectativas iniciales, tanto del equipo de desarrollo como de los tutores, logrando atacar las más importantes limitaciones de la herramienta que motivaron la realización del proyecto. Se implementaron todas las funcionalidades esenciales logrando una solución completa donde es importante mencionar el entorno 3D de calidad.

Sin duda existe un gran abanico de funcionalidades y mejoras a realizar, pero aún así, la herramienta logró mejorar notablemente su interacción con el usuario, reduciendo drásticamente el tiempo de ejecución y la cantidad de errores de usuario. De esta manera IETFEM podría posicionarse como el primer sistema de cálculo de estructuras desarrollado en América Latina, y uno de los muy pocos en el mundo que ejecuta en un ambiente web.

Finalmente, cabe destacar que se considera que la herramienta alcanzó niveles de

performance altamante satisfactorios, logrando soportar la carga de estructuras de más de 1500 elementos. De esta manera, el equipo se asegura el correcto funcionamiento de IETFEM para los estudiantes que lo utilizarán en el correr del año 2016.

6.2. Trabajo a futuro

6.2.1. Trabajo en la interfaz

Si bien la interfaz cumple con los requisitos y el alcance pretendido para el proyecto, existen ciertos puntos para agregar y mejorar.

En primer lugar y quizás como el punto más directo, se encuentra extender las capacidades de la interfaz de usuario para dar soporte a la resolución de el resto de los problemas que actualmente ya es capaz de resolver el motor de calculo IETFEM. Entre estos problemas se encuentran, por ejemplo: estructuras de vigas, pórticos planos, etc.

Además se pueden agregar varias funcionalidades accesorias como cámara ortogonal o selección múltiple, enriqueciendo la experiencia con la herramienta e incluso asemejándola a la de programas comerciales con años de desarrollo. Con el fin de mejorar la experiencia es posible dedicar esfuerzo también a detalles estéticos de la interfaz para hacerla aún más atractiva.

Un detalle técnico que podría ser importante atacar en el futuro son cuestiones de performance. Esto es principalmente, mejoramiento de estructuras de datos y algoritmos, mejor administración de la memoria utilizada y técnicas más avanzadas en cuanto al desarrollo gráfico. Todas los puntos anteriores permitirían la capacidad de manipular y procesar estructuras muy grandes, mejorando también la fluidez de los gráficos en estas situaciones.

6.2.2. Despliegue de la aplicación

Si bien la solución presentada fue realizada con tecnologías web, la misma se ejecuta en un entorno local (sin servidor que la despliegue) y con la intervención del usuario para correr los cálculos en el motor en GNU-Octave. Sin embargo esta decisión responde a un requerimiento de los tutores con el argumento de evitar el mantenimiento de servidores. Dicho esto, el diseño completo incluye elementos del lado del servidor que se pasarán a detallar brevemente quedando como trabajo futuro.

La solución original plantea un servidor web para el despliegue de la aplicación (interfaz de usuario) y un servidor de aplicaciones con la funcionalidad de ejecutar el código del motor de cálculo con los parámetros recibidos desde la interfaz de usuario y devolver los resultados para que la interfaz los despliegue. Todas

las comunicaciones entre servidores e interfaz se realizarían mediante servicios REST.

De implementarse estos cambios se tendrían varias ventajas para la herramienta y los usuarios de la misma:

- Posibilidad de acceder a la herramienta desde cualquier pc o dispositivo con acceso a Internet.
- Evitar el paso manual de instalar y ejecutar GNU-Octave desde linea de comando siendo esto totalmente transparente para el usuario.
- Capacidad de implementación de funcionalidades apoyándose en las posibilidades que ofrece la infraestructura como autenticación o un espacio de trabajo alojado en el servidor que permita guardar proyectos y resultados en la web.

7. Anexos

A. Especificación de Casos de Uso

- **Nombre:** Alta de Material
- **Descripción:** El usuario agrega un nuevo material en el sistema
- **Precondiciones:** Ninguna
- **Postcondiciones:** Existe un nuevo material definido en el sistema
- **Flujo Normal:**
  1. El usuario indica que quiere agregar un nuevo material
  2. El usuario ingresa las propiedades del nuevo material
  3. El sistema indica que se ha agregado un nuevo material
- **Flujo Alternativo:**
  - El usuario ingresa un valor inválido para alguna de las propiedades del material
    1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Baja de Material
- **Descripción:** El usuario elimina un material existente en el sistema
- **Precondiciones:** Existe al menos un material definido en el sistema

- **Postcondiciones:** El material seleccionado por el usuario se elimina del sistema
- **Flujo Normal:**
    1. El usuario selecciona un material e indica que quiere eliminarlo
    2. El sistema indica que se ha eliminado el material

- **Nombre:** Modificación de Material
- **Descripción:** El usuario modifica un material existente en el sistema
- **Precondiciones:** Existe al menos un material definido en el sistema
- **Postcondiciones:** El material seleccionado por el usuario existe en el sistema con sus propiedades modificadas
- **Flujo Normal:**
    1. El usuario indica que quiere modificar un material
    2. El usuario ingresa las nuevas propiedades del material
    3. El sistema indica que se ha modificado el nuevo material
- **Flujo Alternativo:**
    - El usuario ingresa un valor inválido para alguna de las propiedades del material
        1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Alta de Sección
- **Descripción:** El usuario agrega una nueva sección en el sistema
- **Precondiciones:** Ninguna
- **Postcondiciones:** Existe un nueva sección definido en el sistema
- **Flujo Normal:**
    1. El usuario indica que quiere agregar una nueva sección
    2. El usuario ingresa el área de la nueva sección
    3. El sistema indica que se ha agregado una nueva sección
- **Flujo Alternativo:**
    - El usuario ingresa un valor inválido para el área de la nueva sección

1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Baja de Sección
- **Descripción:** El usuario elimina una sección existente en el sistema
- **Precondiciones:** Existe al menos una sección definida en el sistema
- **Postcondiciones:** La sección seleccionada por el usuario se elimina del sistema
- **Flujo Normal:**
    1. El usuario selecciona una sección e indica que quiere eliminarla
    2. El sistema indica que se ha eliminado la sección

- **Nombre:** Modificación de Sección
- **Descripción:** El usuario modifica una sección existente en el sistema
- **Precondiciones:** Existe al menos una sección definido en el sistema
- **Postcondiciones:** La sección seleccionada por el usuario existe en el sistema con su área modificada
- **Flujo Normal:**
    1. El usuario indica que quiere modificar una sección
    2. El usuario ingresa la nueva área para la sección seleccionada
    3. El sistema indica que se ha modificado la sección
- **Flujo Alternativo:**
    - El usuario ingresa un valor inválido para el área de la sección
        1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Alta de Nodo
- **Descripción:** El usuario agrega una nuevo nodo en el sistema
- **Precondiciones:** No existe en el sistema un nodo con las mismas coordenadas
- **Postcondiciones:** Existe un nuevo nodo en el sistema
- **Flujo Normal:**

1. El usuario indica que quiere agregar una nueva nodo
2. El usuario ingresa el área de la nueva sección
3. El sistema indica que se ha agregado una nueva sección

- **Flujo Alternativo:**
  - El usuario ingresa un valor inválido para el área de la nueva sección
    1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Baja de Nodo
- **Descripción:** El usuario elimina un nodo del sistema.
- **Precondiciones:** Existe al menos un nodo definido en el sistema
- **Postcondiciones:** El nodo seleccionado por el usuario se elimina del sistema
- **Flujo Normal:**
  1. El usuario selecciona un nodo e indica que quiere eliminarlo
  2. El sistema indica que se ha eliminado el nodo

- **Nombre:** Modificación de Nodo
- **Descripción:** El usuario modifica las propiedades de un nodo existente en el sistema
- **Precondiciones:** Existe al menos un nodo definido en el sistema
- **Postcondiciones:** El nodo seleccionado por el usuario existe en el sistema con sus propiedades modificadas
- **Flujo Normal:**
  1. El usuario selecciona un nodo
  2. El usuario cambia las propiedades del nodo por las nuevas
  3. El sistema indica que se ha modificado el nodo
- **Flujo Alternativo:**
  - El usuario ingresa un valor inválido para alguna de las propiedades
    1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Crear grilla
- **Descripción:** El usuario define una grilla auxiliar para el diseño de la estructura y se crea en el sistema
- **Precondiciones:**
- **Postcondiciones:** La grilla definida existe en el sistema
- **Flujo Normal:**
    1. El usuario define las coordenadas de inicio de la grilla
    2. El usuario define la cantidad de lineas y distancia de separación de las mismas
    3. El usuario confirma la creación de la grilla
    4. El sistema indica que se ha creado la grilla con éxito y la misma aparece en la interfaz 3D
- **Flujo Alternativo:**
    - El usuario ingresa un valor inválido para alguna de las propiedades
        1. El sistema indica un mensaje de error correspondiente

- **Nombre:** Modificar Visualización de Propiedades
- **Descripción:** Permite al usuario visualizar de forma gráfica en el modelo 3D apoyos, resortes y fuerzas.
- **Precondiciones:**
- **Postcondiciones:** Dada las opciones de visualización activadas se renderizan en el espacio 3D los gráficos correspondientes.
- **Flujo Normal:**
    1. El usuario activa/desactiva una opción de visualización
    2. El sistema muestra/esconde los gráficos correspondientes en la interfaz
- **Flujo Alternativo:**

- **Nombre:** Nueva Estructura
- **Descripción:** El usuario limpia el modelo y el espacio 3D para comenzar a trabajar con una estructura nueva
- **Precondiciones:**

- **Postcondiciones:** El sistema que en el estado inicial
- **Flujo Normal:**
    1. El usuario elige la opción "Nueva Estructuraz confirma la elección
    2. La interfaz se limpia volviendo al estado inicial
- **Flujo Alternativo:**

- **Nombre:** Guardar Estructura
- **Descripción:** Permite al usuario guardar el trabajo actual para continuar con el mismo en otro momento
- **Precondiciones:**
- **Postcondiciones:** Se descarga un archivo con la información necesaria para poder reconstruir el estado actual del sistema
- **Flujo Normal:**
    1. El usuario elige la opción "Guardar Estructura"
    2. Elige el nombre del archivo a descargar
    3. El archivo se descarga en el dispositivo del usuario
- **Flujo Alternativo:**

- **Nombre:** Abrir Estructura
- **Descripción:** Permite cargar un archivo generado con el procedimiento "Guardar Estructura"dejando al sistema en el estado que este describe.
- **Precondiciones:** Existe un archivo creado con el procedimiento "Guardar Estructura"
- **Postcondiciones:** El sistema queda en el estado descripto por el archivo seleccionado
- **Flujo Normal:**
    1. El usuario elige un archivo del file system
    2. El usuario confirma el cargado
    3. Se aprecia en la interfaz los datos y estructuras cargadas desde el archivo elegido
- **Flujo Alternativo:**

- **Nombre:** Generar Especificación
- **Descripción:** Se genera un archivo reconocible por el motor de cálculo con la especificación de la estructura
- **Precondiciones:**
- **Postcondiciones:** Se descarga un archivo con la especificación de la estructura para utilizar como entrada al motor de calculo
- **Flujo Normal:**
    1. El usuario elige la opción .ªEjecutar IETFEM"
    2. El usuario confirma la descarga del archivo
    3. Se descarga el archivo correspondiente
- **Flujo Alternativo:**

- **Nombre:** Procesar Resultados
- **Descripción:** Se trata de procesar el archivo resultante del motor de calculo, y habilitando una nueva vista para ver la estructura deformada
- **Precondiciones:** Existe un archivo generado por el motor de calculo IETFEM
- **Postcondiciones:** Se procesa el archivo y se genera la vista de exploración de la estructura deformada en la interfaz del sistema
- **Flujo Normal:**
    1. El usuario elige la opción "Procesar resultados"
    2. El usuario selecciona el archivo del file system
    3. Se genera la vista de exploración de la deformada en la interfaz
- **Flujo Alternativo:**
    1. El usuario elige la opción "Procesar resultados"
    2. El usuario elige un archivo invalido
    3. El sistema despliega un mensaje de error correspondiente

- **Nombre:** Escalar Deformada
- **Descripción:** En la vista de exploración de la deformada permite exagerar la deformación en factores lineales para hacerla más apreciable

- **Precondiciones:** Se procesaron los resultados de una estructura generándose en la interfaz la vista de exploración de la estructura deformada
- **Postcondiciones:** Se aprecia la estructura deformada con con el factor de escala dado por el parámetro seleccionado
- **Flujo Normal:**
    1. El usuario elige el valor del factor de escala
    2. Se despliega en la interfaz la estructura deformada con el factor de escala elegido
- **Flujo Alternativo:**
    1. El usuario elige un valor de factor de escala invalido
    2. El sistema despliega un mensaje de error correspondiente

- **Nombre:** Colorear Estructura
- **Descripción:** En la vista de exploración de la deformada permite apreciar en la estructura original las barras coloreadas de acuerdo a tensiones, fuerzas y deformaciones
- **Precondiciones:** Se procesaron los resultados de una estructura generándose en la interfaz la vista de exploración de la estructura deformada
- **Postcondiciones:** Se aprecia la estructura con las escalas de colores correspondientes de acuerdo a los resultados entregados por el motor de cálculos
- **Flujo Normal:**
    1. El usuario activa/desactiva una opción de coloración
    2. Se aprecia en la interfaz la estructura coloreada de acuerdo a los datos del problema
- **Flujo Alternativo:**

B. Entrada del motor: Torre pequeña

El siguiente documento constituye el archivo generado por la interfaz para ser ejecutado desde el motor de cálculo. Dicho archivo, en anteriores versiones de IETFEM (sin interfaz), debía escribirse a mano por el usuario. En él se pueden observar todos los aspectos de la estructura dibujada.

Primero se escriben parámetros seteados por defecto desde la interfaz.

Force Magnitude
kN

Length Magnitude
m

Number of degrees of freedom per node
3

Number of nodes per element
2

Se definen los materiales, con todas sus propiedades.

Number of materials
1

Materials:
Young Modulus gamma alpha (1/C) nu
68947572900 27679.904703 0 0.3

Aquí se setean diferentes temperaturas. Para los casos ingresados desde la interfaz, esta información es innecesaria.

Number of temperature cases
0

Temperature cases:
Value

Se definen las secciones.

Number of sections
1

Sections:
Area
0.00193548

Aquí se definen los nodos, y la posición espacial de cada una de ellos.

Number of nodes
10

Node matrix
Xs Ys Zs
-0.9525 0 5.08
0.9525 0 5.08
-0.9525 0.9525 2.54
0.9525 0.9525 2.54
0.9525 -0.9525 2.54
-0.9525 -0.9525 2.54

-2.54 2.54 0
2.54 2.54 0
2.54 -2.54 0
-2.54 -2.54 0

Posteriormente se definen las barras, indicando material, sección y los nodos de inicio y fin.

Number of elements
25

Conectivity matrix
material section tempcase start end
1 1 0 1 2
1 1 0 1 4
1 1 0 2 3
1 1 0 1 5
1 1 0 2 6
1 1 0 2 4
1 1 0 2 5
1 1 0 1 3
1 1 0 1 6
1 1 0 3 6
1 1 0 4 5
1 1 0 3 4
1 1 0 5 6
1 1 0 3 10
1 1 0 6 7
1 1 0 9 4
1 1 0 5 8
1 1 0 7 4
1 1 0 3 8
1 1 0 10 5
1 1 0 9 6
1 1 0 10 6
1 1 0 7 3
1 1 0 8 4
1 1 0 9 5

Aquí se especifican los nodos que presentan condiciones de desplazamiento, así como su valor para cada coordenada.

Number of displacement conditions nodes
4

Displacement conditions nodes matrix
Displacement node X condition Y condition Z condition

7 0 0 0
8 0 0 0
9 0 0 0
10 0 0 0

Luego se especifican los nodos que presentan fuerzas aplicadas, así como su valor para cada coordenada.

Number of puntual load conditions
4

Puntual loads conditions nodes matrix
Load node FX FY FZ
1 4535.9237 45359.237 -22679.6185
2 0 45359.237 -22679.6185
3 2267.96185 0 0
6 2267.96185 0 0

Análogamente, aquí se definen otro tipo de fuerzas que no son aplicables a los casos ingresados desde la interfaz.

Number of dead volume load conditions
0

Dead volume loads conditions matrix
Element bx by bz

Finalmente, y de igual manera, en esta sección se especifican los nodos que presentan resortes. En este caso, la torre definida no presenta ninguno.

Number of springs conditions nodes
0

Springs conditions nodes matrix
Spring node X condition Y condition Z condition

C. Salida del motor: Torre pequeña

Una vez que se ejecuta el motor de cálculo, el mismo expone un documento con información necesaria para el dibujado de la estructura deformada en la interfaz. Se decidió que el documento contenga además toda la información ingresada en el documento inicial, con el fin de controlar que se está procesando la misma estructura que se dibujó, evitando así problemas de consistencia.

El motor agrega 2 matrices:

La primera matriz contiene el desplazamiento de cada nodo por coordenada. Sólo con esta información ya es posible dibujar la estructura deformada, ya que se mantiene la conectividad y los nodos siempre se unen por líneas rectas.

Desplazamientos de los nodos
u_x u_y u_z
3.47e-004 6.71e-003 -3.86e-004
3.96e-004 6.71e-003 -5.87e-004
1.78e-005 4.48e-004 -1.67e-003
1.11e-004 4.61e-004 -1.80e-003
1.35e-005 4.22e-004 1.07e-003
1.15e-004 4.35e-004 1.19e-003
0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000

La segunda matriz contiene los valores de Deformación, Fuerza y Tensión para cada barra. Esta información se utiliza en la interfaz para colorear la estructura y apreciar, por ejemplo, cuáles barras se comprimen y cuáles se estiran.

Parametros en barras
Deformación Fuerza Tension
2.60e-005 3.47e+003 1.79e+006
-2.56e-004 -3.42e+004 -1.77e+007
-2.27e-004 -3.02e+004 -1.56e+007
1.52e-004 2.03e+004 1.05e+007
1.81e-004 2.42e+004 1.25e+007
-3.91e-004 -5.22e+004 -2.70e+007
2.43e-004 3.25e+004 1.68e+007
-3.67e-004 -4.89e+004 -2.53e+007
2.68e-004 3.57e+004 1.84e+007
6.40e-006 8.54e+002 4.41e+005
2.01e-005 2.68e+003 1.39e+006
4.90e-005 6.54e+003 3.38e+006
-5.35e-005 -7.14e+003 -3.69e+006
-1.25e-004 -1.67e+004 -8.65e+006
7.98e-005 1.07e+004 5.51e+006
-1.48e-004 -1.98e+004 -1.02e+007
5.72e-005 7.63e+003 3.94e+006
-2.32e-004 -3.10e+004 -1.60e+007
-2.37e-004 -3.16e+004 -1.63e+007
1.62e-004 2.16e+004 1.12e+007
1.57e-004 2.09e+004 1.08e+007

3.40e-004 4.53e+004 2.34e+007
-4.29e-004 -5.72e+004 -2.96e+007
-4.76e-004 -6.36e+004 -3.28e+007
2.92e-004 3.90e+004 2.01e+007

Referencias

[1] *GNU-Octave:* https://www.gnu.org/software/octave/

[2] *IETFEM:* Una herramienta de código abierto aplicada a la enseñanza del Método de Elementos Finitos en Ingeniería - http://www.ing.unrc.edu.ar/raei/archivos/img/arc_2015-04-22_02_19_48-07.pdf

[3] *SAP 2000:* https://www.csiamerica.com/products/sap2000

[4] *AxisVM:* http://axisvm.eu/index.html

[5] *MatLab:* http://www.mathworks.com/products/matlab

[6] *Método de los elementos finitos:* http://www.iit.upcomillas.es/ carnicero/Resistencia/Introduccion_al_MEF.pdf

[7] *Computers & Structures.Inc:* https://www.csiamerica.com

[8] *Alcance mundial de SAP2000:* https://www.csiamerica.com/about

[9] *Idea StatiCa:* https://www.ideastatica.com

[10] *CloudCalc:* http://www.cloudcalc.com

[11] *SkyCiv:* https://skyciv.com

[12] *OpenGL:* https://www.opengl.org/documentation/current_version

[13] *Lightweight Java Game Library 3:* https://www.lwjgl.org

[14] *Java OpenGL:* https://en.wikipedia.org/wiki/Java_OpenGL

[15] *JMonkey Engine:* http://jmonkeyengine.org/tour/introduction

[16] *HTML Canvas 2D:* http://www.w3.org/TR/2dcontext

[17] *WebGL:* https://en.wikipedia.org/wiki/WebGL

[18] *Experience Curiosity:* http://eyes.nasa.gov/curiosity

[19] *ThreeJS:* http://threejs.org

[20] *BabylonJS:* http://www.babylonjs.com

[21] *Metodología Kanban:* http://kanbantool.com/es/metodologia-kanban

[22] *Jira:* https://es.atlassian.com/software/jira

[23] *TFS:* https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx

[24] *HTML5:* https://es.wikipedia.org/wiki/HTML5

[25] *JavaScript:* https://www.javascript.com/

[26] *CSS3:* https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada#CSS3

[27] *Bootstrap:* http://getbootstrap.com

[28] *AngularJS:* https://angularjs.org

[29] *Optimizing ThreeJS Blog:* http://www.ianww.com/blog/2012/11/04/optimizing-three-dot-js-performance-simulating-tens-of-thousands-of-independent-moving-objects

[30] *Performance Monitor:* http://learningthreejs.com/blog/2013/06/25/monitor-rendering-performance-within-threejs