VisualARQ Drawings Viewer: Aplicación para consulta, revisión y colaboración en proyectos BIM

Memoria de Proyecto Final de Máster

Máster Universitario en Aplicaciones Multimedia

Área profesionalizadora

Autor: Juan Ramón Cárceles Román

Consultor: Sergio Schvarstein Liuboschetz

Profesor: Laura Porta Simó

3 de enero de 2020

Créditos/Copyright

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada
3.0 España de CreativeCommons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*VisualARQ Drawings Viewer: Aplicación para consulta, revisión y colaboración en proyectos BIM*
Nombre del autor:	*Juan Ramón Cárceles Román*
Nombre del consultor/a:	*Sergio Schvarstein Liuboschetz*
Nombre del PRA:	*Laura Porta Simó*
Fecha de entrega (mm/aaaa):	01/2020
Titulación:	*Máster Universitario en Aplicaciones Multimedia*
Área del Trabajo Final:	*Área profesionalizadora*
Idioma del trabajo:	*Castellano*
Palabras clave	*Arquitectura, BIM, Planos, Visor*

Resumen del Trabajo (máximo 250 palabras):

Este proyecto tiene como objetivo desarrollar una aplicación web complementaria a un software BIM[1] existente (Rhinoceros3D + VisualARQ) que trabaje sobre dibujos inteligentes obtenidos de un modelo BIM y permita realizar tareas de consulta, revisión y colaboración.

Debido al incremento en la complejidad y los requisitos de los proyectos de construcción cada vez son necesarias más herramientas digitales que faciliten el proceso a los implicados en el sector de la arquitectura la ingeniería y la construcción. Actualmente el BIM es la metodología más destacada que pretende resolver el problema y que cada vez es obligatoria en más países. Las herramientas que implementan el BIM tienen el objetivo de permitir hacer más cómodo y eficiente todo el proceso de creación gracias a permitir crear una maqueta virtual del edificio o infraestructura final.

Debido a que el proceso de desarrollo de una construcción consta de muchas fases, desde el diseño hasta la construcción y mantenimiento, es difícil que un solo programa pueda abarcarlas todas. Por este motivo existen varios tipos de herramientas que se utilizan en fases diferentes o para tareas complementarias. El reciente auge del BIM ha hecho que la competencia cada vez sea más dura y que se busquen soluciones que destaquen un producto sobre los demás.

Espero que el resultado sea la base de un producto que siga creciendo para convertirse en una herramienta realmente útil para cualquier profesional implicado en el la arquitectura y/o construcción.

Abstract (in English, 250 words or less):

This project aims to the development of a web app, auxiliary to an existent BIM software (Rhinoceros 3D + VisualARQ), that works on “smart” drawings obtained from a BIM model and that allows carrying out query (consultation), review and collaboration tasks.

Due to the growing complexity and requirements in the construction projects, new digital resources are more and more necessary to facilitate the work process to the professionals involved in the architecture and engineering field. Nowadays BIM is the most emerging

[1] Building Information Modeling

technology in solving this problem and its use is gradually becoming mandatory in many countries. The tools that use the BIM technology aim to make the creation process easier and more efficient, thanks to the possibility to make a virtual model of the final building or infrastructure.

Since the developing process of a construction consists of many phases, from design to construction and maintenance, it's very difficult that they can be all covered with just one program. This is the reason why there are many tools that can be used in different steps of the project and for complementary tasks. The recent rise of BIM has created a very hard competition among brands that leads to the research of new solutions that can distinguish the product among the others.

What I expect is that the result could be the starting point for a growing product, which eventually could be useful for every professional implied in the architecture and engineering field.

Abstract

This project aims to the development of a web app, auxiliary to an existent BIM software (Rhinoceros 3D + VisualARQ), that works on “smart” drawings obtained from a BIM model and that allows carrying out query (consultation), review and collaboration tasks.
Due to the growing complexity and requirements in the construction projects, new digital resources are more and more necessary to facilitate the work process to the professionals involved in the architecture and engineering field. Nowadays BIM is the most emerging technology in solving this problem and its use is gradually becoming mandatory in many countries. The tools that use the BIM technology aim to make the creation process easier and more efficient, thanks to the possibility to make a virtual model of the final building or infrastructure.
Since the developing process of a construction consists of many phases, from design to construction and maintenance, it’s very difficult that they can be all covered with just one program. This is the reason why there are many tools that can be used in different steps of the project and for complementary tasks.
The recent rise of BIM has created a very hard competition among brands that leads to the research of new solutions that can distinguish the product among the others.
What I expect is that the result could be the starting point for a growing product, which eventually could be useful for every professional implied in the architecture and engineering field.

Resumen

Este proyecto tiene como objetivo desarrollar una aplicación web complementaria a un software BIM existente (Rhinoceros3D + VisualARQ) que trabaje sobre dibujos inteligentes obtenidos de un modelo BIM y permita realizar tareas de consulta, revisión y colaboración.
Debido al incremento en la complejidad y los requisitos de los proyectos de construcción cada vez son necesarias más herramientas digitales que faciliten el proceso a los implicados en el sector de la arquitectura la ingeniería y la construcción. Actualmente el BIM es la metodología más destacada que pretende resolver el problema y que cada vez es obligatoria en más países. Las herramientas que implementan el BIM tienen el objetivo de permitir hacer más cómodo y eficiente todo el proceso de creación gracias a permitir crear una maqueta virtual del edificio o infraestructura final.
Debido a que el proceso de desarrollo de una construcción consta de muchas fases, desde el diseño hasta la construcción y mantenimiento, es difícil que un solo programa pueda abarcarlas todas. Por este motivo existen varios tipos de herramientas que se utilizan en fases diferentes o para tareas complementarias. El reciente auge del BIM ha hecho que la competencia cada vez sea más dura y que se busquen soluciones que destaquen un producto sobre los demás.
Espero que el resultado sea la base de un producto que siga creciendo para convertirse en una herramienta realmente útil para cualquier profesional implicado en el la arquitectura y/o construcción.

Palabras clave

Arquitectura, BIM, Planos, Visor

Notaciones y Convenciones

He utilizado un tipo de fuente monospace para indicar elementos o fragmentos de texto que representan código. Por ejemplo:

```
<p>Esto es un ejemplo.</p>
```

Índice

Capítulo 1: Introducción 10
1. Prefacio 10
1.1 El Building Information Modeling 10
1.2 Motivación personal 11
2. Descripción 12
2.1 Contexto 12
2.2 Propuesta 12
2.3 Relevancia 13
3. Objetivos generales 14
3.1 Objetivos principales 14
4. Metodología y proceso de trabajo 15
5. Planificación 16
6. Presupuesto 17
Capítulo 2: Análisis 18
1. Estado del arte 18
2. Análisis del mercado 19
2.1 Comparativa de la competencia 19
2.2 Modelo de negocio 24
3. Público objetivo y perfiles de usuario 26
4. Definición de las especificaciones del producto 27
Capítulo 3: Diseño 28
1. Arquitectura general de la aplicación 28
2. Arquitectura de la información y diagramas de navegación 29
2.1 Diagramas de navegación 29
2.2 Estructura HTML general 30
2.3 El contenedor de los canvas <svg> 30
2.4 Estructura de los archivos SVG que exportará Rhino + VisualARQ 30
2.5 Arquitectura de los archivos almacenados en Google Drive 31
2.6 Arquitectura del archivo de proyecto que será exportado por Rhino + VisualARQ 32
2.7 Arquitectura de clases 33
3. Diseño gráfico e interfaces 36

3.1 Estilos 36
3.2 Bocetos y wireframes 36
4. Lenguajes de programación y APIs utilizadas 40
4.1 Software 40
4.2 JavaScript 40
4.3 API Google Drive 40
4.4 Firebase 42
Capítulo 4: Demostración 43
1. Instrucciones de uso 43
2. Pantallas 46
3. Tests 48
Capítulo 5: Conclusiones y líneas de futuro 49
1. Conclusiones 49
2. Líneas de futuro 50
Bibliografía 51
Anexos 53
Anexo A: Entregables del proyecto 53

Figuras y tablas

Índice de figuras

Figura 1: Modelo BIM de la Villa Savoye creado con VisualARQ 11
Figura 2: Concepto de exportación del modelo BIM con toda la información desde Rhino + VisualARQ 12
Figura 3: Captura de pantalla del contenido de un archivo que imita al que exportará Rhino + VisualARQ 33
Figura 4: A la izquierda el logo de VisualARQ Drawings Viewer y a la derecha el de VisualARQ 36
Figura 5: Dibujos iniciales a mano alzada de las pantallas principales de la aplicación 37

Índice de tablas

Tabla 1: Tabla resumen de la estimación del presupuesto 17
Tabla 2: Tabla con la descripción de cada una de las pantallas principales de la aplicación 29

Capítulo 1: Introducción

1. Prefacio

1.1 El Building Information Modeling

El tema central en el cual se apoya mi proyecto es el BIM (*Building Information Modeling*), ya que se trata de una aplicación complementaria a un software de modelado BIM.

El BIM es una tecnología utilizada en el campo de la arquitectura, ingeniería y construcción que goza actualmente de un crecimiento continuado. El motivo es la conveniencia de aprovechar las ventajas que ofrece en comparación a los métodos de trabajo tradicionales (software CAD) más parecidos a la manera de trabajar anterior a la aparición de los ordenadores.

La tecnología BIM consiste en la creación de un modelo digital de un edificio o infraestructura el cual incluye toda la información del proyecto. Esto se consigue gracias a realizar el modelo 3D mediante elementos constructivos (muros, puertas…) que contienen información como dimensiones, materiales… Las principales ventajas de trabajar con la tecnología BIM son:

- Facilitar tareas de documentación, ya que a partir del modelo 3D se pueden obtener automáticamente los planos y secciones (se puede trabajar en 3D y 2D en cualquier momento).
- Visualizar el proyecto en 3D continuamente y así tomar mejores decisiones de diseño arquitectónico.
- Permite cuantificar información y realizar varios tipos de cálculos (mediciones, costes, cálculos energéticos…) los cuales ayudan a optimizar el proceso y el resultado.
- Aunque se trata de un aspecto avanzado también la posibilidad de colaborar es importante, de tal manera que el modelo 3D funcione como una base de datos común en la cual participen los varios actores de las varias disciplinas implicadas en el proyecto: arquitectos, ingenieros de estructuras, ingenieros de instalaciones, constructores…

Debido a la magnitud y diversidad de las tareas se requiere más de un software para poderlas llevar a cabo todas y normalmente existen paquetes de software que entre ellos se complementan y cada uno de los cuales está enfocado a una tarea diferente del proceso BIM.

Figura 1: Modelo BIM de la Villa Savoye creado con VisualARQ

1.2 Motivación personal

Estudié arquitectura y hasta hace poco trabajé como arquitecto. Actualmente trabajo en una empresa de informática dando soporte y gestionando un producto de arquitectura, concretamente se trata de VisualARQ[2], un plugin para Rhino3d[3], el cual añade características BIM (*Building Information Modeling*).

[2] https://www.visualarq.com/
[3] https://www.rhino3d.com/

2. Descripción

2.1 Contexto

Rhinoceros3D + VisualARQ es un paquete de software que permite crear un modelo 3D de un edificio o infraestructura con información de sus componentes, es decir un modelo BIM (*Building Information Modeling*). Por ejemplo en el caso de un muro el programa te indica su composición, su área, su longitud, etc, en el caso de una biga te indica su perfil, sus dimensiones, etc. Gracias a toda esta información se consigue hacer más eficiente todo el proceso de diseño hasta la construcción y agiliza muchos de los cálculos necesarios.

2.2 Propuesta

Mi propuesta de proyecto consiste en realizar una aplicación web que permita visualizar documentos de proyectos modelados en Rhinoceros3D + VisualARQ. Esta aplicación interpretará un archivo o conjunto de archivos generados por un plugin exportador. Los archivos serán planos del edificio o infraestructura que contendrán toda la información del modelo que la aplicación web necesitará.
La necesidad principal que la de la aplicación cubrirá será el de permitir a los usuarios y sus colaboradores visualizar la documentación técnica del proyecto desde cualquier dispositivo, además de poder realizar tareas de consulta, revisión y colaboración sobre esta. El objetivo de la aplicación web por lo tanto no será el de poder modificar el modelo original, sino el de realizar tareas complementarias.

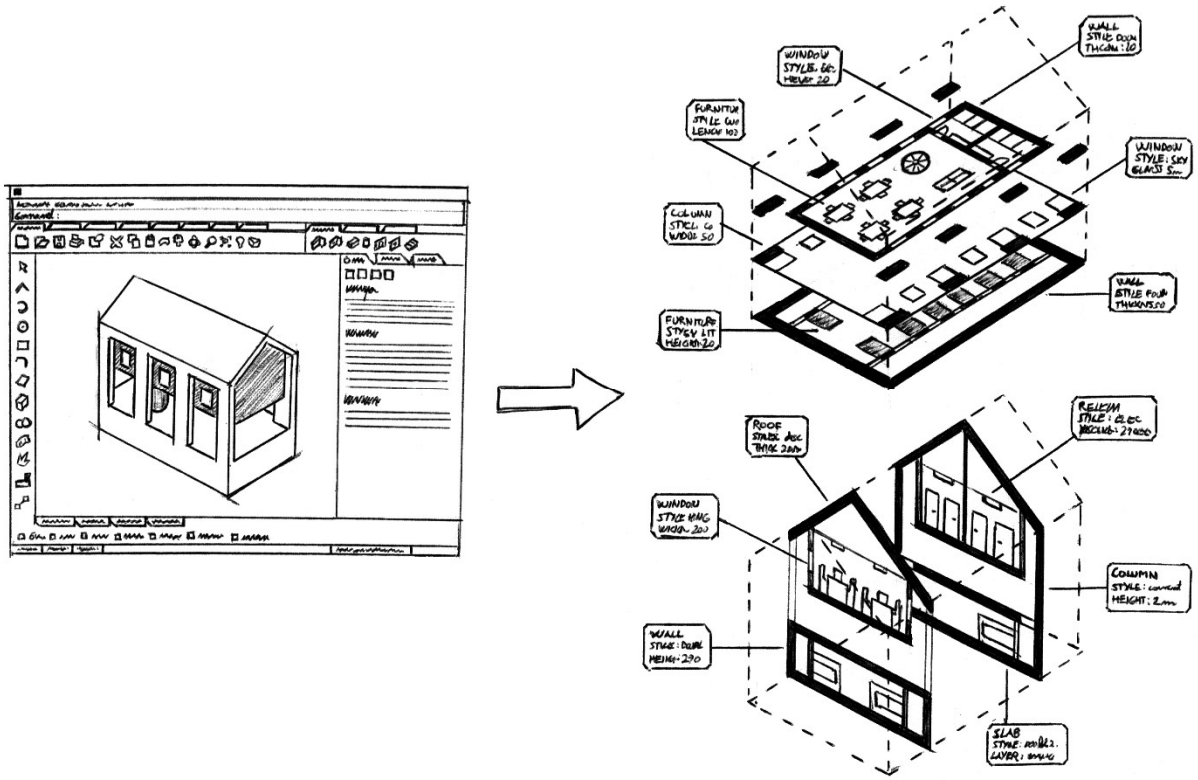


Figura 2: Concepto de exportación del modelo BIM con toda la información desde Rhino + VisualARQ

El proyecto real por lo tanto consiste en dos partes:

- El exportador que generaría el archivo/s con todos los datos.
- La aplicación web que interpretaría el archivo/s exportado para ofrecer el servicio al usuario.

Para el TFM solo trato la parte correspondiente a la aplicación web. La parte del exportador se desarrollará más adelante. Para desarrollar la aplicación se trabajará con archivos generados manualmente que imitarán a los que se exportarían. Se trata de archivos que contendrán información en SVG y JSON principalmente.

2.3 Relevancia

Es relevante ya que actualmente nuestro producto (VisualARQ) no dispone de ninguna herramienta complementaria que permita llevar a cabo funciones de este tipo, es decir funciones de colaboración y revisión. En parte el motivo es que se trata de un producto relativamente joven y pequeño en comparación a la competencia. Otras plataformas hace tiempo que disponen de aplicaciones móviles complementarias. Por eso el objetivo es ofrecer un producto que complemente y de valor a VisualARQ.

3. Objetivos generales

3.1 Objetivos principales

Objetivos de la aplicación:

- Que esté disponible desde cualquier dispositivo.
- Que permita visualizar documentos de un proyecto.
- Que permita consultar información de un proyecto.
- Que permita colaborar.

Objetivos para el cliente / usuario:

- Disponer de un paquete de aplicaciones BIM más completo.
- Poder realizar más tareas sin tener que cambiar de conjunto de productos, es decir sin tener que buscar alternativas lo cual a menudo requiere aprender de cero un nuevo programa.
- Obtener una aplicación BIM para el trabajo que sea fácilmente utilizable desde cualquier dispositivo.

Objetivos personales del autor del TF:

- Aprender a desarrollar un producto y entender todas las dificultades asociadas.
- Aprender a organizar y planificar tareas relacionadas con el desarrollo de aplicaciones, la importancia de cada una de ellas y el tiempo y esfuerzo necesario para completarlas.
- Ofrecer un producto válido que pueda ser utilizado pero usuarios reales de VisualARQ, lo cual haga de VisualARQ un producto más atractivo y de esta manera se incrementen sus ventas.

4. Metodología y proceso de trabajo

Se trata del desarrollo de un producto nuevo como complemento de uno existente, por este motivo algunas decisiones han sido tomadas en base a aspectos reales.

El contacto constante con compañeros de trabajo que desarrollan el software principal ha sido imprescindible ya que me ha ayudado a tomar decisiones tanto de estructuración del codigo del proyecto hasta entender limitaciones y poder reorientar o buscar alternativas.

5. Planificación

La última fase gracias a disponer de la base de toda la aplicación desarrollada hasta el momento ha servido sobre todo para conseguir desarrollar las últimas características que muestran el verdadero potencial del producto. Algunas estaban ya previstas y fueron desplazadas a esta última fase tras la reorganización de la planificación durante la PEC4, como es el caso de *compartir documentos*, lo cual se ha transformado en *colaboración*.
Otras tareas no estaban previstas ya que no podía prever hasta donde iba a llegar como es el caso del uso de *service workers* para poder hacer la aplicación instalable. En otros casos es porque cuando empecé no tenía los conocimientos técnicos suficientes y gracias a haberlos aprendido durante el desarrollo he sido capaz de incluirlas en la planificación, este es el caso de las *notificaciones push*.

Viendo como se ha cambiado la planificación final creo que esta última tiene más sentido que la inicial ya que es más realista. No obstante me parece complicado y más todavía sin poder trabajar en equipo y con poca experiencia crear una planificación completa al inicio y que esta no sufra varios cambios.

El aspecto más importante que tuve que cambiar en la planificación durante el desarrollo del proyecto y sobre el cual no estoy contento fue la arquitectura del código de la aplicación. Concretamente la decisión de hacer uso de programación orientada a objetos, la cual estaba prevista para la primera fase (PEC3) pero terminó en la segunda ya que no fue hasta ese momento que vi más claro como tenía que hacerlo y consecuentemente perdí tiempo.

La planificación final se puede encontrar referenciada en el **Anexo A: Planificación del proyecto**.

6. Presupuesto

Se ha realizado una estimación del coste que ha supuesto el proyecto hasta ahora y de lo que se prevé en el futuro con todo lo que faltaría para que una primera versión del producto estuviese lista.

El documento con la estimación detallada se encuentra en el **Anexo A: Estimación de presupuesto**. A continuación se muestra una tabla resumen:

	Estimación del presupuesto	
	Horas	Coste
Primera fase (TFM)	257	7.710 €
Segunda fase	944	28.320 €
Total	1201	36.030 €

Tabla 1: Tabla resumen de la estimación del presupuesto

Capítulo 2: Análisis

1. Estado del arte

El contexto, como ya he mencionado en más de una ocasión en el capítulo anterior es el campo de la arquitectura, la ingeniería y la construcción. Más en concreto la metodología BIM, la cual es implementada por varios paquetes de software actualmente.

El incremento en la complejidad de la construcción y los elevados requisitos por parte de la administración ha hecho que el uso BIM se haya convertido en obligatorio en muchos países. Ante esta situación el técnico se encuentra con la obligación de elegir entre una serie de productos.

Hay varias dificultades como el elevado coste que tienen estos programas, la dificultad de su aprendizaje y el tener que cambiar de plataforma cada vez que se cambia de fase en el proyecto entre otras. También la rigidez a la hora de diseñar es algo muy criticado por parte de los arquitectos lo cual hace que se muestran reacios y prefieren técnicas convencionales más manuales que les dan total libertad aunque luego tengan que repetir el trabajo dos veces.

Ningún programa de los que actualmente implementa el BIM consigue abarcar todas las fases, lo cual no es equivocado ya que a menudo son técnicos diferentes los que se encargan de estas fases diferentes. Lo que si esperan es poder llevarlas a cabo dentro de la misma familia de software, es decir un paquete que incluye varios programas, los cuales al haber sido desarrollados específicamente uno para el otro, ofrecen la máxima compatibilidad y así se evita la perdida de datos en el momento de compartir archivos. Un ejemplo es el conjunto formado por Revit, Navisworks y BIM360, todos en este caso productos de Autodesk.

Por ultimo teniendo en cuenta que el proceso de creación de una obra arquitectónica se divide en varias fases, cabe mencionar que la gran mayoría de los programas que existen en este campo están pensados para la fase de diseño, probablemente porque es la más importante. Pero para fases más avanzadas o complementarias que enriquecen el proceso principal hay menos. Como puede ser el caso de herramientas de colaboración o revisión.

2. Análisis del mercado

2.1 Comparativa de la competencia

Actualmente los softwares BIM líderes que desarrollan la parte principal del proceso (modelado) y sirven de base a otros complementarios son Revit (Autodesk), ArchiCAD (Graphisoft) y Allplan (Nemetschek). Teniendo en cuenta que el proyecto que planteo consiste en una aplicación BIM complementaria a un software principal como los que he mencionado, voy a analizar las aplicaciones complementarias de las que dispone cada uno de estos. Además voy a incluir un par de aplicaciones independientes también relacionadas con tareas complementarias BIM pero especializadas en archivos 2D, y por último un par especializadas en visualizar archivos IFC (formato estándar en el campo AEC).

BIM 360 de Revit (Autodesk) (www.autodesk.com/bim-360/)
BIM 360 es un servicio complementario a los productos BIM de Autodesk, lo cual es casi como decir Revit (también existen otros como Navisworks). Se trata de una plataforma el objetivo de la cual es ofrecer herramientas de colaboración y gestión del proyecto. De hecho una de las características más importantes es el Revit Cloud Worksharing, el cual permite a todos los actores vinculados en el proyecto encontrarse en un punto.

Características destacadas:

- Acceso desde cualquier lugar (es una aplicación para dispositivos móviles). Utiliza el servicio Cloud de Autodesk para alojamiento y sincronización de proyectos.
- Almacena varios tipos de archivos del proyecto en carpetas: modelo, láminas, fotos…
- Revisión de documentos (se pueden añadir comentarios y anotaciones para notificar de errores).
- Tareas de colaboración y coordinación entre todos los participantes. Se especifica quien participa y en qué grado y tarea.
- Se puede ver el modelo 3D ver en qué parte está trabajando cada uno y que cambios realiza.
- Incluye un línea del tiempo en la cual se especifica cada equipo en qué fase actúa.

Coste:
A partir de $420/año por usuario.

BIMx de ArchiCAD (Graphisoft) (www.bimx.archicad.com/en/)
ArchiCAD por si solo dispone de una herramienta de colaboración llamada team work. Esta permite que varias personas trabajen sobre un mismo modelo simultáneamente. Este servicio no requiere ninguna aplicación externa pero utiliza el servicio BIM Cloud de Graphisoft.
La aplicación BIMx se trata de un complemento con el principal objetivo de poder visualizar el proyecto en dispositivos móviles, pensada tanto para los técnicos como para los clientes. Está disponible para

Android y para iOS, y desde hace poco también como una aplicación web para poder utilizarla desde el ordenador. BIMx también utiliza el servicio de BIM Cloud.

Características destacadas:

- Exportación integrada desde ArchiCAD, se eligen los documentos que se quieren subir a la aplicación (será el 3D más los layouts elegidos) y estos se cargan automáticamente.
- Posibilidad de navegar el 3D como si de un videojuego se tratara, incluso con realidad virtual lo cual es muy útil de cara a mostrar el proyecto a los clientes.
- La característica principal que lo diferencia es el hecho de disponer en el 3D de botones mediante los cuales se puede acceder a los layouts. También funciona al contrario, es decir desde un dibujo se puede acceder mediante un botón en el dibujo al 3D con una vista concreta. De este tipo de enlaces entre el modelo 3D y los layouts 2D sale el nombre de hyper-model que es el que se le da al modelo que se exporta.
- El modelo 3D se puede seccionar dinámicamente.
- En el modelo 3D se pueden seleccionar los elementos constructivos para ver su información. Esto no está disponible en los dibujos 2D.
- Medir (distancia, área y ángulo) haciendo snap. Esto no está disponible en los dibujos 2D.
- En los dibujos se pueden añadir anotaciones y comentarios como se haría sobre un PDF.
- Servicio para guardar y compartir modelos mediante un enlace (BIMx Model Transfer).
- Los documentos cargados se pueden actualizar individualmente con nuevas versiones, no es necesario exportar todo de nuevo.

BIMx Pro es solo para acceder a características de colaboración avanzadas, es decir especificar lo que hace cada uno y sincronizar con el modelo central. Además permite comunicarse con otras aplicaciones para mandar o recibir información.

Coste:
La versión estándar es gratuita.
La versión pro cuesta alrededor de $50.

BIMPLUS de Allplan (Nemetschek) (www.bimplus.net/es/)
Se trata de una aplicación web, por lo tanto accesible desde cualquier dispositivo, el principal objetivo de la cual es que las varias disciplinas implicadas en el proyecto pueden colaborar. Sirve para implementar en Allplan un Common Data Environment, es decir un centro común con todos los datos del proyecto BIM. Allplan Share es el servicio cloud que utiliza para compartir datos y almacenar.

Características destacadas:

- Visor 3D llamado BIM Explorer.
- Visualización de todos los modelos en los cuales trabaja cada equipo simultáneamente. Esto permite detectar errores entre las partes, por ejemplo con el control de colisiones.

- Coordinación de tareas entre equipos mediante el Task Board.
- Se pueden añadir archivos complementarios como imágenes.
- Se pueden seleccionar los objetos en el 3D y ver sus datos.
- Se pueden ocultar por categorías.
- Se pueden crear anotaciones sobre una vista del 3D en concreto. Se puede especificar a quien se asigna y la fecha límite entre otros aspectos.
- Visualización del 3D como realidad virtual.
- Posibilidad de incorporar datos desde cualquier otro software mediante el API.

Coste:

El precio depende del almacenamiento, va desde 2GB gratuitamente hasta 100GB por 70€ por usuario al mes.

También dispone de Addons, como la conexión con Excel. Cada uno de los módulos extra tiene un precio de 10€ por usuario al mes.

Revu de Bluebeam (www.bluebeam.com/solutions/revu)

Descubrir esta aplicación fue para mí una sorpresa ya que trabaja específicamente con documentación 2D. Se trata principalmente de una aplicación de escritorio con una versión para iPad con menos características. El hecho de que sea principalmente para ordenador no sorprende ya que no complementa a ningún otro gran software BIM.

Se puede entender como un visor de archivos PDF especialmente pensado para cualquier tipo de dibujos técnicos ya que contiene gran cantidad de herramientas para consultar, anotar en PDFs y gestionarlos.

Esta misma empresa produce también una aplicación llamada Bluebeam Drawings para dispositivos móviles pero es bastante simple y no tiene mucho éxito así que no la voy a tratar.

También se vende como herramienta de colaboración con otros agentes del proyecto con lo que llaman *Studio Projects* y *Studio Sessions*:

- *Studio Projects*: permite almacenar documentación de proyectos.
- *Studio Sessions*: permite a equipos colaborar en las tareas de revisión (comentar, modificar y actualizar los archivos).

Al trabajar con dibujos en formato PDF estándar y no estar vinculado a ninguna aplicación BIM directamente los dibujos no contienen información de los objetos que aparecen, sino que se trata de líneas y sombreados como un dibujo CAD convencional.

Coste:

De $349 a $599 según la versión.

Building Information Drawings de Thorton Tomasetti y Core Studio

El estudio de arquitectura Thorton Tomasetti de Nueva York dispone de un departamento de I+D llamando Core Studio que ha creado una aplicación interna que a partir de un modelo de Revit y usando el formato DXF genera unos dibujos con información asociada a la cual se accede seleccionando los elementos constructivos que aparecen en el dibujo (de ahí el concepto de Building Information Drawings). Esto fue requerido por el estudio de arquitectura ya que les resultaba realmente útil.
Toda la información que se de este proyecto ha llegado a mí a través de los programadores de la empresa en la cual trabajo ya que han colaborado a veces en talleres realizados por Core Studio.

Visores IFC genéricos: Tekla BIMsight y Solibri Model Viewer (Nemetschek)
El formato IFC es un formato de archivo estándar para el intercambio de información entre softwares BIM. Este incluye principalmente información geométrica y de los tipos de elementos constructivos. Es también el formato requerido para entregar proyectos BIM.
Debido a la difusión de este formato en el sector no es de extrañar que existan gran cantidad de visores de archivos IFC, dos de estos son BIMsight de Tekla y Solibri MV.
Estos están disponibles solo para ordenador y a veces disponen de versiones simplificadas para dispositivos móviles pero las cuales no tienen mucho éxito.

Características destacadas:

- Este tipo de software se centra en poder visualizar el modelo 3D y consultar a su información mediante la selección de sus elementos constructivos. No incluyen dibujos 2D ya que no forman parte del IFC.
- También ofrece la posibilidad de poder esconder los elementos por categorías las cuales se corresponden con las categorías IFC. Esto suele estar disponible en cualquier aplicación que disponga de visor 3D.

Coste:
Estas aplicaciones acostumbran a ser gratuitas.

Conclusion
Las aplicaciones analizadas son por lo general proyectos grandes desarrollados por grandes empresas y que incluyen muchas funciones avanzadas. Mi intención no es la de intentar copiar uno de esos productos ya que no sería un objetivo realista, sino más bien entender cuáles son las características más importantes que ofrecen y pensar cómo estas pueden ser aplicadas de manera sencilla en mi TFM para crear una aplicación que disponga de estas.

Las más importantes son:

- La **disponibilidad** en cualquier lugar es una característica muy importante ya que sin esta el resto no tendría sentido. En mi caso esto se conseguirá creando la aplicación como una página web con opción de instalación. Además tendrá que ser responsive y mostrar un formato

adecuado para poderla usar cómodamente tanto desde un Smartphone como desde una Tablet.

- La **visualización** del proyecto es un aspecto indispensable que en las aplicaciones directamente vinculadas a un producto BIM, como son las tres primeras que mencioné (BIM 360, BIMx y BIMPLUS), siempre está presente tanto en 3D como en 2D. En mi caso tengo la intención de centrarme más en la documentación 2D ya que creo que esta es especialmente importante para los implicados en la fase constructiva ya que tratándose de dibujos previamente preparados para mostrar una parte específica del edificio resultan más útiles técnicamente que el 3D. Además en las tres primeras aplicaciones comentadas los dibujos son tratados como simples PDFs sin funcionalidad añadida. Es por este motivo que mi intención es crear una documentación 2D que tenga funciones avanzadas más parecidas a las dos últimas aplicaciones analizadas. Estas características las mencionaré en el apartado de consulta y en el de revisión. La vista 3D también estará presente, ya que de cara a tener una idea general o a mostrar el proyecto al cliente es importante, pero no será desarrollada más que a nivel de prototipo.
- La **colaboración** es el aspecto más importante que tratan prácticamente todas las aplicaciones BIM. Es un tema muy importante ya que es muy normal trabajar en equipo y las herramientas de colaboración pueden hacer que el trabajo sea mucho más eficiente. Desgraciadamente son temas complejos de desarrollar ya que requieren la sincronización de varios usuarios sobre una base de datos centralizada a la cual puedan acceder todos para aportar su trabajo y poder ver lo que hacen los demás. Esta característica requiere de un buen servicio de back-end con base de datos y por eso cada una de las empresas mencionadas dispone de uno. En mi caso ya se de antemano que no voy a disponer de este servicio en el trabajo y por lo tanto como alternativa utilizaré el servicio de Google Drive mediante su API. Si después de la primera versión el resultado es bueno probablemente consiga convencer a la empresa de empezar a implementar un servicio de almacenamiento propio. El aspecto de la colaboración se puede implementar a varios niveles, lo importante es que esté presente. Mi intención es aplicarlo inicialmente a una escala pequeña para compartir tareas de revisión que comentaré más adelante y para compartir archivos.
- La **consulta** de datos del modelo es otro aspecto importante. Esta es más fácil de implementar que la colaboración y mi intención es que sea una de las características destacadas de mi TFM. Como he comentado en el apartado de visualización, los dibujos 2D tendrán funciones avanzadas y una de ellas será la consulta de información de los elementos que se encuentren en este, es decir se podrán seleccionar los objetos de los dibujos (ej: ventana) y se verá en una tabla todos sus datos como el área, el tipo… Otra función relacionada con la consulta será la posibilidad de medir distancias, áreas y ángulos. Ni la consulta de información ni las herramientas de medición estarán disponibles en el 3D. Por último habrá un apartado de consulta de datos del proyecto el cual no requerirá de ningún tipo de vista. Con este se podrán realizar cómputos como por ejemplo la superficie total de muros de un tipo.

- La **revisión** de los documentos de forma activa es el último aspecto importante que voy a tratar. La finalidad es la de poder indicar si hay errores y poder opinar sobre ellos. Todas las aplicaciones disponen de esta función, cada una en un nivel diferente. Lo más típico es que esta función se lleve a cabo como si de un trabajo manual se tratase, es decir sobre un dibujo poder añadir anotaciones. En mi caso quiero poder aprovechar la posibilidad de disponer de la información de cada objeto en los dibujos para poder realizar anotaciones específicas sobre un objeto o sobre más de uno, es decir etiquetándolos con un comentario. Además se podrán añadir comentarios no ligados a objetos concretos sino a partes del dibujo indicadas con una forma geométrica. A los comentarios se les podrá aplicar un nivel de prioridad.
  Para que esto funcione correctamente es importante que desde la aplicación web se pueda guardar la nueva información de los comentarios que se haya añadido. Es aquí donde entra en mi proyecto la necesidad de poder almacenar información generada por el usuario, lo cual se hará mediante el API de Google Drive. Además estará disponible la posibilidad de poder compartir archivos para ofrecer funciones de colaboración. La colaboración estará basada en la posibilidad de mandar el dibujo con las anotaciones y que otro usuario pueda modificarlas y/o contestarlas.

2.2 Modelo de negocio

Partiendo de que se trata de una aplicación que depende de archivos generados por un exportador concreto de un software de pago y que yo ya trabajo para la empresa que lo desarrolla la aplicación web por si sola será gratuita y servirá para añadir características al software base. No obstante habrá dos posibilidades de uso:

- Totalmente libre sin necesidad de crear una cuenta. Esta opción no permitirá almacenar proyectos, tampoco añadir información en ellos ni compartirlos, sino tan solo visualizarlos y consultar información cargando el archivo a la web.
- Por otra parte accediendo con una cuenta se podrán almacenar proyectos, añadir información como comentarios e imágenes y se podrá compartir mediante un enlace con otros usuarios. Además hacer uso de una cuenta permitirá que en el momento de exportar se pueda elegir si se quiere guardar el archivo directamente en la aplicación web (introduciendo las credenciales en el momento de exportar).

Teniendo en cuenta que como he mencionado este proyecto tiene relación con la empresa en la cual trabajo y que si la primera versión que desarrollo para el TFM les interesa puede que el proyecto prospere hay aspectos que he planteado de antemano para saber la cantidad de recursos disponibles. Es por eso que con la finalidad de ahorrar y hacerlo más cercano a la realidad he decidido aprovechar el API de Google para realizar la autenticación y el almacenamiento de los datos de los usuarios. Es decir el usuario accederá a la plataforma mediante su cuenta de Google y todos los proyectos que guarde y consuma se almacenaran en una carpeta de su Google Drive específica para la aplicación (teniendo en cuenta que cada usuario dispone de 15GB gratuitos). El usuario tendrá que consentir el acceso a Google Drive pero la gestión de los archivos la realizará totalmente la aplicación.

Una opción de pago no se contempla pero se podría valorar para acceder a opciones avanzadas como poder mandar información de comentarios de vuelta a VisualARQ y que estos apareciesen en el modelo.

3. Público objetivo y perfiles de usuario

El principal usuario es el usuario actual de Rhinoceros3D y VisualARQ, es decir mayoritariamente arquitectos e ingenieros. Además esta podría ser usada por otras personal relacionadas de alguna manera a este ámbito como otros técnicos o incluso clientes.

Pese a que la arquitectura a nivel mundial es un mercado muy grande, el objetivo de la aplicación sigue siendo muy específico, es por este motivo que probablemente en el futuro se añadirán nuevas funciones además de la consulta y la revisión para que resulte atractiva a mayor cantidad de usuarios.

4. Definición de las especificaciones del producto

- Disponer de una aplicación web mediante la cual un usuario pueda almacenar documentación de proyectos realizados con Rhinoceros3D + VisualARQ.
- Permitir visualizar los dibujos 2D (plantas, secciones…) que haya exportado del proyecto.
- Permitir obtener la información de los elementos que aparecen en los dibujos.
- Permitir realizar tareas de medición sobre los dibujos (cálculo de distancia, área y ángulo).
- Permitir añadir comentarios a los dibujos, ya sea marcando un elemento concreto o dibujando una zona.
- Permitir colaborar con otros usuarios en la revisión de un proyecto gracias a la posibilidad de permitir el acceso de todos los miembros del equipo a un mismo conjunto de documentos centralizado y a poderse etiquetar en los comentarios que se creen.

No todas las características mencionadas en la lista anterior han podido ser completadas. Es el caso de las herramientas de medición o la posibilidad de crear comentarios dibujando una zona. No obstante estas forman parte de la lista ya que considero que son imprescindibles para que la aplicación quede completa en su primera versión.

Capítulo 3: Diseño

1. Arquitectura general de la aplicación

Se trata de una aplicación sin back-end.

Esta es servida mediante el servicio Github pages. Esto lo consigo gracias a tener el proyecto en un repositorio público en Github, Esto me resulta conveniente ya que es gratuito y me permite personalizar el dominio con uno propio.

Como almacenamiento de archivos de los proyectos utilizo Google Drive al cual se conecta la aplicación mediante su API.

Google Drive resuelve además la autenticación de los usuarios.

Notificaciones mediante el servicio Firebase Cloud Messaging y una Firebase Cloud Function.

Base de datos mediante el servicio Firestore para almacenar datos relacionados con las notificaciones, concretamente el token que identifica cada dispositivo al cual se puede mandar una notificación.

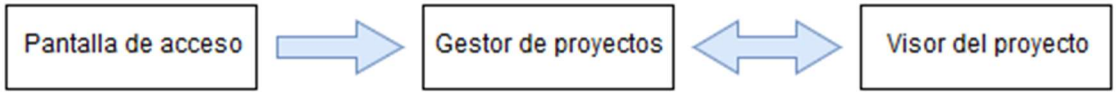
2. Arquitectura de la información y diagramas de navegación

2.1 Diagramas de navegación

La estructura de navegación de la aplicación a trazos generales consta de una pantalla con un listado de proyectos que el usuario habrá cargado y una pantalla de consulta del proyecto en la cual se encontrará el visor 2D con sus varias herramientas.

La simplicidad es fruto de tatar de imitar lo que sería un programa tradicional de un ordenador de escritorio. Es en la pantalla del visor del proyecto en la cual se encontrará la mayor parte de la lógica de la aplicación.

Las tres pantallas son por lo tanto las siguientes:



Pantalla de acceso	Gestor de proyectos	Visor del proyecto
Esta pantalla incluirá varias opciones. Las primeras serán las de autenticación del usuario. A continuación habrán otras como por ejemplo la de subir un proyecto para visualizarlo sin necesidad de acceder y por lo tanto sin opción de poder guardarlo. Otra será la posibilidad de consultar un proyecto de muestra.	Esta pantalla incluye principalmente el listado de proyectos del usuario. Además es desde esta pantalla desde donde se llevará a cabo la operación de carga de un nuevo proyecto a la aplicación. Se accederá desde aquí también a acciones generales de cada proyecto como eliminarlo, gestionar el equipo o cambiarle el nombre, las cuales también deberán estar disponibles también desde dentro del propio proyecto.	Esta es la pantalla más importante de la aplicación ya que es desde la cual se realizan las operaciones de consulta sobre los documentos de un proyecto. Incluye el *viewport*, la lista de dibujos del proyecto, la barra de herramientas y un cuadro en el cual se mostrará información si es requerido por la operación llevada a cabo.

Tabla 2: Tabla con la descripción de cada una de las pantallas principales de la aplicación

Teniendo en cuenta las tres posibles pantallas me surgió la duda inicialmente de como estructurar el proyecto, es decir de si crear tres páginas diferentes o si crear todo en una sola página.

Después de valorar los pros y los contras me decidí por desarrollar toda la aplicación en una sola página ya que para el usuario resulta más cómodo, especialmente se agiliza el paso del gestor de proyectos al visor, ya que es posible que mientras uno trabaja en un proyecto necesite acceder al gestor de proyectos varias veces. Así se evita tener que cerrar el proyecto para acceder a la lista de proyectos en otra página.

Además sería totalmente factible hacer uso de las flechas del navegador para volver a la lista de proyectos ya que es posible modificar la historia de la ventana.

Por último en el caso de querer ver varios proyectos a la vez usando varias pestañas del navegador simplemente habría que introducir la url específica de cada proyecto en una pestaña nueva.

2.2 Estructura HTML general

La estructura HTML se compone de una barra de navegación (`<header>`) y de un cuerpo (`<main>`). Además existen otros contenidos secundarios posicionados de manera flotante que aparecen solo cuando son requeridos como cuadros de dialogo emergentes.

2.3 El contenedor de los canvas `<svg>`

Dentro del elemento `<main>` es donde se encuentra el contenedor de los dibujos, el cual es uno de los elementos más importantes. La gestión de los dibujos SVG la he intentado implementar de la manera más sencilla posible. Esta se realiza de la siguiente manera:

El dibujo solicitado desde la lista de dibujos del visor se pide mediante el API, una vez este se recibe se guarda su contenido (SVG) en una variable que alberga varios datos del proyecto y finalmente el contenido del dibujo se añade al contenedor HTML. Cualquier otro dibujo que sea solicitado seguirá el mismo proceso y por lo tanto será primero guardado en la variable y posteriormente añadirlo al contenedor. Cuando en el contenedor ya hay más de un dibujo o cuando se solicite un dibujo que ya se solicitó antes, lo que se hará es ocultar el anterior y mostrar el actual mediante css.

2.4 Estructura de los archivos SVG que exportará Rhino + VisualARQ

El hecho de todavía no disponer del exportador me ha obligado a investigar y decidir cuál sería la mejor manera de estructurar el contenido de los dibujos SVG. Para ello he preparado casi manualmente con la ayuda de herramientas de dibujo vectorial tres dibujos que representan un proyecto arquitectónico. Este proceso me ha ayudado a concretar la estructura XML más óptima que han de tener estos y a adaptar la aplicación para que pueda interpretarla.

He agrupado los elementos SVG formando entidades (elementos independientes, por ejemplo un muro o una ventana) que corresponden a los que tiene el modelo BIM. Cada grupo (`<g>`) se identifica por disponer de un atributo que lo identifica como conjunto seleccionable el cual no es oficial y un atributo `data` con el id (no utilizo el atributo `id` ya que habría en el documento más de un elemento con el mismo id por el hecho de que un mismo elemento puede aparecer representado en más de un dibujo). Cada uno de los grupos alberga una serie de elementos también agrupados según su representación gráfica:

```
<g selectable data-id="column-7" data-category="columns">
   <g class="column-sectioned solid">
      <rect x="2041.154" y="969.479" width="70.866" height="70.866" />
   </g>
   <g class="column-sectioned curve">
      <rect x="2070.324" y="998.648" width="12.527" height="12.527" />
   </g>
</g>
```

La utilidad del atributo `selectable` es que me permite evitar tener que añadir un *click event listener* a cada grupo para detectar cuando el usuario quiere seleccionarlo. Lo que he hecho ha sido crear un único *event listener* al cargar la aplicación en el contenedor general de todos los dibujos (punto 2.3). Este detecta cualquier clic en un hijo suyo gracias a la propagación de eventos, una vez el evento llega al *event listener* comprueba si el target que lanzo el evento tiene algún ancestro con el atributo `selectable`, y en caso positivo ese será el elemento a seleccionar. Para mostrar la selección se utiliza

css y se guarda el `data-id` del elemento seleccionado. Si se hace clic fuera de cualquier elemento con ese atributo simplemente se deselecciona cualquier elemento que en ese momento lo esté.

El valor del `data-id` del elemento seleccionado guardado en la variable se utiliza para gestionar la selección al cambiar de dibujo. Concretamente permite deseleccionar el elemento en el dibujo actual si este está presente y añadir la selección al elemento en el nuevo dibujo si este está presente.

Los estilos css de los dibujos

El css que se aplica al `<svg>` de los dibujos y que se exporta con el archivo, se sitúa en un único lugar desde el cual todos los dibujos lo reciben y lo aplican mediante clases, de esta manera se reduce el código y se controlan los estilos css desde un solo punto. Este último aspecto es importante en el caso de que más adelante se permita modificar la representación de los elementos desde la aplicación para por ejemplo imprimir un pdf.

Esta serie de estilos css se cargan la primera vez que se solicita un dibujo, aunque la otra posibilidad sería al entrar en el espacio de trabajo. Actualmente se inyectan en el `<head>` en una etiqueta `<style>`. La intención era situarlos con la etiqueta `<style>` dentro de un contenedor específico del proyecto como por ejemplo el `<main>`, de tal manera que si más adelante se añade la posibilidad de tener más de un proyecto abierto a la vez estos al estar dentro de cada uno de los contenedores para cada proyecto y con el atributo `scope` permitiría apuntar solo a los dibujos de cada proyecto. Desafortunadamente el atributo `scope` se eliminó y si realmente más adelante permito abrir más de un proyecto a la vez tendré que buscar una alternativa.

Ahora al cerrarse un *workspace* se elimina la etiqueta `<style>` del `<head>` de tal manera que al abrir otro no haya más de una simultáneamente.

Representación gráfica de la selección

La representación gráfica de la selección se realiza mediante la adición de una clase css. Esta clase en vez de situarse una sola vez en el grupo que engloba todo el elemento, se sitúa en cada uno de sus hijos, los cuales corresponden a grupos de elementos con representación gráfica similar. Es en ese mismo nivel donde se encuentran las clases que dan el estilo a los dibujos.

Pese a tener todas las clases el mismo nivel de especificidad, debido a que el estilo de los dibujos se añade dinámicamente (cuando se solicita el primer dibujo) en un `<style>` al final de `<head>` este sobrescribe la hoja de estilos general que incluye el css para indicar selección y esta deja de tener efecto. Para evitar este problema y no tener que preocuparme por situar el `<style>` de los dibujos delante del `<link>` css lo que he hecho ha sido añadir mayor especificidad a las reglas de selección añadiendo otro nivel que indica los posibles elementos svg a los que afecta en su interior, por ejemplo:

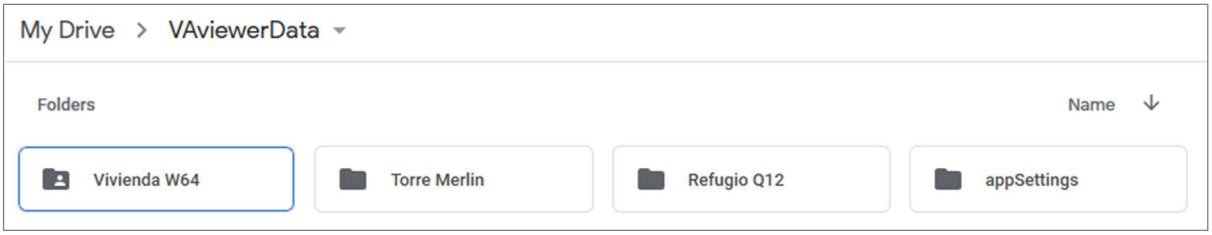
```
.selected line
```

2.5 Arquitectura de los archivos almacenados en Google Drive

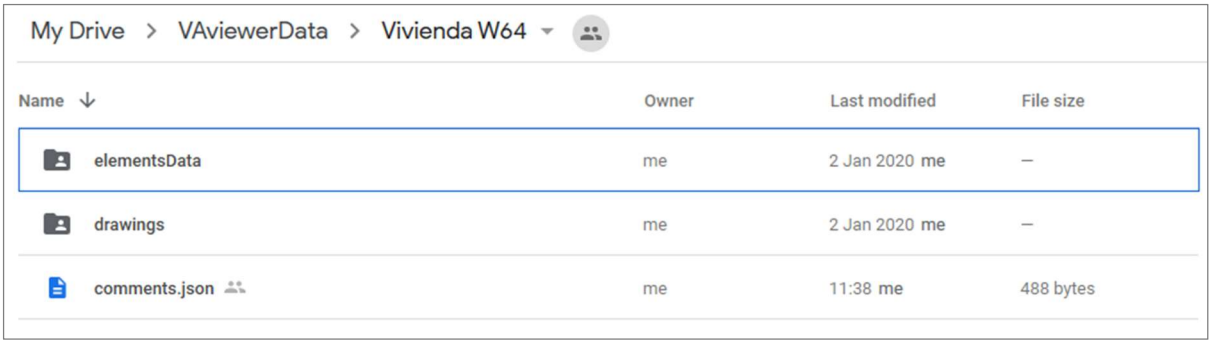
Todos los archivos que gestiona la aplicación los almaceno en una carpeta que se crea la primera vez que se guarda un proyecto, actualmente tiene el nombre de VAviewerData. Por ahora esto lo he hecho con una carpeta estándar visible por el usuario en vez de con una carpeta oculta específica para aplicaciones. He decidido hacerlo así por comodidad mientras trabajaba y también para que el usuario pueda gestionar los contenidos de sus proyectos también accediendo a su Google Drive directamente.

Para identificar esta carpeta actualmente uso su nombre, no obstante tendré que buscar una alternativa ya que si existiese otra carpeta con ese nombre o si se le cambiase el nombre la aplicación dejaría de funcionar correctamente. Una opción sería almacenar el id de esa carpeta visible por el usuario en una carpeta oculta para la aplicación o en una base de datos.

Dentro de la carpeta de la aplicación (VAviewerData) se encuentran las carpetas de los proyectos además de una carpeta llamada *appSettings*. Dentro de *appSettings* habrán datos de la aplicación como por ejemplo las miniaturas de cada proyecto y un archivo JSON con ajustes generales que se creará la primera vez que este se necesite.



Cada carpeta de proyecto incluirá una carpeta *drawings* para los archivos SVG de los dibujos y una carpeta *elementsData* para los JSON con datos de los elementos del dibujo. Además podrá haber si es necesario un archivo para ajustes específicos del proyecto y uno para almacenar los comentarios ambos en formato JSON.



Los archivos JSON relativos a ajustes de la aplicación que he mencionado se podrían guardar en una carpeta oculta para la aplicación ya que el usuario no necesita para nada acceder a ellos directamente y podría borrarlos accidentalmente.

El contenido de los archivos svg correspondientes a los dibujos no se modificara nunca. Por ejemplo en el caso de añadir las representaciones de los comentarios lo que se hará es guardar un archivo JSON con la información de los comentarios y estos se pintaran cada vez que el proyecto abra.

2.6 Arquitectura del archivo de proyecto que será exportado por Rhino + VisualARQ

Este será el archivo que generará el exportador cuando sea desarrollado y que será interpretado por el formulario de la aplicación. La aplicación lo que hace actualmente es leer cada una de sus partes y enviarlas a Google Drive para crear un nuevo proyecto.

El archivo tiene estructura JSON y está compuesto por tres entradas. La primera “projectInfo” que contiene un objeto donde actualmente se encuentra solo el título del proyecto aunque más adelante incluirá otros datos como el propietario. Las otras dos entradas que incluye el archivo son “drawings” y “elementsData”, cada una de las cuales tiene asociada un objeto que contiene los dibujos y los datos de los elementos respectivamente.

En el caso de los datos de los elementos se utiliza el nombre de la categoría como key, por ejemplo *columns* o *windows*. En el caso de los dibujos se utiliza el nombre de cada dibujo. Si el hecho de usar el nombre del dibujo como key se volviese un inconveniente ya que dos dibujos tienen el mismo nombre entonces tendré que modificar la estructura en forma de array de objetos, cada uno de los cuales contendrá el nombre del dibujo y su contenido. De momento no lo he hecho así para simplificar el proceso de lectura.

Un aspecto que cabe mencionar es que para poder almacenar correctamente el contenido SVG como un *string* es necesario que este esté *url-escaped*, lo cual implica por ejemplo que haya que escapar las comillas (\").

Ejemplo de la estructura del archivo:


```
{
  "projectInfo": {
    "title": "Villa Savoye"
  },
  "drawings": {
    "drawing 1": "<svg data-name=\"drawing 1\" xmlns=\"http://www.w3.org/2000/svg\" xmlns:xlink=\"http://www.w3.org/1999/xlink\"
    viewBox=\"0 0 500.00 500.00\"><g selectable=\"\" data-category=\"columns\" data-id=\"column-1\"><rect x=\"21\" y=\"129.7\"
    fill=\"#B1FFE2\" stroke=\"#000\" width=\"131.167\" height=\"64.7\"/><text transform=\"matrix(1 0 0 1 26.00 150.00)\"><tspan
    style=\" font-family:sans-serif; font-size:12pt;\" fill=\"#000\">Column 1</tspan></text></g><g selectable=\"\"
    data-category=\"columns\" data-id=\"column-3\"><rect x=\"103.583\" y=\"407.75\" fill=\"#B1FFE2\" stroke=\"#000\"
    width=\"131.167\" height=\"64.7\"/></g></svg>",
    "drawing 2": "<svg data-name=\"drawing 1\" xmlns=\"http://www.w3.org/2000/svg\" xmlns:xlink=\"http://www.w3.org/1999/xlink\"
    viewBox=\"0 0 500.00 500.00\"><g selectable=\"\" data-category=\"columns\" data-id=\"column-1\"><rect x=\"348\" y=\"394.7\"
    fill=\"#B1FFE2\" stroke=\"#000\" width=\"131.17\" height=\"64.7\"/></g><g selectable=\"\" data-category=\"columns\"
    data-id=\"column-2\"><rect x=\"22.5833\" y=\"220.75\" fill=\"#B1FFE2\" stroke=\"#000\" width=\"131.167\" height=\"64.7\"/><text
    transform=\"matrix(1 0 0 1 28.00 241.00)\"><tspan style=\" font-family:sans-serif; font-size:12pt;\" fill=\"#000\"
    stroke-width=\"0.2\">Column 2</tspan></text></g></svg>"
  },
  "elementsData": {
    "columns": {
      "instances": {
        "column-1": {
          "content": "Information for the column 1"
        },
        "column-3": {
          "content": "Information for the column 3"
        }
      }
    },
    "walls": {
      "instances": {
        "wall-1": {
          "content": "Information for the wall 1"
        },
        "wall-2": {
          "content": "Information for the wall 2"
        }
      }
    }
  }
}
```


Figura 3: Captura de pantalla del contenido de un archivo que imita al que exportará Rhino + VisualARQ

2.7 Arquitectura de clases

A continuación se van a explicar algunas de las clases presentes en la aplicación elegidas por su relevancia. Para consultar todas las clases que forman la estructura de la aplicación se puede acceder directamente al repositorio del proyecto mediante este enlace:

https://github.com/juanramoncarceles/visualarq-drawings-viewer

Clase Application

Esta es la clase padre que es instanciada una vez y la cual contiene el método `start()` que se ejecuta al arrancar la aplicación.

En la mayoría de casos es la encargad de gestionar a las demás, por ejemplo gestiona la creación de espacios de trabajo (workspace) que veremos a continuación.

Clase Workspace

Esta corresponde al espacio de trabajo de un proyecto. Se crea una instancia de esta clase cuando se accede a un proyecto, ya sea desde la lista de proyectos como directamente con su url. Tiene la función de preparar el espacio de trabajo por ejemplo creando la lista de dibujos disponibles, de gestionar las herramientas activas, de indicar si hay cambios por guardar y del cambio de dibujos entre otras tareas. Actualmente el usuario no puede crear más de un *workspace* a la vez.

Un aspecto interesante es como gestiona mediante el método `setDrawing()` los dibujos. Cualquier cambio que se realice sobre un dibujo, ya sea seleccionar un elemento o crear un comentario solo repercutirá sobre ese dibujo y no será hasta el momento de cambiar de dibujo que se decidirá lo que debe pasar con el nuevo dibujo. Esto se ha hecho así para optimizar y porque los dibujos se cargan desde Google Drive a medida que se solicitan, es decir que lo más probable es que nunca estén todos disponibles en el navegador y por lo tanto es mejor tratarlos justo antes de mostrarlos.

Clase Drawing

Es una clase que tiene una referencia al elemento svg correspondiente. Cuando se accede a un proyecto (cuando se crea un *workspace*) es cuando se crean tantas instancias de esta clase como dibujos tenga el proyecto. Los objetos dibujo inicialmente se crean solo con el nombre e id del dibujo, es decir sin contenido gráfico, y se añaden a una propiedad del objeto Application que guarda información de todos los proyectos. Es en el momento que se hace clic en un dibujo de la lista, cuando se añade el contenido gráfico al objeto dibujo (en el caso de que aún no lo tenga) mediante el método `setContent()`.

Para optimizar aún más su gestión por parte del *workspace* disponen de propiedades booleanas como por ejemplo *commentsChanged*.

Clase Tool

He creado clases para las herramientas de la aplicación que he creado hasta ahora. Lo primero que he hecho para implementar las herramientas ha sido crear una clase abstracta base de la cual heredaran todas las herramientas. Esta clase contiene información del nombre de la herramienta y del botón desde la cual se activa. Luego he creado otra clase abstracta (ElementSelection) de la cual heredaran todas aquellas herramientas que requieran seleccionar elementos (si JavaScript soportase la herencia múltiple no hubiese hecho que ElementSelection herede de Tool, sino que la herramienta herede directamente de ambas). Es a partir de la clase abstracta ElementSelection que por lo tanto he implementado las dos clases correspondientes a las herramientas que hay actualmente:

- ElementData
- AddComment

Clase Comment

Cuando se crea un comentario se crea una nueva instancia de la clase Comment con los datos del comentario y este se añade al *array* de comentarios, que es una propiedad del *workspace* actual. Además mediante un método de ese nuevo objeto comentario se crea su primera representación gráfica, la cual corresponde a la del dibujo actual. Las representaciones gráficas se irán colocando en un elemento `<g>` al final del `<svg>` correspondiente y al mismo tiempo se guardará una referencia en un *array* de representaciones que tiene el objeto comentario.

Los datos que se guardaran en Drive sobre cada objeto comentario serán los siguientes:

```
{ elementId: "45763453", content: "Hola", mentions: ["anyone@mail.com"] }
```

Estos son los mismos que son necesarios para crear una instancia de la clase Comment.

Clase MainPanel

El panel principal es un elemento muy importante de la aplicación. Este es una ventana que aparece al lado o encima de los dibujos y que muestra contenido relacionado con la acción que se está llevando a cabo. Por ejemplo si se quieren ver las propiedades de un elemento, al seleccionarlo estas aparecerán en el panel, o si se comenta un elemento, el formulario para introducir el texto aparecerá también en el panel.

Esta clase permite controlar todo lo relacionado con este panel. Tiene métodos para añadir y quitar contenido así como para anclarlo en diferentes sitios (este último no se puede utilizar todavía).

Un aspecto importante es que está preparado para mostrar varios contenidos de categorías diferentes simultáneamente los cuales se organizan en pestañas. Esto se puede comprobar por ejemplo si se selecciona un elemento que tiene un comentario ya que en el panel aparecerán dos secciones, una para las propiedades del objeto y otra para el comentario asociado.

Además de crear clases también he creado algunos métodos globales, añadidos explícitamente al objeto *window* para que sean accesibles desde cualquier sitio. Esto lo he hecho ya que hay datos generales que necesito obtener desde varios puntos del código a menudo como es el caso de obtener la información del usuario actualmente autenticado.

3. Diseño gráfico e interfaces

3.1 Estilos

En general he intentado mantener el diseño lo más simple posible y he evitado el uso de elementos redondeados.

Logotipo

El logotipo se inspire claramente en el del producto principal que es VisualARQ. Partiendo de su logotipo y teniendo en cuenta que la característica principal de la aplicación es la de revisar proyectos lo que he hecho ha sido simbolizarlo con una lupa sobre el logotipo original, haciendo zoom sobre la parte correspondiente.

Figura 4: A la izquierda el logo de VisualARQ Drawings Viewer y a la derecha el de VisualARQ

Paleta de colores

El utilizado el menor número de colores posible. Los que más destacan son los azules, que siguen la línea del logotipo, y el naranja que utilizo en casos puntuales para llamar la atención, por ejemplo cuando se indica que hay que guardar cambios o que se ha recibido unan notificación. La mayoría de elementos son de color gris o blanco.

Fuentes

La fuente utilizada es Oswald de Google Fonts la cual es muy similar a la utilizada en el nombre de VisualARQ.

3.2 Bocetos y wireframes

A continuación muestro todo el material gráfico realizado en las primeras fases de diseño del producto.

Dibujos

En las primeras fases del proceso cree tres dibujos a mano para representar las pantallas principales.

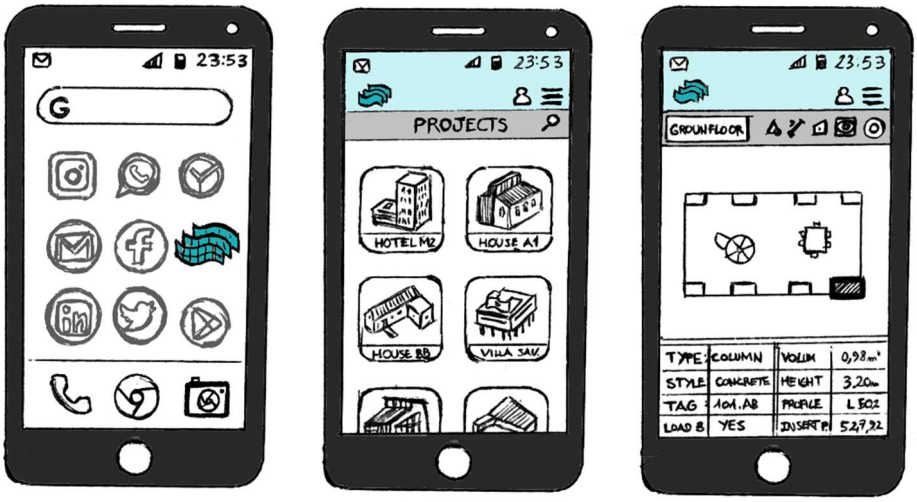


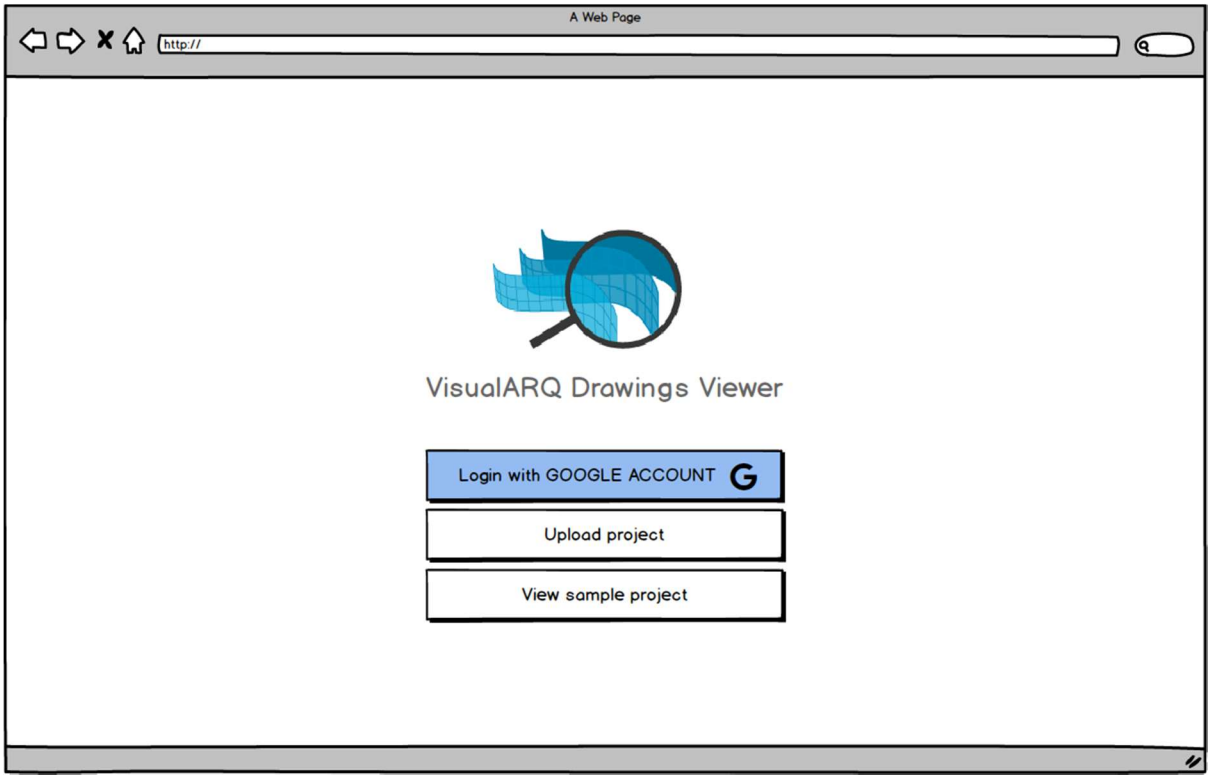
Figura 5: Dibujos iniciales a mano alzada de las pantallas principales de la aplicación

Wireframes

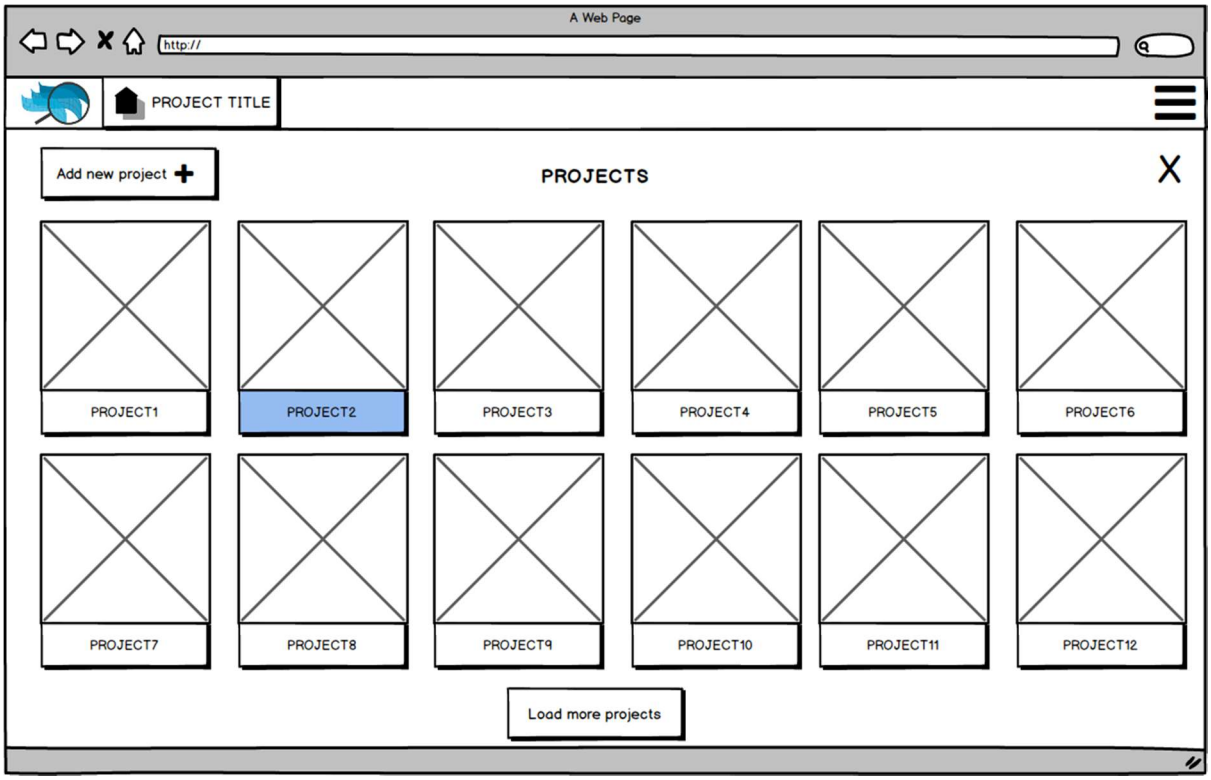
Para realizar los wireframes utilicé la herramienta Balsamiq en su versión de prueba.

Estos corresponden a cada una de las tres pantallas principales para la versión para ordenador y para teléfono. El mayor cambio en la versión para Smartphone (menos de 500px) es que la barra de navegación se sitúa debajo y el contenedor del viewport arriba. Esto lo he hecho así con el objetivo de acercar los botones a la parte inferior, cercano al dedo pulgar cuando el teléfono se coge con una sola mano.

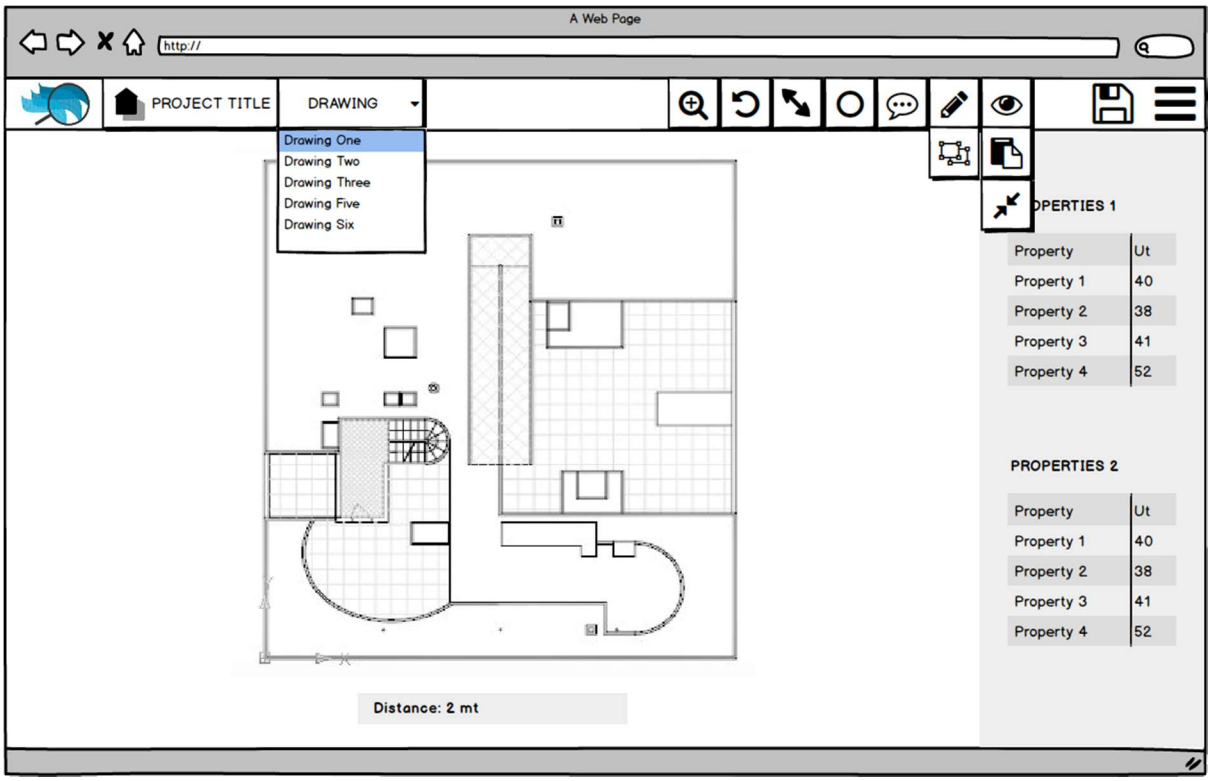
Pantalla de acceso - Desktop



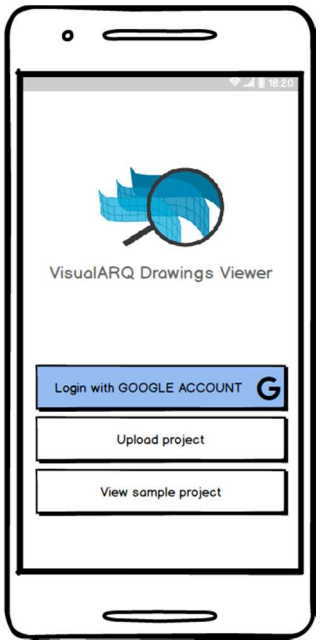
Pantalla lista de proyectos - Desktop



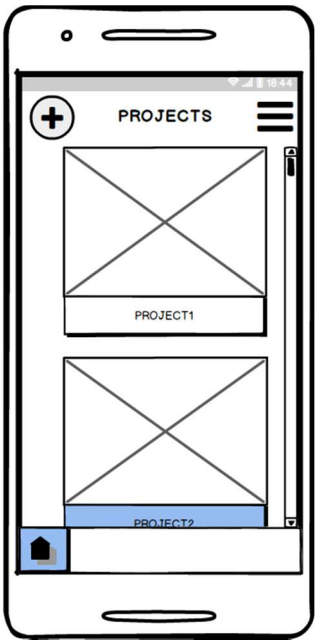
Pantalla Visor proyecto - Desktop



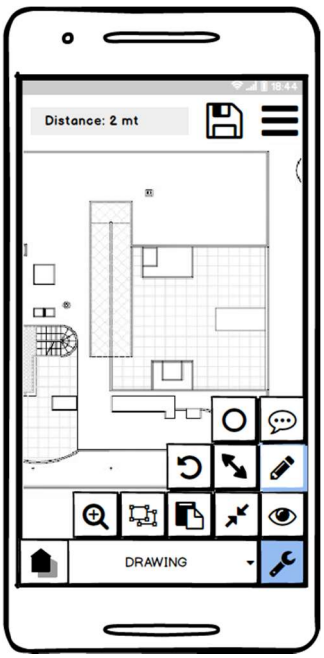
Pantalla acceso - Smartphone



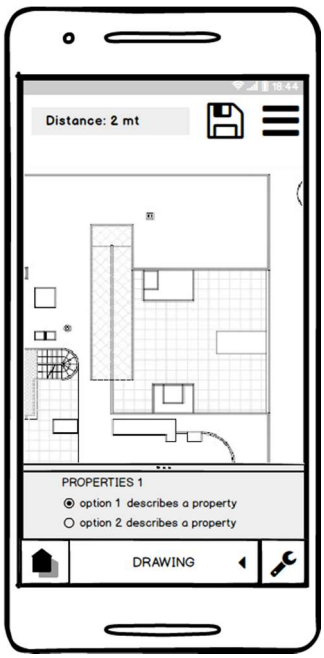
Lista de proyectos - Smartphone



Visor de proyecto (herramientas) - Smartphone



Visor de proyecto (tabla datos) - Smartphone



4. Lenguajes de programación y APIs utilizadas

4.1 Software

He utilizado Rhinoceros3D con VisualARQ para crear el modelo BIM de ejemplo.
He utilizado Expression Design para modificar el svg exportado de Rhinoceros3D.
He utilizado Visual Studio Code para todo el código.
Otras herramientas para la documentación han sido Word y Excel.

4.2 JavaScript

Como lenguaje de programación he utilizado JavaScript. Debido a la fragmentación del código en varios archivos para organizarlo por clases he tenido que utilizar un bundler ya que los navegadores todavía no soportan la sintaxis import/export, que es la que permite fraccionar el código. Como bundler he elegido Webpack ya que lo he usado en otras ocasiones y lo he configurado por ahora para que genere un único archivo main.js que es el que vinculo al final del archivo index.html. Por ahora he mantenido la configuración bastante básica, más adelante si fuese necesario añadiré complementos para por ejemplo hacer el JavaScript compatible con versiones de navegadores anteriores, para minificar el código…
Inicialmente propuse utilizar Typescript aunque lo descarté para acelerar el proceso. Actualmente me doy cuenta que sería una ventaja haberlo utilizado desde el principio y que lo tendré que añadir tarde o temprano.

4.3 API Google Drive

Debido a el requisito de no tener que mantener un back-end busqué un servicio alternativo. Me decidí por Google Drive ante otros como Dropbox ya que Google me resultaba más cómodo y es más conocido.
Utilizar el API de Google Drive ha resultado muy útil ya que ha resuelto a la vez el aspecto de la autenticación de los usuarios. Además la librería que proporcionan simplifica mucho cualquier tarea relacionada con el API.
Para aprender a utilizarla he consultado la documentación oficial, la cual incluye varios ejemplos y he consultado repetidamente Stackoverflow cuando esta no era suficientemente clara. Un aspecto interesante es que las carpetas se consideran archivos también, solo que con un formato propio.

Permisos

Uno de los aspectos incomodos para el usuario de usar Google Drive es el hecho de que la primera vez que accede a la aplicación con su cuenta de Google se le solicitara que de varios permisos a la aplicación para que esta puede leer, modificar y editar archivos de su drive. Esto podría echar atrás a los usuarios. Lo mejor será pedir permisos de forma progresiva según el usuario lo necesite y de esta manera entenderá porque se le está pidiendo eso.

Funciones del API implementadas

Las operaciones básicas que he implementado durante esta etapa han sido:
Listar archivos:

A partir de una query con varios parámetros esta retorna un array de objetos con los datos solicitados, por ejemplo nombre e id.

Obtener contenido de un archivo (de tipo texto):

A partir de un id devuelve el contenido, este lo utilizo para archivos JSON y SVG los cuales llegan como un string.

Obtener datos de un archivo específico:

A partir de su id, se reciben los campos que se necesiten de los que pone a disposición el API. Son datos del archivo pero no su contenido.

Creación de una carpeta:

A partir de su nombre y padre. Devuelve su id.

Carga de un archivo:

Esta ha sido una de las más difíciles ya que el ejemplo de la documentación no es fácil ya que incluye tres opciones según el tamaño del archivo. Finalmente encontré en Internet un ejemplo con el método para archivos pequeños, de hasta 5MB. Más adelante (probablemente después del TFM) tendré que implementar otro método que permita más capacidad, sobre todo cuando haya una función para cargar imágenes del proyecto.

Eliminar un archivo o carpeta:

Eliminar de momento lo he utilizado para eliminar proyectos, lo cual se puede llevar a cabo desde el menú contextual que aparece sobre los ítems proyecto de la lista.

Un aspecto importante era reflejar los cambios en el front-end tras haber recibido la respuesta del servidor de que la carpeta del proyecto se eliminó correctamente.

No obstante la eliminación de un proyecto no está del todo implementada ya que tendré que ampliarla para que haga más cosas como borrar su miniatura en el caso de que disponga de una, ya que estas se encuentran en una carpeta separada. También falta gestionar las variables currentProject y lastUploadedProject en el caso de que el proyecto esté en estas. Además si se quisiese borrar el proyecto actual que está abierto primero habría que poder cerrarlo y poner la aplicación en un estado equivalente al inicial.

Actualizar un archivo:

Actualizar el contenido de un archivo lo he utilizado para guardar los comentarios añadidos ya que por ahora es el único contenido que se puede añadir sobre los dibujos.

Renombrar un archivo:

El método para renombrar lo he implementado pero todavía no lo he utilizado en ningún sitio. Lo más probable es que haga una prueba para cambiar el nombre a un proyecto.

A partir de las funciones básicas mencionadas he creado una serie de funciones propias que incluyen varias de estas para realizar tareas más complejas específicas de la aplicación. Estas funciones las he tenido que crear también como asíncronas ya que siempre alguna parte de la aplicación las ha de esperar para dar una respuesta al usuario. Las principales funciones son:

Creación de un proyecto:

A partir de un archivo subido a la aplicación se realizara primero una comprobaciones de su validez y si existe ya un proyecto con ese nombre, si el proceso es exitoso se procede fragmentando cada una de sus partes (principalmente dibujos SVG y datos de elementos JSON) y se envía a Drive. Todos los datos se guardan formando una estructura concreta de carpetas. Esta función además almacena los contenidos del proyecto n el objeto global lastUploadedProject. Esto evita que se tengan que ir a buscar los contenidos al back-end si se acaban de subir. El hecho de guardarlos en el objeto permite también poderlos consultar sin conexión a internet.

Listar proyectos:

Toma los nombres e id de los proyectos, con estos datos se crea la lista de proyectos en HTML.

Coger proyecto:

A partir del id del proyecto devuelve los nombres e id de los archivos de este, dibujos SVG y JSON con datos de elementos

4.4 Firebase

He utilizado tres servicios de Firebase para resolver la falta de back-end ante la necesidad de algunas funciones avanzadas que lo requerían como por ejemplo la posibilidad de recibir notificaciones push o enviar emails.

Lo elegí porque fue el que me pareció más completo y al estar vinculado a Google más fácil de implementar en el proyecto actual.

Los servicios de Firebase que he utilizado son:

Firebase Cloud functions

Este lo utilizo para enviar emails cuando se añade a un nuevo miembro a un equipo ya que no tenía la posibilidad de hacerlo de un navegador.

La otra situación es para poder enviar notificaciones push, para lo cual necesitaba una función que se ejecutase y que mediante el servicio Firebase Cloud Messaging enviase una notificación a un usuario cuando este fuese etiquetado en un comentario.

Firebase Cloud Messaging

Este servicio ya lo he mencionado en el punto anterior ya que en mi caso funcionan conjuntamente. Es el encargado de enviar notificaciones. El envío se ejecuta mediante la función anterior.

Firestore

Ha sido necesario como base de datos para almacenar los tokens que genera el servicio anterior y que identifican a cada dispositivo que utiliza la aplicación y que ha aceptado poder recibir notificaciones. Estos tokens los lee Firebase Cloud Messaging para saber cómo hacer llegar el mensaje al dispositivo.

Capítulo 4: Demostración

1. Instrucciones de uso

Enlace para acceder a la aplicación:

**Es importante acceder con https, sino la aplicación podría no funcionar.*

https://visualarqapp.ramoncarceles.com

Repositorio:

https://github.com/juanramoncarceles/visualarq-drawings-viewer

Credenciales de acceso a la aplicación

Se recomienda acceder a la aplicación con las siguientes cuentas de Gmail creadas a propósito para realizar pruebas. Si se accede con la propia nos encontraremos con un aviso de peligro de Google de que se está accediendo a una aplicación que todavía no ha sido validada.

Cuentas de Gmail para realizar pruebas:

	Usuario 1	Usuario 2
Dirección	visualarqdrawingsviewer@gmail.com	lauragonzales01010101@gmail.com
Contraseña	VisualARQ0.	LauraGonzales0.

Se recomienda realizar las pruebas en el navegador Chrome y si no fuese posible en Firefox.

Acceder desde una ventana de incognito es una buena idea para no tener que cerrar sesión de la cuenta que tengamos actualmente en Chrome, no obstante hacer uso de una ventana de incognito no permitirá poder probar los pasos de instalación de la aplicación ni de recibir notificaciones push. Si se quieren probar habrá que acceder sin modo incognito.

**Todos los proyectos arquitectónicos presentes en la aplicación tienen el mismo contenido y solo cambia su nombre y su imagen.*

Pasos

1. Acceso

Acceder mediante la cuenta del Usuario 1 proporcionado.

Tras acceder se deberá ver una lista de proyectos disponibles.

2. Borrar un proyecto

Borrar uno de los proyectos mediante el menú contextual sobre un proyecto con la opción Delete.

3. Invitación de un nuevo miembro

Nótese que algunos proyectos tienen un icono en la parte superior derecha que indica que se está colaborando.

Hacer clic secundario sobre un proyecto y seleccionar la opción Share Project (es indiferente si en ese proyecto ya se está colaborando o no).

Aparecerá el cuadro de dialogo de gestión del equipo.

Introducir la dirección de email del Usuario 2 proporcionado con la finalidad de invitarlo.

Dejar marcada la opción *Notify new members* de la esquina inferior izquierda.

Una vez aceptemos podemos comprobar si queremos como ese miembro habrá recibido un email de invitación. Conservar el email si se quieren comprobar las notificaciones más adelante.

Ahora si vuelvo a abrir el cuadro de dialogo de colaboración veré que el Usuario 2 aparece en la lista de miembros de ese proyecto.

Si lo deseo desde esa ventana también puedo eliminar a un miembro.

Una vez haya finalizado añadiendo al Usuario 2 a un proyecto puedo proceder.

4. Acceso a un proyecto

Acceder a un proyecto en el cual hayamos añadido al Usuario 2.

Una vez dentro consultar los dibujos que se dese de la lista de dibujos situada en la esquina superior izquierda en la versión desktop o en la barra inferior en Smartphone.

5. Herramienta “información de los elementos”

Con por lo menos un dibujo cargado ejecutar la herramienta “información de los elementos”.

Icono: 

Esta herramienta permitirá seleccionar elementos del dibujo y ver su información en una tabla de datos que aparecerá en el panel lateral. La información mostrada será muy similar en todos los elementos ya que es de prueba, observar el campo “nombre” que es el único que será diferente para cada elemento. Esta información correspondería a la exportada por el software BIM.

Cambiar de dibujo mientras un elemento esté seleccionado para ver cómo se mantiene la selección de este.

6. Herramienta “añadir comentario a elemento”

Cambiar a la herramienta “añadir comentario”.

Icono: 

Tras activarla seleccionar el elemento del dibujo que se desee comentar. Los comentarios se representan con una línea roja discontinua. En el caso de que ya hubiese elementos con comentarios seleccionar uno que no tenga.

Tras haber seleccionado un elemento aparecerá el panel lateral con un formulario para introducir el texto. Además tendremos la posibilidad de etiquetar a un miembro del equipo. En este caso nos aparecerá el usuario que añadimos previamente.

Si se desea comprobar el funcionamiento de las notificaciones push habrá que acceder previamente con el otro usuario a la aplicación mediante el enlace situado en el correo de invitación que recibió. Es aconsejable por lo menos abrir dos navegadores diferentes si no se puede probar desde dos dispositivos diferentes. Esto no funcionará si se abre la aplicación desde una ventana de incognito.

Tras enviar el formulario se cerrará el panel y una representación gráfica se añadirá alrededor del elemento indicando que ahora tiene un comentario. También el usuario mencionado recibirá una notificación.

Cambiar de dibujo para comprobar como la representación gráfica del comentario se ha añadido en todas las vistas de este mismo elemento.

7. Guardar los comentarios añadidos

Guardar los comentarios añadidos mediante el botón de guardar situado en la esquina superior derecha. Para comprobar que se han guardado correctamente se puede cerrar completamente la aplicación y posteriormente volver a entrar al mismo proyecto. O desde el otro usuario con el cual se comparte el proyecto cargar la página para que le aparezca.

8. Editar y borrar comentarios

Si se desea editar o borrar un comentario se deberá activar de nuevo la herramienta “información de los elementos” y seleccionar el elemento comentado.

En el panel lateral aparecerá además de las propiedades del propio elemento el contenido del comentario y tendremos a disposición los botones editar y borrar.

9. Instalación

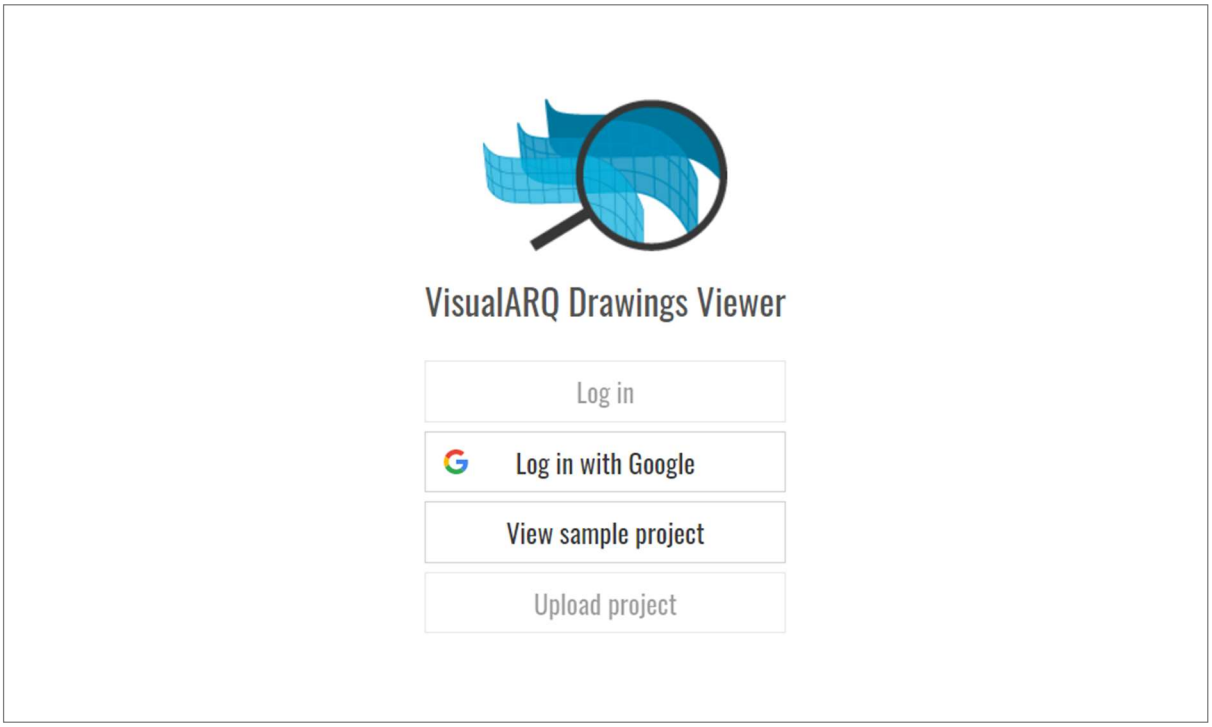
La instalación se podrá realizar mediante el botón que se encuentra al pie del menú lateral al cual se acceder desde la esquina superior derecha. Este botón solo aparecerá si el navegador cumple los requisitos y si no se hace uso de una ventana de incognito.

Para ver una demostración de estos pasos consultar el video de la presentación académica de este proyecto.

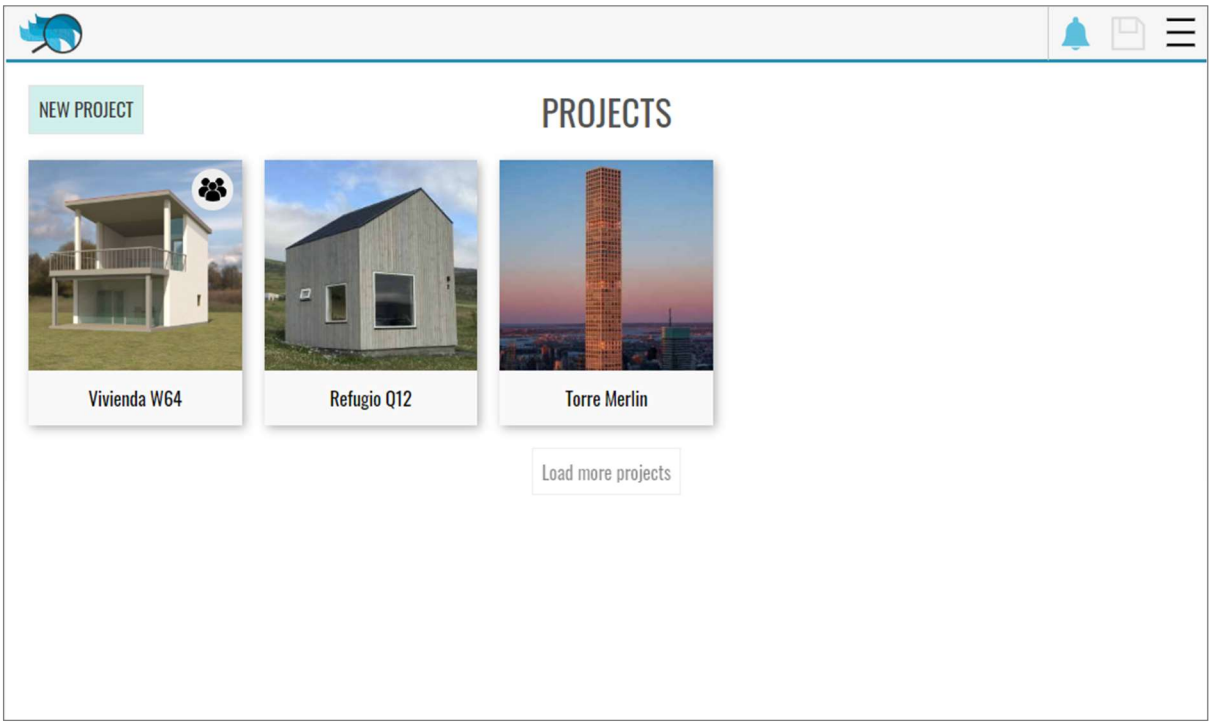
2. Pantallas

Capturas de pantalla de la aplicación final.

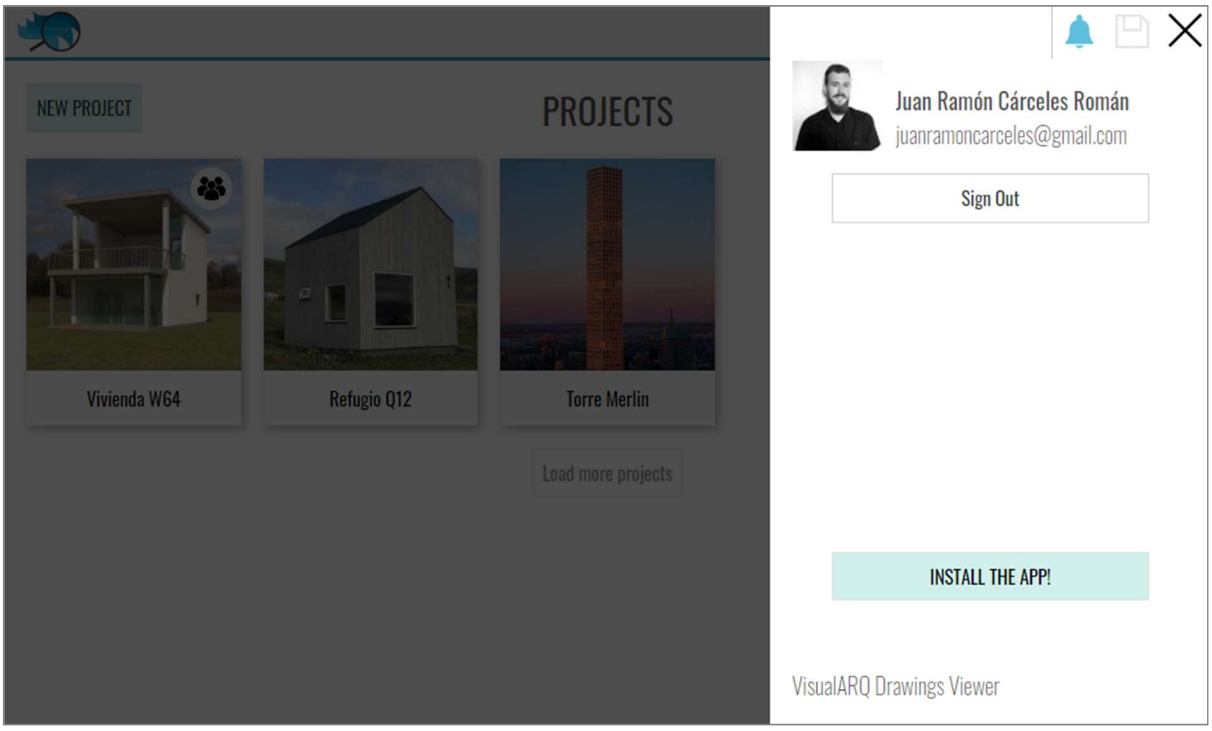
Pantalla de acceso



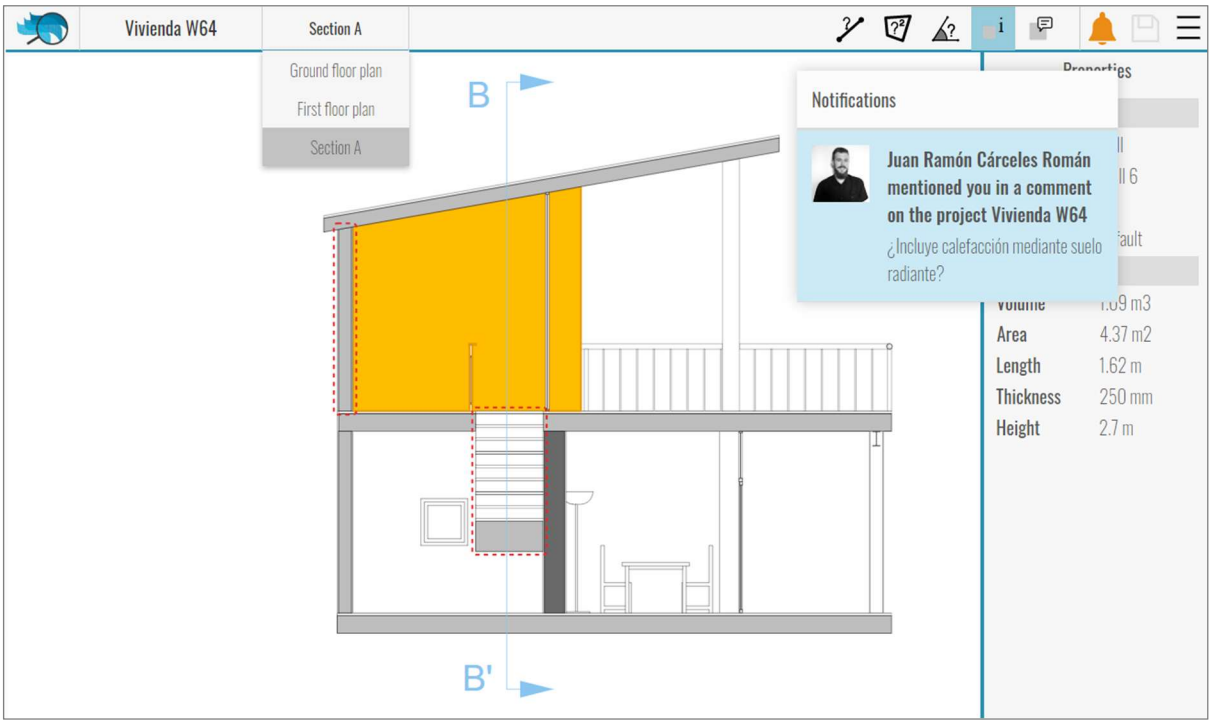
Listado de proyectos



Pantalla con el menú lateral abierto



Visor del proyecto con el panel de datos lateral y la bandeja de notificaciones abierta



3. Tests

He tenido la suerte de poder hacer probar la aplicación a varias personas de mi entorno a medida que la desarrollaba y en especial en la fase final. El perfil de estos usuarios en muchos casos coincidía con el de los usuarios objetivo al ser arquitectos.

Los test fueron llevados a cabo de tal manera que yo estaba presente y daba unas mínimas indicaciones de lo que debían intentar hacer y cuando el usuario no sabía que hacer le guiaba. El resultado no siempre fue positivo pero sí que fue mucho más útil de lo que esperaba ya que surgieron ideas que no tuve en cuenta inicialmente. La causa de estos problemas es que me he concentrado mucho más en la programación que en el diseño de la interfaz y la experiencia de usuario.

A continuación se describen los resultados más relevantes obtenidos (ninguno de estos ha podido ser añadido todavía a la aplicación):

Navegación

El botón 'projects list' que ahora utilizo para abrir y cerrar la lista de proyectos y al cual le pongo el nombre del proyecto actual resulta poco intuitivo como utilizarlo. Una vez un usuario ha accedido a un proyecto no sabía volver a la lista de proyectos. De hecho algunos hacían clic en la flecha atrás del navegador, lo cual me ha hecho ver clara la necesidad de modificar la historia del navegador cuando se accede a un proyecto para que si alguien hace clic vuelva realmente a la lista de proyectos en vez de salir de la aplicación.

Como sugerencias, una opción es poner un icono de una flechita hacia abajo en el botón, algo que también debería hacer en el botón de los dibujos para entender que se trata de botones que albergan contenido. Otra opción era poner una flecha back propia entre el logo de la aplicación y el título del proyecto o incluso utilizar el propio logo de la aplicación para volver.

Gestión de proyectos

Algunos usuarios se esperaban que los proyectos se pudiesen cerrar y que por lo tanto al volver a la lista de proyectos ninguno apareciese seleccionado. Les confundía incluso ver la lista de proyectos abierta y que la url se quedase con la del proyecto actual. Actualmente a partir de la primera vez que se selecciona uno de la lista siempre hay uno abierto.

Sé que poder cerrar un proyecto es necesario y lo tengo previsto.

El tema de la url no obstante lo he hecho así por comodidad ya que si alguien quiere consultar la lista de proyectos sin cerrar el proyecto puede hacerlo y es solo en el momento de seleccionar otro proyecto cuando cambia a otro.

Pantalla inicial visor

Cuando accedían a un proyecto les resultaba confuso que no hubiese nada, ya que el visor estaba vacío y no sabían por dónde empezar.

Mi idea al inicio era la de seleccionar un dibujo por defecto. No obstante las propuestas de los usuarios han sido muy útiles ya que han propuesto crear un dashboard donde te aparezcan por ejemplo todos los dibujos disponibles así como otros archivos o datos del proyecto como imágenes o información de miembros.

Comentarios

Sobre los comentarios no hubieron muchas dudas, simplemente me recomendaron mirar como lo hacían otras aplicaciones como Word. Este dispone por ejemplo de diferentes estados de visibilidad.

Capítulo 5: Conclusiones y líneas de futuro

1. Conclusiones

En general estoy satisfecho con el resultado obtenido porque considero que he llegado a un grado de desarrollo de la aplicación suficiente como para poder presentarla en mi empresa y que esta sea considerada un producto valido en el cual invertir.

Como aspectos concretos considero que he aprendido mucho en cuanto a programación y a hacer uso de servicios de terceros como APIs.

En cuanto a los objetivos, no todos han sido cumplidos, en algunos casos por falta de tiempo y en otros por falta de conocimientos, ya que he ido aprendiendo a medida que avanzaba el proyecto.

Me he centrado mucho en la programación ya que es lo que me interesa y quizás debería haber dedicado más tiempo a otros aspectos.

También he aprendido la importancia de la planificación y de seguir un orden claro a la hora de desarrollar un proyecto. Cuanto más claro esté el diseño y la estructura más fácil será la fase de implementación, ya que sino a medida que se implementa surgen muchas dudas y se acaba perdiendo mucho tiempo y recursos.

La planificación original para el proyecto fue modificada varias veces normalmente debido a que habían tareas que conllevaban más tiempo del previsto o porque simplemente las repriorizaba.

Considero que la metodología empleada ha sido correcta aunque creo que desarrollar un proyecto solo es mucho trabajo y hay que tomar muchas decisiones. Tener la posibilidad de trabajar en equipo hubiese sido muy valioso.

En resumen pese a las dificultades me quedo con la satisfacción de haber tenido la oportunidad de desarrollar un producto real que quizás algún día pueda llegar a ser útil para profesionales de los campos de la arquitectura, ingeniería y construcción

2. Líneas de futuro

Como línea de futuro, teniendo en cuenta que esta aplicación es solo una parte de un proyecto mayor que incluye también el exportador, el primer paso será continuar con el desarrollo de este último. Lo cual, pese a que dispongo de la ayuda de los desarrolladores de VisualARQ, no será una tarea fácil, es por este motivo que no hay casi aplicaciones que dispongan de "dibujos inteligentes". Como alternativa en el caso de que no fuese posible adaptaría la aplicación para que en vez de dibujos trabajase con modelos 3D, cambiando el canvas SVG actual por un canvas WebGL.

Bibliografía

Rodríguez, José Ramón. Definición de objetivos (2.1). ***En La gestión del proyecto a lo largo del trabajo final***. (pp.10-11). Universitat Oberta de Catalunya.

Rodríguez, José Ramón. Definición del alcance (2.2). En ***La gestión del proyecto a lo largo del trabajo final***. (pp.11-13). Universitat Oberta de Catalunya.

Rodríguez, José Ramón. Planificación (3). En ***La gestión del proyecto a lo largo del trabajo final***. (pp.17-23). Universitat Oberta de Catalunya.

What is the state-of-the-art of BIM (Building Information Modeling) today?
https://www.youtube.com/watch?v=cgSeRxTKteU

BIM Software Tools for all Occasions
https://www.e-zigurat.com/blog/en/bim-software-tools-for-all-occasions/

The Ultimate BIM Software List for 2019
https://www.lodplanner.com/bim-software/

Google Drive Api
https://developers.google.com/drive

MDN web docs | Using Promises
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises

Styling & Customizing File Inputs the Smart Way
https://tympanus.net/codrops/2015/09/15/styling-customizing-file-inputs-smart-way/

MDN web docs | JavaScript modules
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules

MDN web docs | DOMParser
https://developer.mozilla.org/en-US/docs/Web/API/DOMParser

MDN web docs | Parsing and serializing XML
https://developer.mozilla.org/en-US/docs/Web/Guide/Parsing_and_serializing_XML

How to create a custom context menu for your web application
https://dev.to/iamafro/how-to-create-a-custom-context-menu--5d7p

Sitepoint | How to Build Push Notifications for Web Applications
https://www.sitepoint.com/how-to-use-push-notifications-for-web-applications/

PWA Codelab
https://codelabs.developers.google.com/codelabs/your-first-pwapp

Firebase | Cloud Functions
https://firebase.google.com/docs/functions

Firebase | Firestore
https://firebase.google.com/docs/firestore

Medium | Send email with a Firebase cloud function
https://medium.com/@edigleyssonsilva/cloud-functions-for-firebase-sending-e-mail-1f2631d1022e

Web Fundamentals | BroadcastChannel API: A Message Bus for the Web
https://developers.google.com/web/updates/2016/09/broadcastchannel

Web Fundamentals | Progressive Web Apps on Desktop
https://developers.google.com/web/progressive-web-apps/desktop

Web Fundamentals | Patterns for Promoting PWA Installation (mobile)
https://developers.google.com/web/fundamentals/app-install-banners/promoting-install-mobile

Anexos

Anexo A: Entregables del proyecto

Planificación del proyecto: Archivo pdf con el desglose de todas las tareas llevadas a cabo.

Estimación de presupuesto: Archivo pdf con el detalle de la estimación de presupuesto del proyecto.