UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Desarrollo de interfaces gráficas a partir de mockups

Alexander Berguer

Sebastián Mateo

Gonzalo Melgar

Programa de Grado en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Mayo de 2017

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Desarrollo de interfaces gráficas a partir de mockups

Alexander Berguer

Sebastián Mateo

Gonzalo Melgar

Tesis de Ingeniería presentada al Programa de Grado en Computación, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Ingeniero en Computación.

Director:
Dr. Ing. Daniel Calegari

Montevideo – Uruguay
Mayo de 2017

Montevideo – Uruguay
Mayo de 2017

RESUMEN

El paradigma de Ingeniería Dirigida por Modelos (MDE por sus siglas en inglés) propone la construcción de software basado en una abstracción de su complejidad, a través de la definición de modelos y en un proceso de construcción semi-automático guiado por transformaciones de estos modelos. Una transformación de modelos, es la generación automática de un modelo de destino a partir de uno de origen, de acuerdo con un conjunto de reglas que describen cómo ciertos elementos del modelo origen, serán transformados en ciertos otros del modelo de destino. El uso de mecanismos automáticos tiende a mejorar la eficiencia y eficacia del proceso de construcción de software.

Existen propuestas para la creación automática de interfaces gráficas (GUI por sus siglas en inglés) a partir de mockups, es decir, bocetos gráficos de la GUI. Estas propuestas permiten la definición de aspectos estructurales y de comportamiento de una GUI a través de lenguajes gráficos de dominio específico. No obstante, dado que el foco es la esquematización rápida de la GUI, se dejan de lado variados aspectos necesarios para la construcción final de la misma.

Por otro lado, existe un estándar para la especificación de interfaces gráficas llamado Interaction Flow Modeling Language (IFML). IFML soporta la descripción de GUIs de manera independiente de la plataforma, para aplicaciones accedidas o instaladas en sistemas tales como computadoras de escritorio, laptops y teléfonos móviles. El foco de la descripción está en la estructura (contenido) y el comportamiento (navegación entre elementos, eventos, asociación con lógica y datos del negocio) de la GUI tal y como es percibida por el usuario final.

IFML podría ser considerado una capa intermedia que permite abstraer ciertos aspectos de la interfaz y expresar otros, por ejemplo el consumo de datos desde una base de datos, que sean considerados en un paso posterior para la generación automática de la interfaz. El objetivo general de este proyecto es evaluar la relación existente entre la especificación de aspectos de contenido, navegación entre elementos y eventos, de los lenguajes de especificación de mockups y su correspondiente especificación en IFML.

Para ello se realizó un estudio de los editores de mockups existentes en el mercado, evaluando sus puntos fuertes y débiles, así como de la relación existente entre la especificación de aspectos de contenido y navegación del mockup y su correspondiente especificación en IFML.

Se propuso un nuevo enfoque basado en MDE, que involucra la utilización de diagramas de mockups al principio del proceso de construcción de software, utilizando IFML como capa de abstracción, con el fin de generar interfaces en tecnologías específicas. Al modelo generado se le podría aplicar potencialmente cualquier transformación para llevarlo a alguna tecnología concreta.

Para validar el enfoque planteado, se desarrolló un prototipo de editor de mockups, así como también un módulo de procesamiento, que se encarga de transformar el mockup en un modelo IFML. Además, se definió una transformación que permite generar código HTML desde la especificación de la interfaz en IFML.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación

La Ingeniería Dirigida por Modelos (Model-Driven Engineering, MDE [1]) es un paradigma de ingeniería de software que se basa en la creación de modelos con distintos niveles de abstracción, incrementando la productividad y reutilización. Dentro de MDE existe un paradigma llamado 'Desarrollo Dirigido por Modelos (MDD) [2]', el cual hace uso de los modelos para facilitar el desarrollo de software. Esto se logra mediante un proceso de construcción semi automático guiado por transformaciones aplicadas a dichos modelos. Una transformación de modelos es la generación automática de un modelo destino a partir de uno origen, de acuerdo a un conjunto de reglas que describen cómo ciertos elementos del modelo de origen serán transformados en ciertos otros del modelo de destino.

En la actualidad, dentro del mundo del desarrollo de Software, la construcción de prototipos visuales de las interfaces, comúnmente llamados mockups [3], es cada vez más utilizada para determinar los requerimientos iniciales de un sistema a desarrollar. Los desarrolladores utilizan mockups para especificar aspectos de estructura, tales como la disposición de elementos dentro de la interfaz, tamaños, jerarquías, entre otros, así como también especifican el comportamiento de una interfaz, tales como las interacciones entre los elementos o navegación entre páginas. Debido al avance tecnológico en los últimos años, han surgido diversas herramientas (ej.: Axure[4], Balsamiq[5], Pencil[6]) para la creación de mockups con muchas prestaciones distintas y diversos componentes.

A su vez, existe un lenguaje de modelado de interfaces gráficas llamado Interaction Flow Modeling Language (IFML)[7]. Fue adoptado como estándar [8] para la especificación de interfaces, por el consorcio Object Management Group (OMG)[9], el cual se dedica al cuidado y establecimiento de diversos estándares de tecnologías. IFML permite describir GUIs independientemente de la plataforma, las cuales pueden ser aplicaciones de escritorio, móviles o web.

Aplicando los conceptos vistos anteriormente, se puede ver a IFML como una capa, que permite especificar los aspectos de estructura y comportamiento de los elementos que forman un mockup, estableciendo una correspondencia entre dichos elementos y los que componen IFML. Vale destacar, que IFML también permite especificar aspectos de negocio y datos, pero debido al contexto de este proyecto no serán considerados. En un paso posterior, aplicando técnicas de MDE y transformaciones, se puede llevar a cabo un proceso de generación automática y obtener así el código de las interfaces para plataformas específicas.

Es por esto que la principal motivación de este proyecto es definir un enfoque con el cual poder reutilizar los mockups, incluyéndose como parte del proceso de construcción de software, aprovechando así el tiempo invertido en su desarrollo.

Para ello se define como principal objetivo el evaluar la relación existente entre la especificación de aspectos de contenido, navegación entre elementos de los lenguajes de especificación de mockups y su correspondiente especificación en el lenguaje IFML. Entrando más en detalle, los objetivos específicos del proyecto son los siguientes:

- Evaluar las funcionalidades provistas por los lenguajes de especificación de mockups existentes en el mercado.
- Estudiar IFML e identificar las construcciones provistas por el mismo para la especificación de aspectos de contenido, navegación entre elementos y eventos.
- Analizar la relación existente entre un lenguaje de especificación de mockups e IFML.
- Desarrollar prototipos aplicando MDD que permitan evaluar en la práctica dicha relación

1.2. Organización del documento

El resto del documento está organizado de la siguiente manera: En el Capítulo 2 se define el marco teórico, presentando los conceptos necesarios para la comprensión del proyecto. Luego en el Capítulo 3 se detalla el estudio de diversas herramientas de mockups, mostrando un resumen del análisis de cada una de ellas. En el Capítulo 4 se brinda el contexto de la realidad planteada, junto con el análisis realizado entre los mockups y los lenguajes involucrados. En este capítulo también se presenta el diseño y la arquitectura de la solución propuesta. En el Capítulo 5 se presentan los detalles de la implementación del prototipo realizado como prueba de concepto del enfoque planteado. El Capítulo 6 muestra un caso de estudio completo, detallando las acciones realizadas entre las distintas etapas del proceso. En el Capítulo 7 se presentan las conclusiones así como también una evaluación de la solución planteada, finalizando con los trabajos futuros propuestos.

Junto al documento se incluyen dos anexos. En el Anexo 1 se puede ver una descripción en alto nivel de algunos de los componentes principales del metamodelo de IFML, y el Anexo 2 detalla cada una de las herramientas de mockup estudiadas, brindando una breve descripción de cada una de sus características.

Capítulo 2

Marco Teórico

En este capítulo se presentan los principales conceptos necesarios para la comprensión de este proyecto, particularmente el de Ingeniería Dirigida por Modelos y el lenguaje de modelado IFML.

2.1. Ingeniería Dirigida por Modelos

Un problema común hoy en día, es el crecimiento de la complejidad del desarrollo de software, debido a la demanda por parte de los clientes de tener más funcionalidades en menos tiempo. Además, con los avances en tecnología en casi todas las áreas profesionales, ha sido necesario desarrollar software en múltiples dominios. Esto lleva a problemas tales como la preparación que requiere un equipo de desarrollo en torno a estas áreas; se debe tener cierto conocimiento del dominio del sistema, requiriendo un periodo de adaptación en el que no se está desarrollando. Es por esto que surge la necesidad de idear nuevas herramientas dentro de la ingeniería de software, con la cual de cierta forma poder separar los dominios del desarrollo, disminuyendo la complejidad y así poder cumplir con los tiempos, costos y alcances de los proyectos. De este modo se presenta un nuevo enfoque llamado Ingeniería Dirigida por Modelos (MDE por sus siglas en inglés), el cual podemos definir como un paradigma de ingeniería de software que centra el modelado como actividad principal del ciclo de vida de un sistema de software; actividades como el diseño, construcción y mantenimiento de software, entre otros. Esto se logra a través de la definición y construcción de modelos, para luego aplicar transformaciones entre ellos. Uno de los objetivos principales es aplicar las ventajas del modelado a las activida-

des de ingeniería de software. En particular para el desarrollo de software, la aplicación de este enfoque se denomina Model Driven Development (MDD). MDE se ha vuelto muy importante, llegando a ser una fuerte área de estudio por parte de investigadores que velan por la importancia de usar modelos como los artefactos principales para el ciclo de vida de un sistema de software.

Antes de entrar en detalle es necesario definir algunos términos para comprender mejor el tema [10]:

Abstracción

En el contexto de la ingeniería de software, abstraer significa destacar una serie de características esenciales de un sistema u objeto, desde un determinado punto de vista, ignorando aquellas otras características que no son relevantes desde esa perspectiva. En otras palabras, dado un punto de vista, nivel o jerarquía dentro de un sistema, se debe tomar en cuenta solamente las características las cuales son relevantes en ese contexto, simplificando así el dominio del problema.

Modelo

El OMG define un modelo como 'una representación de una parte de la funcionalidad, estructura y/o comportamiento de un sistema'[11]. Siendo más general y teniendo en cuenta la definición anterior, se define al modelo como una abstracción simplificada de un sistema o concepto de la realidad, expresado en un lenguaje bien definido y con un propósito determinado.

Metamodelo

Siguiendo la filosofía de MDE, se puede decir que un metamodelo es un modelo que especifica los conceptos de un lenguaje, las relaciones entre ellos y las reglas estructurales que restringen los posibles elementos de los modelos válidos, así como aquellas combinaciones entre elementos que respetan las reglas semánticas del dominio. Cada modelo se escribe en el lenguaje que define su metamodelo, lo que se denomina conformidad; esto es que el modelo cumple con las reglas del metamodelo. Dado que un metamodelo es también un modelo, viene definido a su vez por su meta-metamodelo. El OMG propone una organización en 4 niveles [2] como se muestra en la Figura 2.1.

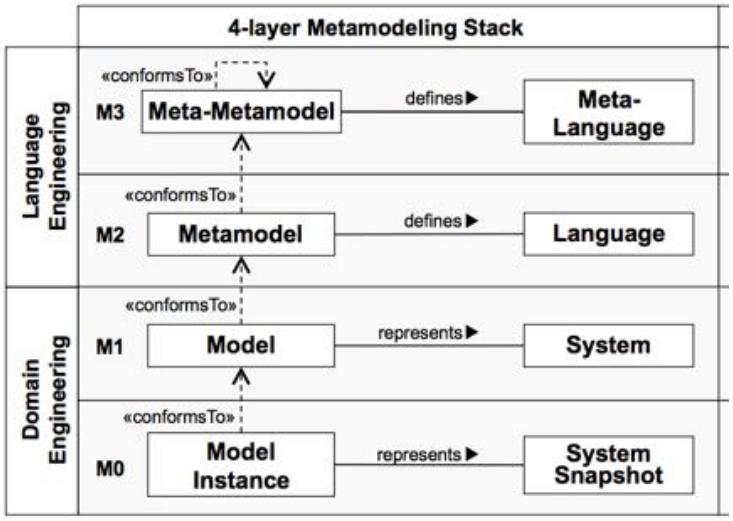


Figura 2.1: Organización en cuatro niveles

La mitad superior de la figura se encarga de construir modelos para definir lenguajes de modelado, mientras que la mitad inferior, es responsable de construir modelos para dominios particulares. En la capa superior, nombrada M3, residen los lenguajes de metamodelado, los cuales especifican los conceptos para definir metamodelos. Si bien en teoría se podrían definir infinitos niveles de metamodelados, en la práctica los meta-metamodelos pueden ser definidos por ellos mismos, por lo que no tiene sentido ir más allá de ese nivel de abstracción. En la capa M2 residen los metamodelos que representan lenguajes de modelado y pueden ser instanciados para construir modelos de la capa M1, los cuales representan sistemas definiendo conceptos del dominio. En la capa M0 se encuentran las instancias de los conceptos del dominio, los cuales representan entidades del mundo real.

Transformación

Ya que los modelos son las piezas claves en MDE, se debe contar con mecanismos de manipulación de los mismos. Es aquí donde surgen las transformaciones como otro componente clave en MDE, y se definen como un conjunto de reglas que describen cómo un modelo origen, expresado en un lenguaje determinado, puede ser transformado en otro modelo en un lenguaje destino. Más concretamente una transformación es el proceso de convertir un modelo en otro modelo del mismo sistema, junto a la especificación de las reglas. Estas transformaciones se pueden clasificar con distintos criterios, ya sea por el nivel de

abstracción, tipo de lenguaje, direccionalidad, entre otros. A continuación se describe la clasificación más importante, la cual clasifica a las transformaciones atendiendo al tipo de modelo destino:

- **Modelo a modelo (M2M):** Son transformaciones que generan modelos a partir de otros modelos, esto es, que tanto el origen como el destino son modelos. Uno de los lenguajes más utilizados a la hora de especificar transformaciones M2M es el lenguaje QVT [12] (acrónimo del inglés query-view-transformation). Es un estándar propuesto por el OMG capaz de expresar transformaciones, realizar consultas (query) para la selección y filtrado de elementos en el modelo de entrada y también generar views que permiten la visualización específica de los metamodelos.
- **Modelo a texto (M2T):** Son transformaciones que generan cadenas de texto a partir de modelos, las cuales normalmente son utilizadas para generación de código fuente o generación automática de documentación. Para este caso el OMG ha propuesto lenguajes de especificación de transformaciones, siendo el más utilizado el 'MOF Model to Text Language (MTL)' [13] que particularmente fue utilizado en esta investigación, utilizando la implementación provista por la herramienta Acceleo [14].

En la Figura 2.2 se ilustra cómo interactúan los conceptos previamente definidos en la arquitectura MDE en un ejemplo de transformación M2M.

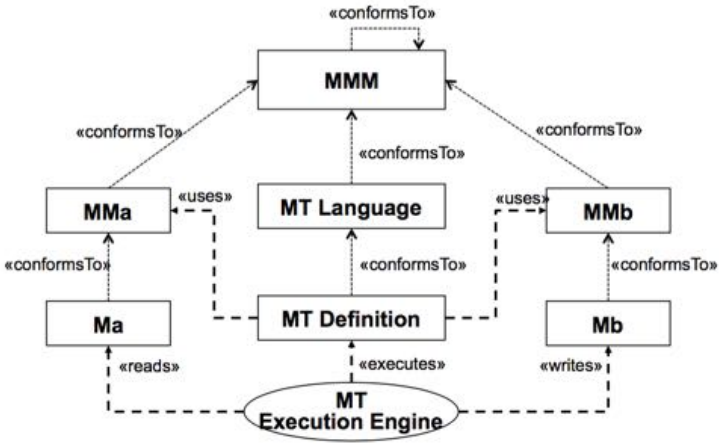


Figura 2.2: Transformacion de Modelos [2]

En ella se puede apreciar un modelo **Ma** el cual es conforme al metamodelo **MMa**. Al modelo **Ma** se le aplican transformaciones con el objetivo de generar

otro modelo (**Mb**) el cual de manera análoga es conforme a otro meta modelo **MMb**. La transformación al ser también un modelo, es conforme a un lenguaje de transformaciones, y junto con los metamodelos anteriores son conformes a un meta-metamodelo, que está por encima del nivel de abstracción y que se define a sí mismo según lo propuesto por el OMG. Finalmente la transformación es llevada a cabo mediante un motor de ejecución de transformaciones, el cual es el encargado de aplicar las reglas de transformación y generar el modelo destino.

2.2. Mockups y Web

En un ámbito general, un mockup se puede definir como un modelo que representa total o parcialmente un objeto o una idea, con fines de demostración. Pueden representar tanto como software, dispositivos o hasta un edificio. Este proyecto se basa en el uso de mockups que representan páginas web. Estos podrán ser creados mediante algún software diseñado con ese fin o incluso utilizando un papel. A la vista, el mockup se parecerá mucho al resultado final esperado, pero éste no tendrá la lógica de negocio necesaria, ya que su finalidad es representar una abstracción de la interfaz que sea mas sencilla de construir. La realización de mockups ayuda mucho al proceso de diseño de la interfaz de usuario, pudiendo realizar los cambios en etapas tempranas de desarrollo y no cuando se tiene un sistema final desarrollado. En la Figura 2.3 se puede ver un ejemplo de mockup, el cual representa de forma general la estructura de un sitio web.

Figura 2.3: Ejemplo de mockup

En este caso, al ser los mockups representaciones de páginas web, es fácil observar que existirá una relación entre ellos y el lenguaje utilizado para desarrollar dichas páginas. El 'HyperText Markup Language' (HTML, por sus siglas en inglés) [15], es el lenguaje estándar a la hora de crear y representar visualmente una página web. Determina el contenido de la misma, pero no su funcionalidad. Cuando se habla de hipertexto se refiere a los enlaces que logran conectar una página web con otra, ya sea dentro de un mismo sitio o no, lo que permite la navegabilidad. Cuando se habla de Markup, se refiere a que se utiliza un lenguaje de marcado, el cual incluye elementos especiales denominados tags o etiquetas, los cuales son interpretados por el navegador para luego desplegar la página web con la estructura tal como fue concebida. Algunos de los tags principales existentes son: `<head>`, `<body>`, `<header>`, `<a>`, `<p>`, `<div>`, `<img>`, entre otros.

2.3. IFML: The Interaction Flow Modeling Language

En esta sección se pretende brindar un panorama general con respecto al lenguaje de modelado Interaction Flow Modeling Language (IFML) [8][7], comentando el alcance del mismo y sus aspectos relevantes para la aplicación en este proyecto.

Antes de presentar IFML, es necesario definir los lenguajes de dominio específico (DSL por sus siglas en inglés) [2]. Estos lenguajes proporcionan los conceptos y notaciones de un dominio particular, los cuales permiten a los expertos de dominio poder expresar los modelos del sistema en el nivel de abstracción adecuado. IFML en particular, es un DSL de modelado que se encarga de describir interfaces gráficas independientemente de la plataforma que se utilice, ya sean aplicaciones de escritorio, móviles o web. IFML hace foco principalmente en la estructura, comportamiento e interacción de una aplicación tal y como la percibe el usuario final; es capaz de especificar la estructura de una interfaz de usuario, los elementos que la componen así como la interacción entre los elementos y la interacción del usuario con la interfaz. Este lenguaje tiene también la capacidad de referenciar y asociar datos a los componentes de la interfaz, utilizando los objetos de dominio que proveen los datos. También puede incorporar lógica de negocio, haciendo uso de acciones

que definiremos más adelante, aspectos que influyen en la experiencia de usuario. Su principal objetivo es proveer las herramientas para poder definir los modelos que describen el front-end de una aplicación, independientemente de la tecnología en que se realice.

IFML cubre varios aspectos de una interfaz de usuario, los que se detallan a continuación:

- **La composición de la vista:** dentro de las cuales están las unidades de visualización que componen la interfaz, su organización, cuáles son desplegados simultáneamente y cuáles en mutua exclusión, así como también la jerarquía que existente entre ellas.
- **El contenido de la vista:** referente al contenido que será desplegado en la aplicación al usuario y que entradas serán obtenidas del usuario y provistas a la aplicación.
- **Los comandos:** son los eventos de interacción que son soportados.
- **Las acciones:** elementos de la lógica de negocio, ejecutados mediante eventos.
- **Los efectos de la interacción:** cual es el efecto que tienen los eventos y las acciones sobre el estado de la interfaz.
- **El vínculo de parámetros:** especifica qué datos son comunicados entre los elementos de la interfaz y las acciones que se llevan a cabo.

La especificación de IFML está compuesta por 4 elementos principales:

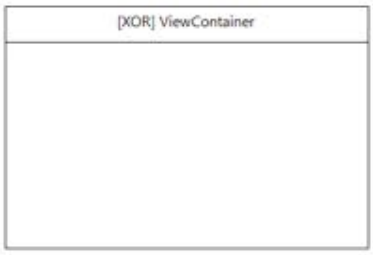
1. **El metamodelo IFML**: el cual especifica la estructura y la semántica de IFML utilizando el estándar MOF.
2. **El perfil UML de IFML**: que define una sintaxis basada en el Unified Modeling Language (UML por sus siglas en inglés) [16] para expresar los modelos IFML. Particularmente, el perfil extiende conceptos de varios tipos de diagramas UML, como pueden ser, los diagramas de clases, máquinas de estado, etc.
3. **La sintaxis visual de IFML**: la cual ofrece una forma concreta de expresar los modelos IFML, un diagrama capaz de encapsular los aspectos de la interfaz de usuario que de otra manera si solamente estuviéramos usando UML tendríamos que expresarlo utilizando múltiples diagramas distintos.

4. **El XMI** [17] **del modelo IFML**: expresada con la sintaxis de XML [18] para la serialización de los modelos. Lo que permite contar con un formato de intercambio específico, que nos provee de portabilidad.

2.3.1. Componentes

Un diagrama IFML está formado por varios componentes, algunos de los cuales serán descritos a continuación:

Nombre	**Notación Gráfica**	**Descripción**
ViewContainer		Elemento de la interfaz que contiene elementos, muestra contenido y permite la interacción con otros ViewContainers. Cada uno de los mismos se puede estructurar en una jerarquía de subcontainers con su estructura propia, como por ejemplo en una aplicación web podría contener una página principal la cual incluye diversas secciones que serán manejadas como un container particular.
Default ViewContainer		La propiedad de default, determina que el ViewContainer será presentando por defecto cuando su ViewContainer padre es accedido.

Nombre	Notación Gráfica	Descripción
XOR ViewContainer		Si un ViewContainer es marcado como XOR significa que los ViewContainer hijos podrán ser mostrados de uno a la vez.
Landmark ViewContainer		La propiedad de Landmark determina que el ViewContainer puede ser accedido desde cualquier otro elemento que se encuentre a su mismo nivel, sin necesidad de definir todas las interacciones (InteractionFlows).
ViewComponent		Elemento de la interfaz que muestra contenido o permite una entrada de datos. Un ViewComponent puede tener parámetros de entrada o de salida.
NavigationFlow		NavigationFlow es una subclase de InteractionFlow y especifica una dependencia de entrada-salida. El origen del enlace genera una salida que es asociada con el destino del enlace.
DataFlow		Subclase de NavigationFlow que especifica el Intercambio de datos entre ViewComponents o Actions como consecuencia de una interacción previa con el usuario.

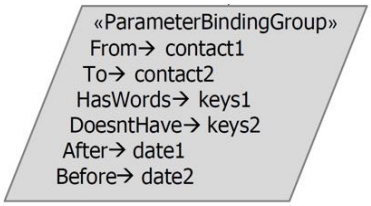
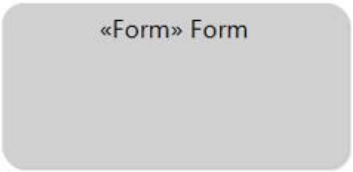
Nombre	Notación Gráfica	Descripción
Action		Una pieza de la lógica de negocio activada por un evento.
Parameter Binding		Especificación de como un parámetro de entrada de un origen se asocia con un parámetro de salida de un destino.
Parameter Binding Group		Conjunto de ParameterBindings asociados a un InteractionFlow (siendo su navegación o flujo de datos).
Event		Afecta el estado de la aplicación, lo que puede generar una navegación, o el traspaso de parámetros entre elementos.

Tabla 2.2: Lista de componentes IFML

A su vez IFML posee un conjunto de elementos disponibles que son extensiones de componentes ya existentes:

Nombre	**Notación Gráfica**	**Descripción**
Select Event		Un SelectEvent es un evento que soporta la selección de uno o más elementos de un conjunto. Cuando es disparado, causa que el o los elementos seleccionados sean pasados como un parámetro al destino del NavigationFlow asociado.
Submit Event		Un SubmitEvent es un evento que realiza la acción de submit, el cual se realiza pasando los parámetros del ViewContainer asociado, al ViewComponent o al Action que sea destino del NavigationFlow del evento.
List		Es un ViewComponent que se utiliza para mostrar una lista de objetos. Cuando una lisita se asocia con un Event, significa que cada elemento desplegado en la lista puede ser usado para disparar dicho evento.
Form		Es un ViewComponent que se utiliza para representar un formulario de ingreso de datos.

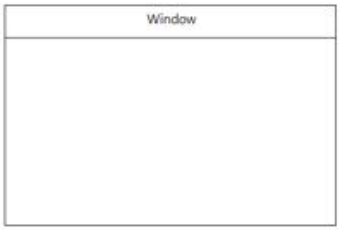
Nombre	Notación Gráfica	Descripción
Details		ViewComponent utilizado para mostrar los detalles de una instancia de DataBinding específica.
Window		Es una extensión de un ViewContainer que se utiliza para representar una ventana de una interfaz de usuario. Puede ser Modal o Modeless dependiendo del comportamiento requerido. Una ventana Modal se abre como una nueva ventana, y se desactiva la interacción con lo que queda atrás. En cambio una ventana Modeless, abre una nueva ventana, pero se permite que el usuario interactúe con las otras partes de la interfaz.

Tabla 2.4: Extensiones por defecto de IFML

2.3.2. Ejemplos básicos de representación IFML

Tienda de Libros

En la figura Figura 2.4 se muestra un ejemplo básico que utiliza algunos de los componentes de la Subsección 2.3.1. Éstos forman un diagrama de IFML sencillo, creado a partir de diagramas de mockups que ilustran el punto de vista del usuario a alto nivel:

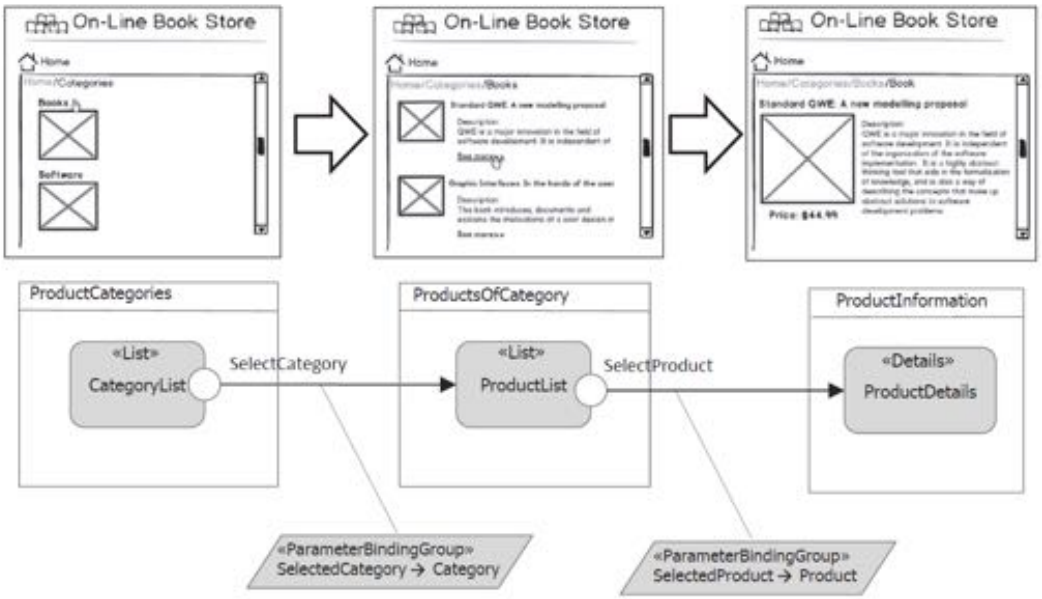


Figura 2.4: Ejemplo de representación IFML: Tienda de Libros

Se representa una tienda de libros, la cual presenta tres pantallas distintas:

- **Listado de categorías:** donde se muestra una lista con las categorías existentes en dicha página. La correspondencia en IFML de esta pantalla es un `ViewContainer`, el cual dentro tiene un elemento de tipo `List` que representa el listado de categorías existentes. Dicha lista tiene asociado un `SelectEvent` el cual tiene un `NavigationFlow` asociado, con origen el evento y destino el `ViewContainer` de la siguiente página. A su vez dicho `NavigationFlow` tiene asociado un `ParameterBindingGroup` el cual contiene la información de la categoría seleccionada de la lista.
- **Listado de libros de una categoría específica:** similar a la pantalla anterior, solamente que se muestran en una lista los libros para una categoría en particular. La correspondencia en IFML de esta pantalla, como se ve en la Figura 2.4, es muy similar a la del listado de categorías, posee un `ViewContainer` el cual contiene un elemento `List`, en este caso representando los libros, el cual tiene asociado un `SelectEvent` vinculado con un `NavigationFlow`. Éste tiene como destino el `ViewContainer` de la siguiente página y posee un `ParameterBindingGroup` asociado que contiene la información del libro seleccionado de la lista.

- **Detalle de un libro específico:** encargada de mostrar los detalles de un libro específico previamente seleccionado. La correspondencia en IFML está dada por un `ViewContainer` que contiene un componente `Details`, el cual muestra los detalles de la instancia `DataBinding` especifica, que en este caso la el libro seleccionado desde la página anterior, el cual es enviado a través del `ParameterBindingGroup` previamente descrito.

Asistente de configuración

El segundo ejemplo seleccionado se trata de un asistente de configuración de tres pasos, con la posibilidad de navegar ellos. Estos forman un diagrama de IFML sencillo, creado a partir de diagramas de mockups que ilustran el punto de vista del usuario a alto nivel:

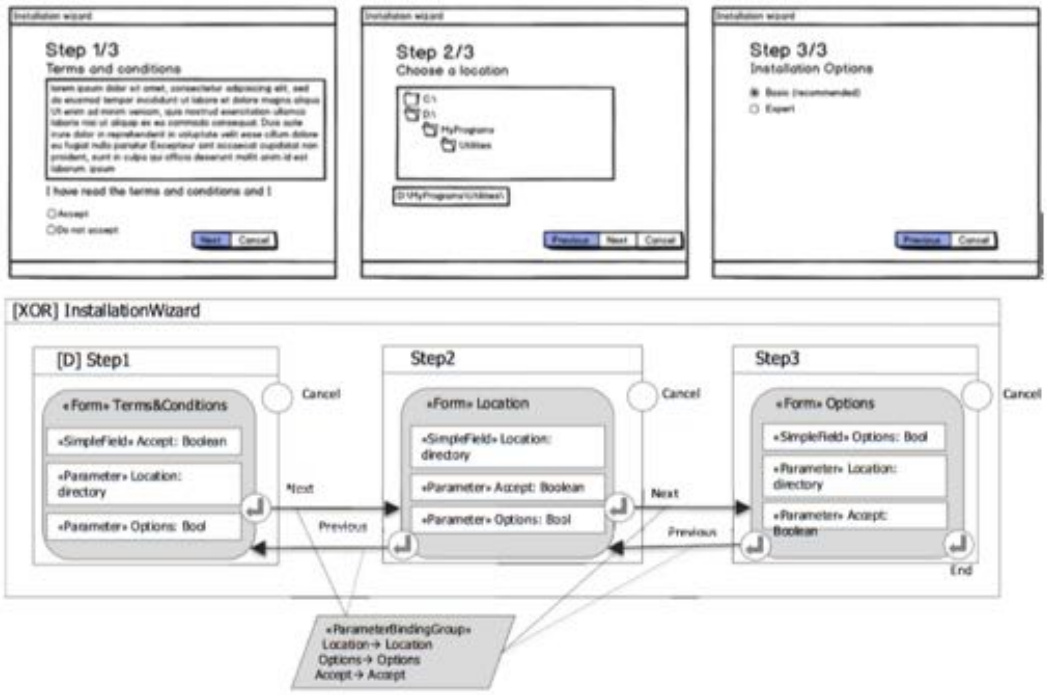


Figura 2.5: Ejemplo de representación IFML: Asistente de Configuración

Se muestra un asistente típico de tres pasos, donde cada paso contiene un formulario con información particular de cada caso. En IFML esto se ve representado con un `XOR ViewContainer` padre que tiene tres `sub ViewContainers` que representan cada paso.

Se analiza cada caso individualmente:

- **Paso 1:** muestra los términos y condiciones para poder continuar con el asistente. Posee un cuadro de texto que muestra los términos, dos radio buttons y botones para cancelar y seguir con el proceso. La correspondencia IFML está dada por un `Default ViewContainer` ya que es el primer paso del asistente, el cual contiene un elemento `Form`, que contiene un `SimpleField` que representa la opción seleccionada del radio button, y dos `Parameter` para la ubicación y la opción seleccionada. El mismo tiene asociado dos `Events` un `SubmitEvent` el cual redirige al próximo paso mediante un `NavigationFlow` que posee un `ParameterBindingGroup` asociado que contiene la información establecida. También tiene un `Event` que cancela todo el flujo.
- **Paso 2:** permite elegir un directorio dentro para continuar con el asistente. Posee tres botones; uno para continuar otro para volver y un tercero para cancelar. La correspondencia IFML está dada por un `ViewContainer`, el cual contiene un elemento `Form`, que contiene un `SimpleField` que representa la opción seleccionada del directorio, y dos `Parameter` para la ubicación seleccionada y la opción seleccionada. El mismo tiene asociado tres `Events` un `SubmitEvent` el cual redirige al próximo paso, un `SubmitEvent` el cual redirige al paso previo y un `Event` que cancela todo el flujo. Tanto el `SubmitEvent` de avanzar como de regresar contienen un `NavigationFlow` que posee un `ParameterBindingGroup` asociado con la información necesaria.
- **Paso 3:** encargada de establecer una configuración específica para completar el asistente. La correspondencia IFML está dada por un `ViewContainer`, el cual contiene un elemento `Form`, que contiene un `SimpleField` que representa la opción seleccionada del radio button, y dos `Parameter` para la ubicación seleccionada y la opción seleccionada. El mismo tiene asociado tres `Events` un `SubmitEvent` el cual finaliza el proyecto, un `SubmitEvent` el cual redirige al paso previo y un `Event` que cancela todo el flujo. Tanto el `SubmitEvent` de avanzar como de regresar contienen un `NavigationFlow` que posee un `ParameterBindingGroup` asociado con la información necesaria.

2.3.3. Metamodelo IFML

El metamodelo específico de IFML, se divide principalmente en 3 paquetes:

- El paquete de núcleo.
- El paquete de extensión.
- El paquete de DataTypes.

El paquete de núcleo contiene los conceptos que generan la interacción e infraestructura del lenguaje en términos de `InteractionFlowElements`, `InteractionFlows`, y los `Parameters`, los cuales fueron mencionados previamente. Así mismo, los conceptos que se encuentran en el núcleo pueden ser extendidos en el paquete de extensión, agregando comportamientos y parámetros que se adecúen a las necesidades del usuario que busca extender el mismo. Y por último, el paquete de `Datatypes`, que contiene los datatypes personalizados definidos por IFML. En el Anexo 1 se puede ver una descripción en alto nivel de algunos de los componentes principales del metamodelo de IFML. Se hará hincapié en las áreas más relevantes que hacen referencia a este proyecto, las cuales son: Modelo IFML, View Elements, Interaction Flow Model, Interaction Flow Model Elements y Events. Se hará una mención general de modelo por componente, explicando de forma general su estructura. Por su parte, en esta sección se hará foco en el metamodelo principal de IFML, el cual establece la estructura general del lenguaje y todos sus elementos.

A grandes rasgos el metamodelo de IFML se conforma tal como se aprecia en la Figura 2.6:

Dicho metamodelo es el contenedor de más alto nivel para el resto de los elementos de IFML. Contiene principalmente un `InteractionFlowModel`, un `DomainModel` y opcionalmente podría contener `ViewPoints`. Como se puede apreciar en el metamodelo, un `ViewPoint` presenta un aspecto específico del sistema, mientras que un `InteractionFlowModel` es la vista que obtiene el usuario de toda la aplicación y todos sus componentes. El `ViewPoint` tiene una referencia a un conjunto de `InteractionFlowModelElements`, los cuales definen una parte funcional completa del sistema. Un `InteractionFlowModelElement` es una clase abstracta que generaliza cada elemento de un `InteractionFlowModel`. Por otra parte un `DomainModel` representa la descripción del contenido y el comportamiento dentro del `InteractionFlowModel`. El mismo `DomainModel` es-

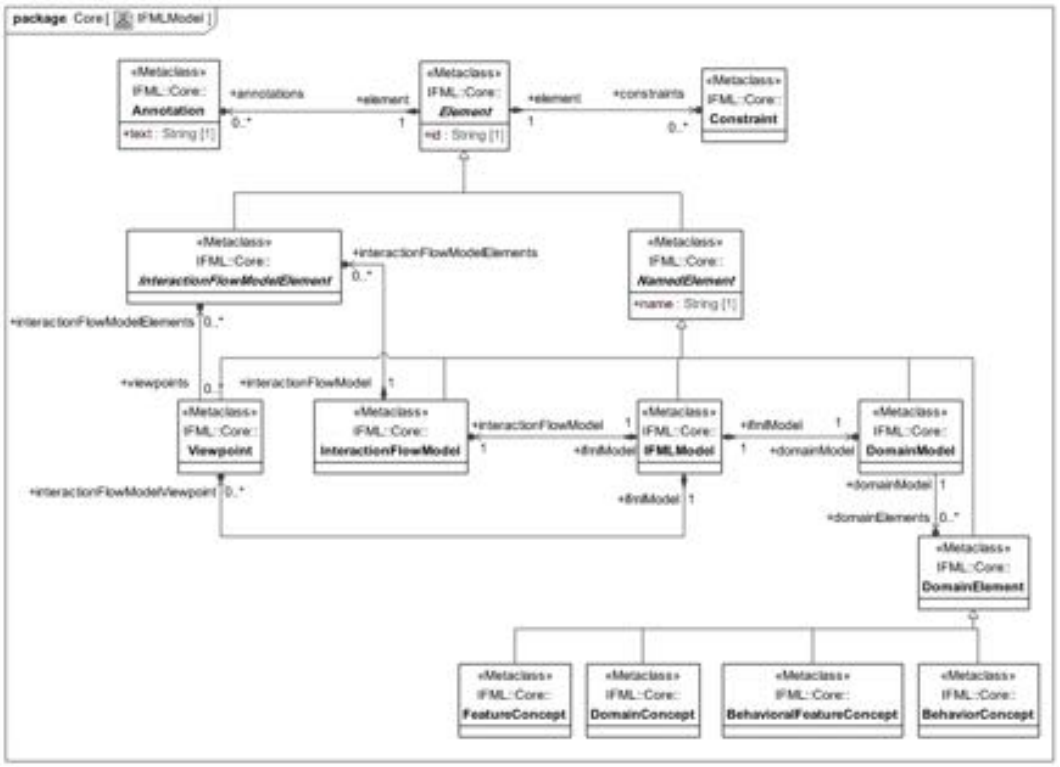


Figura 2.6: Metamodelo de IFML (reducido)

tá compuesto de `DomainElements`, los cuales pueden ser conceptos, propiedades, comportamientos y métodos (en el diagrama se encuentran representados como `DomainConcept, FeatureConcept, BehaviorConcept y BehavioralFeatureConcept` respectivamente). Otro punto a destacar es que para cualquier elemento existente, se pueden definir `Annotations` o Restricciones (`Constraints`).

2.3.4. Extensión

El metamodelo de IFML posee un conjunto básico de elementos, pero esto puede no ser suficiente a la hora de expresar una interfaz completa. Es por esto que se brindan mecanismos de extensión sobre todos los elementos de IFML, con el fin de permitir al desarrollador ampliar la gama de componentes a usar en su diseño. Ya existen extensiones incluidas en el estándar de IFML, y es posible agregar extensiones propias, diseñadas con un fin específico.

El objetivo principal que se busca a la hora de agregar extensiones es el de proveer de expresividad al lenguaje, hacer que los conceptos o las notaciones sean menos abstractas y que se adapten mejor a los conocimientos de los

diseñadores, etc.

Las ventajas de las extensiones se ven incrementada cuando se realizan bajo la perspectiva de un dominio de aplicación específico, donde puede ser necesario contar con determinados estilos en las interfaces, determinada terminología o algunas restricciones específicas de la tecnología en cuestión. Por ejemplo, las aplicaciones web poseen varios conceptos y terminologías que difieren con los de las interfaces gráficas de aplicaciones de escritorio.

A modo de ejemplo, se puede ver que una extensión de un `ViewContainer` puede ser considerada como una página web, esto sería un `ViewContainer` que denota una interfaz web y podría contener atributos específicos tales como estilos, referencias de la página, entre otros. También, en una aplicación web se utilizan muchos tipos de listas para mostrar contenido, IFML provee el tipo `List`, que ofrece unas funcionalidades mínimas, pero puede ser extendida para soportar interfaces más reales. Las posibles extensiones pueden ser las siguientes: `Scrollable List, Dynamically-sorted List, Nested List,` entre otras.

2.4. Mockup Driven Development

Debido a que el objetivo de este proyecto es evaluar la viabilidad de utilizar mockups para generar código, utilizando como medio un enfoque basado en MDE, fue necesario investigar la existencia de estudios previos que partieran de esa base. En esta instancia, se presenta un enfoque MDE realizado por José Rivero et al. [19] para el reúso de mockups, al que llamaron Mockup Driven Development.

Como se vio anteriormente, los mockups han probado ser capaces de mejorar la eficiencia al momento de capturar nuevos requerimientos. Una de sus ventajas es que son entendibles tanto para el usuario final como para el desarrollador. Este enfoque consiste en usar un metamodelo que ayude a representar los mockups en una manera tal que sea independiente de la herramienta y de la tecnología. Un esquema general del flujo se presenta en la Figura 2.7.

Se proveen un conjunto de parsers (1), uno para cada herramienta de creación de mockups, que luego de pasar por una fase de procesamiento (2), permite representar cada mockup concreto en una instancia del metamodelo propuesto (3). Luego se pueden crear generadores para tecnologías concretas para transformar un modelo a código en determinada tecnología. Estos generadores deben escoger los widgets adecuados para cada componente, según la tecnología que

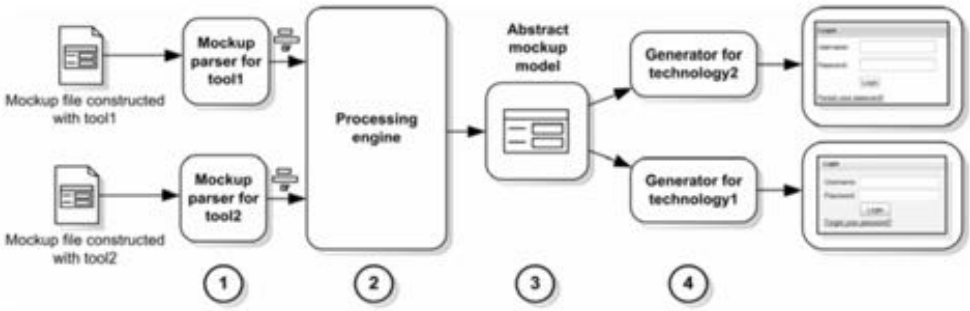


Figura 2.7: Enfoque Mockup driven development

se requiera.

Cada parser debe recorrer el archivo de mockups y retornar una colección de controles, los cuales se agrupan en páginas o en controles compuestos. Al final de la fase de procesamiento, se obtiene un modelo UI que consiste en páginas conteniendo una colección de componentes jerarquizada.

2.4.1. Metamodelo

Se presenta un metamodelo, el cual identifica los componentes principales de varias herramientas de construcción de mockups. El esquema es el siguiente:

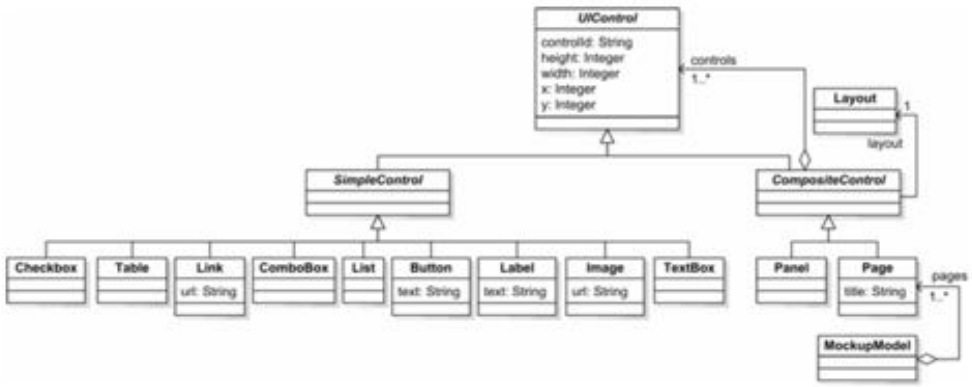


Figura 2.8: Metamodelo Mockups

El metamodelo tiene un objeto raíz, que es instancia `MockupModel` visto en el diagrama de enfoque general. Un `MockupModel` tiene una colección de páginas (instancias de `Page`) las cuales a su vez tienen una colección de `UIControls`, que representan componentes de un mockup. Cada `UIControl` tiene información acerca de su ancho, largo y posición, y puede ser simple o compuesto.

Por último cada `CompositeControl`, incluyendo las páginas, pueden tener su propio `Layout` para organizar sus componentes hijos.

Los autores afirman que el metamodelo propuesto es suficiente para representar los aspectos estructurales de un gran número de interfaces de usuario. A modo de simplificación, en esta instancia se evitó la especificación de aspectos de comportamiento, como la navegación o interacciones.

2.4.2. Extensibilidad

En relación a la extensibilidad de este esquema, se puede decir que agregar una nueva herramienta consiste en implementar un parser, tal que reciba un archivo mockup y retorna una colección de controles, agrupados en páginas o componentes compuestos.

Por otro lado, agregar una nueva tecnología de destino implica construir un generador para dicha tecnología, el cual reciba una instancia del metamodelo y retorne los artefactos correspondientes de la implementación concreta.

2.5. Conclusiones

En este capítulo fueron detallados los principales conceptos relacionados al paradigma Ingeniería Dirigida por Modelos, siendo su objetivo principal el aumento de la productividad en el proceso de ingeniería de software. Se mostró también que MDE ayuda a la reducción de errores en dicho proceso, logrando de esta forma un proceso eficiente ya que se aumenta el nivel de abstracción y se utilizan transformaciones para generar otro tipo de modelos de forma automática.

Además se presentó en detalle IFML, un lenguaje de modelado específico de dominio, que es apropiado para resolver la realidad planteada en este proyecto. Dicho lenguaje, además de permitir modelar elementos típicos de interfaces de usuario, permite ser extendido de forma sencilla.

Además, se vio que existen otras investigaciones con un enfoque similar al buscado en este proyecto, pero en estos trabajos se define un metamodelo desde cero, el cual intenta contemplar todos los posibles elementos que contenga una herramienta de mockup. En este punto se ve la principal diferencia con el enfoque buscado en este trabajo, el cual sugiere utilizar el estandar IFML, propuesto por el OMG.

Capítulo 3

Estudio de herramientas de Mockup

En este capítulo se brinda un panorama general del estudio realizado sobre herramientas de generación de mockups existentes en el mercado. El análisis involucró el estudio de un total de sesenta herramientas a partir de una serie de características definidas acorde a los intereses del proyecto. Para ello, fue necesario investigar la documentación de cada una de las herramientas, y en caso de ser posible, se realizaron diseños básicos para poder evaluar de primera mano sus funcionalidades.

A continuación se presentan las características de evaluación, un resumen de las herramientas evaluadas y un estudio comparativo de todas ellas.

3.1. Características a evaluar

Se presenta un listado con cada una de las características de interés a evaluar en cada herramienta de desarrollo de mockups, con una breve descripción.

Plataforma
Indica la plataforma que soporta la herramienta, por ejemplo: Plataforma web, Windows, Linux, Mac OS, etc.

Tipo de aplicación
Indica el tipo de aplicación para el cual se pueden generar diagramas. Por ejemplo: Aplicación web, de escritorio, móvil, etc.

Código abierto
Indica si la herramienta es de código abierto o no. En caso de serlo se le asignará el estado verde, si tiene alguna porción del código como abierto y otra no sería amarillo y rojo sería en caso de no serlo.

Eventos
Indica si la herramienta soporta eventos de interacción. Dependiendo de la cantidad de eventos soportados (click, hover, drag, etc) será la clasificación resultante.

Licenciamiento
Indica el tipo de licencia de la herramienta, los valores posibles son: verde si es gratuito, amarillo si tiene diferentes niveles de licenciamiento y rojo si es una herramienta paga exclusivamente.

Documentación
Indica el nivel de la documentación provista por la herramienta para el entendimiento del funcionamiento de la misma. Se le asignará verde en caso de que la documentación sea completa, actualizada y de fácil entendimiento. Si no posee alguna de las características antes nombrada se le asignará el color amarillo y en caso de que la documentación sea nula se le asignará el color rojo.

Formatos
Indica el formato en el que se pueden exportar los diagramas diseñados en la herramienta. Dado que se necesita procesar dicho diagrama, es necesario obtener información interpretable al momento de exportar el mockup, siendo entonces verde si el formato es legible e interpretable luego de la exportación, amarillo si el formato por más que sea legible, no es muy común y es difícil de interpretar (puede ser un formato propietario para el cual no existen especificaciones), y rojo si no se puede exportar el mockup en un formato manipulable.

Navegación
Indica el nivel de navegación entre elementos de los mockups desarrollados por la herramienta. En caso de que soporte navegación clara entre cualquier tipo

de elemento es verde, si no permite la navegación desde cualquier elemento del diagrama pero sí de algunos, entonces es amarillo, y en caso de no proveer manejo de navegación es rojo.

Componentes
Indica la variedad de componentes existentes dentro de la herramienta, como lo pueden ser, dropdowns, sliders, carouseles, canvas, mapas etc. Dependiendo de la variedad mencionada se clasificará usando el semáforo.

Usabilidad
Indica qué tan amigable es la herramienta con el usuario. Si la herramienta es muy clara y de fácil manejo es categorizada como verde. Si no es del todo intuitiva es amarilla, de lo contrario se categoriza como roja.

Asociación con datos
Indica si soporta asociación con fuentes de datos. En caso de poder asociar el diagrama con múltiples fuentes de datos, se categoriza como verde, si solo se puede realizar dicha asociación con algunas pocas se considera amarillo y si no es posible se considera rojo.

Genera código
Indica si se puede generar código a partir del diagrama diseñado y en qué tecnología se genera dicho código. En caso de poder generar código en varias tecnologías, se listan dichas tecnologías y se categoriza como verde, si solo se puede generar código para una tecnología se considera amarillo y si no es posible generar código se considera rojo.

Estilos personalizables
Indica si la herramienta permite personalizar estilos, ya sea colores de los elementos, fuentes de letras, tamaños de los componentes etc. Los valores posibles en esta clasificación serían: verde si permite una personalización total, amarillo si permite personalizar sólo algunos elementos y rojo si no permite ninguna personalización.

3.2. Estudio de herramientas

En esta sección se presenta un resumen del estudio de herramientas realizado. El estudio completo se puede consultar en el Anexo 2, en el cual se detalla cada herramienta brindando una breve descripción de sus características, y comentando los aspectos vistos en la Sección 3.1

Para visualizar este estudio se presenta la Tabla 3.3 que muestra una visión general del análisis realizado sobre cada herramienta, utilizando una notación específica como se indica a la Tabla 2.2.

Tipos de diagrama soportados	Metáfora del Semáforo
Soporta diagramas de páginas web y aplicaciones de escritorio. Soporta diagramas para dispositivos móviles.	Rojo: indicando que cumple un criterio insatisfactorio. Amarillo: indicando que cumple un criterio regular. Verde: indicando que cumple un criterio satisfactorio. Gris: indicando que no se puede determinar el criterio.

Tabla 3.2: Notación específica del estudio

En Tabla 3.3 las columnas se corresponden con el listado de características de la Sección 3.1 numeradas de la siguiente forma:

1. Tipo de aplicación que genera
2. Código abierto
3. Eventos
4. Licenciamiento
5. Documentación
6. Formatos que soporta
7. Navegación
8. Componentes
9. Usabilidad
10. Asociación con datos
11. Genera código
12. Estilos personalizables

	1	2	3	4	5	6	7	8	9	10	11	12
Antetype		✗	✓	✗	✓	✗	✓	✓	✓	✗	✗	✓
Atomic		✗	−	✗	−	−	✓	✓	✓	✗	−	✓
Axure		−	✓	✗	✓	✓	✓	−	✓	✗	−	✓
Balsamiq		✗	✗	✗	✓	✓	−	✓	✓	✗	✗	−
Easel												
EasyMocks												
filesquare		✗	−	−	−	✗	✗	✓	−	✗	✗	✗
Fireworks												
FlairBuilder		✗	✓	✗	−	✓	✓	✓	−	✗	−	✓
Flinto		✗	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗
FluidUI		✗	−	−	✓	−	−	✓	−	✗	−	✓
FluidIA												
ForeUI		✗	−	✗	✗	−	✓	−	−	✗	−	✓
FrameBox		✗	✗	✓	✗	✗	✗	−	−	✗	✗	✗
FrameLab25												
Gliffy		✗	✗	−	✓	−	−	✗	✓	✗	✗	✓
HotGloo		✗	−	✗	✓	✗	−	✓	✓	✗	✗	✓
InDesignCC												
Indigo Studio		✗	✓	✗	✓	−	−	✓	✓	✗	−	✓
InPresso Screens												
InVision		✗	−	−	✓	−	✗	✓	−	✗	−	✗
iPlotz		✗	✓	✓	✓	✗	✓	✓	✓	−	−	✓
iRise		✗	✓	✗	✗	−	✓	✓	−	✗	−	✓
Jumpchart		✗	−	−	−	−	✗	−	✗	✗	−	✗

	1	2	3	4	5	6	7	8	9	10	11	12
Just in mind Pro		✗	✓	✗	−	✗	✓	✓	✓	−	✗	✓
JustProto	●	●	●	●	●	●	●	●	●	●	●	●
Kony	●	●	●	●	●	●	●	●	●	●	●	●
Layout it!		✗	✗	✓	−	✗	✗	✓	✓	✗	−	✗
Lumzy		✗	−	−	−	✗	−	✓	✓	✗	✗	✗
MacAD WinAD	●	●	●	●	●	●	●	●	●	●	●	●
Maqetta		✓	✓	✓	−	−	✓	✓	−	✗	−	−
Marvel		✗	−	−	−	−	−	−	−	✗	−	✓
Mockabilly	●	●	●	●	●	●	●	●	●	●	●	●
Mockflow Wireframe Pro		✗	−	−	−	−	✓	✓	−	✗	−	−
Mocking Bird		✗	−	✗	✗	✗	−	✓	✓	✗	✗	−
Mockplus		✗	−	−	−	✗	−	✓	−	✗	✗	✓
Mockup Builder	●	✗	●	✗	−	●	●	●	●	✗	●	✓
Mockups Me		✗	−	✗	−	✗	✓	−	✓	✗	✗	✓
Moqups		✗	−	−	−	✗	−	✓	✓	✗	✗	✓
Napkee		✓	−	✓	−	−	−	−	−	✗	−	−
NinjaMock		✗	−	✓	−	✓	✓	✓	✓	✗	✓	✓
Omnigraffle	●	●	●	●	●	●	●	●	●	●	●	●
Pencil Project		✓	−	✓	✗	−	✓	✓	−	✗	−	−
Pidocco		✗	✓	✗	✓	✗	✓	−	✓	✗	−	−
Precursor		✗	✗	✓	−	✗	✗	−	−	✗	✗	✓
Proto.IO		✗	✓	✗	✓	✗	✓	✓	✓	✗	−	✓

	1	2	3	4	5	6	7	8	9	10	11	12
Protoshare												
QuirkTools Wires												
Savah												
Snapup												
Solidfy												
Uxpin												
Velositey												
Visio												
Visual Paradigm												
Weld												
Wireframe sketcher												
Wireframe.CC												
Wirefy												
Wiremagic												

Tabla 3.4: Resumen de análisis de herramientas

3.3. Preselección

Luego de analizar cada una de las herramientas, se realizó un estudio pormenorizado de las tres que presentan mejores características, siendo éstas candidatas para la realización de este proyecto. Muchas fueron descartadas debido a que su formato exportable no es lo suficientemente legible para nuestro propósitos. Por otro lado, hay herramientas que no poseen versión gratuita, así como otras características que se detallan a continuación, que no son favorables para un proyecto de este estilo.

3.3.1. Indigo

Indigo[20] es una herramienta que se puede utilizar tanto en sus versiones de escritorio para PC o para MAC. Esto es una gran ventaja, ya que se puede utilizar en dos de los sistemas operativos más usados en la actualidad. Si bien es una herramienta sin licenciamiento gratuito, algo a destacar es que proveen licencias de estudiante, solamente registrando al usuario con una cuenta de correo con extensión .edu. Se pueden realizar mockups tanto para aplicaciones web, móvil y de escritorio. La interfaz es muy intuitiva, tiene una paleta drag and drop a la derecha con una cantidad bastante buena de componentes para utilizar, a los cuales se le pueden personalizar los estilos, tamaños, etc. A dichos componentes se le pueden asignar una gran variedad de eventos e interacciones, con los cuales se puede navegar entre páginas o cambiar el estado de la página actual. En la Figura 3.1 se puede ver la interfaz de la herramienta y sus elementos de trabajo.

Por otro lado, permite exportar en formatos PDF, PNG y HTML, pero este último utiliza librerías propias para mostrar el diseño realizado. Además de los formatos a los que exporta, los proyectos se guardan en un formato propio utilizando la sintaxis de XML. Se genera un archivo .proj, que especifica el tipo del dispositivo y el tamaño del mismo, asi como tambien cual es la página inicial. Luego están los archivos .screen en los cuales se especifica cada página, sus elementos con las jerarquías definidas y los eventos e interacciones. El poder identificar los elementos, sus posiciones, interacciones, estilos, etc. dentro de los archivos del proyecto, es más de lo que se logró obtener de las demás herramientas analizadas, por esa razón es que Indigo se encuentra dentro de las herramientas con posibilidades de ser la elegida para continuar con el proyecto.

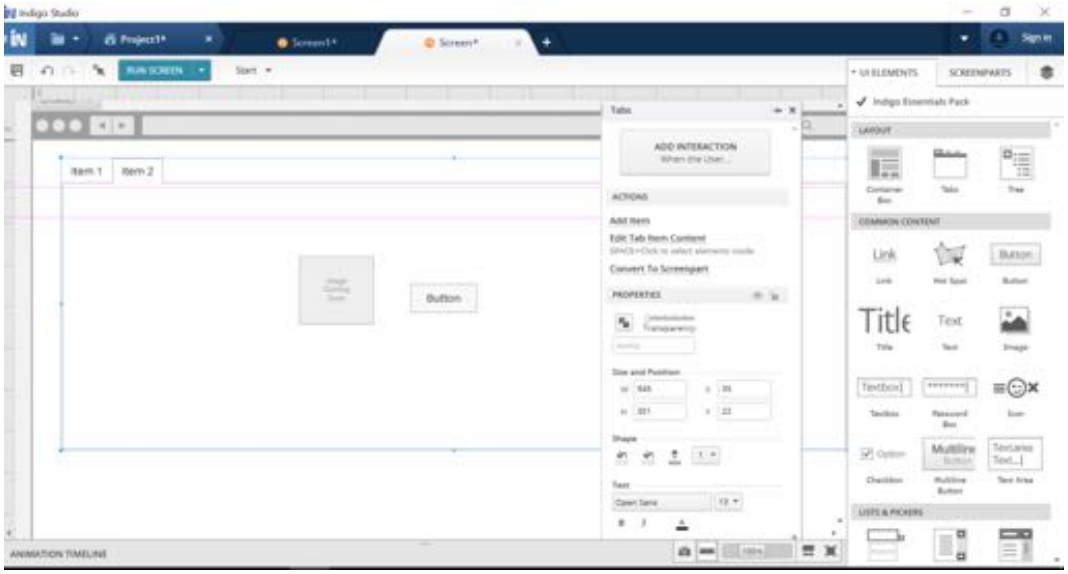


Figura 3.1: Interfaz de usuario de Indigo

3.3.2. Axure

Axure [4] es una herramienta muy utilizada en el ámbito del desarrollo de software. Es muy completa, permite realizar mockups para interfaces web, mobile y aplicaciones de escritorio. Su usabilidad es bastante buena, tiene menús intuitivos, sitemap del mockup, etc. La herramienta fue preseleccionada debido a su buena documentación, su soporte en línea, tiene entrenamiento online y tutoriales, además de una comunidad y foros de consulta. Otro aspecto tenido en cuenta, fue que la herramienta cuenta con la posibilidad de incluir librerías externas, con componentes predefinidos, por ejemplo componentes para mockups de iOS o Android, así como también permite incluir templates, armados por usuarios de la comunidad, con los layouts más usuales en las interfaces. A esto se le suma otra característica: los desarrolladores proveen una API, desarrollada en C#, la cual permite interactuar y manipular el archivo del mockup generado por la herramienta. Esto permite tener un control más preciso de los componentes, pudiendo manipular mediante código los componentes del mockup y usarlo a nuestra conveniencia, por ejemplo para generar un XML o un JSON con los contenidos del mockup. En la Figura 3.2 se muestra una captura de la interfaz de la aplicación.

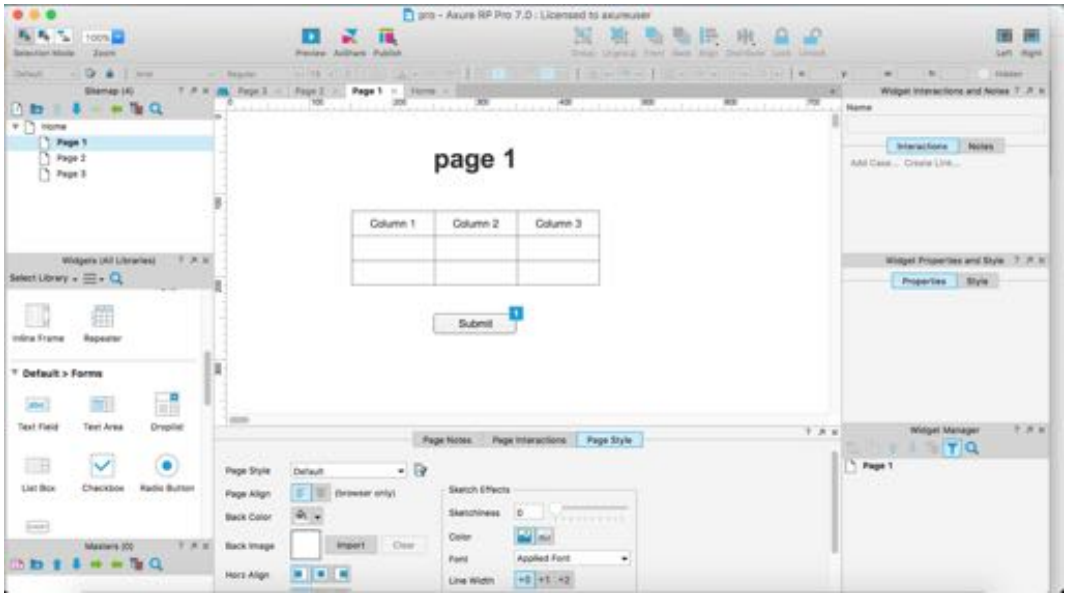


Figura 3.2: Interfaz de usuario de Axure

3.3.3. NinjaMock

NinjaMock [21] es una herramienta de plataforma web para el desarrollo de mockups. Posee una gran variedad de componentes para el desarrollo de diagramas, así como una versión gratuita cuya única limitación es la cantidad de diagramas por usuario (más específicamente, hasta tres diagramas por usuario). Posee una interfaz bastante intuitiva al usuario, permitiendo realizar diagramas de forma sencilla.

Figura 3.3: Interfaz de usuario de NinjaMock

Otro de los aspectos destacables es la forma exportación de los proyectos. Permite guardar el diagrama en lo que llaman "NinjaPackage", el cual básicamente es un archivo en formato JSON con la estructura del diagrama. Dentro de este archivo se observa una estructura bastante legible que contiene una lista con todas las páginas existentes en el diagrama, indicando su identificador, tipo de página, nombre, entre otros. A su vez cada página contiene una colección de elementos "hijos" que establecen una jerarquía de los elementos insertados en la página. Es importante mencionar que cada componente tiene su identificador propio y tipo de elemento, lo cual facilita el mapeo a realizar en etapas futuras de la realización del proyecto.

3.4. Conclusiones

Es de fundamental importancia tener en cuenta que la investigación expuesta en la sección anterior se desarrolla en marco de un proyecto académico, y por esta razón se tomó la decisión de descartar el uso de las herramientas preseleccionadas, dada su naturaleza de código propietario. Si bien cada una de las herramientas preseleccionadas posee varias características destacables, ninguna cumplía el requerimiento de que sean de código abierto. Además de este aspecto, otros influyeron en la decisión de cuál herramienta utilizar, los cuales se listan a continuación:

- **Indigo** no fue tenida en cuenta por no tener licenciamiento gratuito y no ser de código abierto.
- **Axure** fue descartada ya que es una herramienta sin un plan de licenciamiento gratuito, por lo que se estaría acotando el conjunto de usuarios que podrían utilizar la herramienta propuesta a futuro. Esto último sumado a que la herramienta no es de código abierto.
- **NinjaMock** tampoco es una herramienta de código abierto y además su uso es a través de una plataforma web, lo que generaba una dependencia a un servicio externo.

Es entonces donde surge la necesidad de buscar alternativas, por lo que se procedió al estudio de una nueva versión de **Pencil** [6], herramienta de código abierto que fue estudiada previamente. Al momento de realizar este estudio, la nueva versión de la herramienta se encontraba en su etapa beta de desarrollo

y sin fecha definida para su liberación final. Esto conlleva a que se debería trabajar con una versión incompleta y posiblemente inestable, liberada con fines de testing. Ya que era necesario esperar un tiempo indefinido para obtener una versión libre de errores y accesible para su modificación, se decidió no utilizar esta versión. Al mismo tiempo el costo de desarrollar una nueva funcionalidad dentro de una aplicación desconocida sería elevado, por lo que se decidió implementar una herramienta propia que cumpla con los requerimientos necesarios para este proyecto, tomando como referencia determinadas características de las herramientas estudiadas, como por ejemplo la interfaz de usuario, distribución de componentes, entre otros.

Capítulo 4

Contexto, Análisis y Diseño de la solución

En este capítulo se brinda el contexto de la realidad planteada, el análisis de la relación entre mockups e IFML así como también la relación establecida entre IFML y HTML, además de mostrar el diseño del enfoque propuesto en este proyecto.

4.1. Contexto

El trabajo realizado en este proyecto se basa fuertemente en un paradigma que propone usar los mockups como herramienta para especificar los elementos de una aplicación y las relaciones entre los mismos.

Como fue visto en el Capítulo 2, los mockups son un artefacto que de ser utilizados de forma correcta, pueden mejorar la eficiencia al momento de capturar nuevos requerimientos o de definir la estructura de una aplicación. Una de sus principales ventajas está dada por su nivel de comprensión tanto para el usuario final como para el desarrollador. Ya que los mockups especifican el contenido y navegación de una interfaz, podrían ser utilizados como una sintaxis alternativa de IFML, estableciendo una capa intermedia que permite abstraer ciertos aspectos de la interfaz, para en un paso posterior proceder con la generación automática del código fuente. Otra de las ventajas de utilizar esta capa intermedia, es que permite abstraerse de la tecnología que se utilice, tanto de entrada como de salida, es decir, se podría utilizar cualquier editor de

mockups, solamente construyendo un analizador que lo interprete en términos de IFML, así como también generar código para cualquier tecnología destino.

Para realizar esta abstracción, se tomó como base el estudio realizado por José Matías Rivero et al. en Mockup Driven Development[19] mencionado en la Sección 2.4, solamente cambiando el metamodelo de mockups que definieron sus autores, por el uso de IFML como capa de abstracción. El esquema que se propone en este proyecto es básicamente el mismo, pero como el objetivo de esta investigación es validar el enfoque, se tomará en cuenta solamente una herramienta de edición de mockups, junto con un motor de procesamiento que genera el modelo IFML. Ya que IFML se enfoca principalmente en la estructura y navegación de una interfaz, no se tomarán aspectos de estilos de los mockups, ya que no son representables en este lenguaje. Si bien se podrían extender los elementos del lenguaje para representar más atributos, se estaría alejando de la filosofía de IFML. Este lenguaje de modelado también incorpora referencia a datos, pero en esta investigación no se trabajará con ellos.

En la Figura 4.1 se muestra un diagrama del enfoque planteado, mostrando el flujo que se sigue, partiendo de un mockup hasta llegar a tener código generado, pasando por IFML.

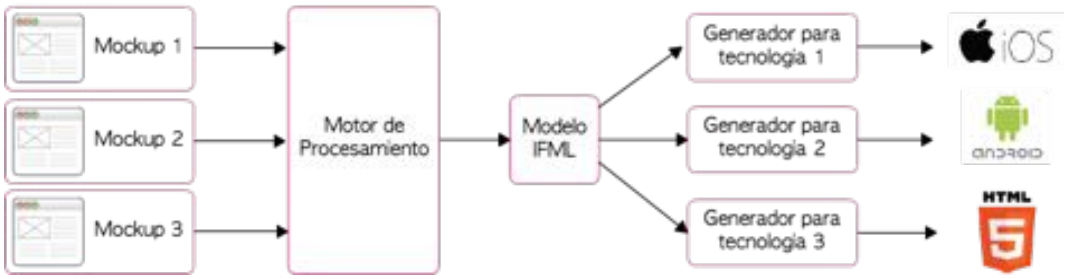


Figura 4.1: Esquema del enfoque propuesto

Al modelo IFML generado se le puede aplicar potencialmente cualquier transformación M2T para llevarlo a alguna tecnología concreta, pero como se mencionó anteriormente, esta investigación tiene como objetivo validar la aplicación de MDE en este contexto. Es por esto que se decidió continuar enfocándose exclusivamente en tecnologías web. Los mockups serán creados utilizando una herramienta desarrollada en el marco de este proyecto, para luego obtener su representación en IFML y su correspondiente código fuente.

La idea final es obtener a partir del mockup, un conjunto de páginas web, conteniendo sus diversos componentes y el manejo de eventos de navegación.

El código generado será en HTML, utilizando además librerías de estilo y funcionalidades.

4.2. Análisis del problema

Ya teniendo un esquema definido, surge la necesidad de plantear la relación existente entre sus componentes, lo que lleva a estudiar la relación entre un mockup e IFML y su correspondencia, así como también la relación y correspondencia entre los elementos propios de IFML y los elementos de HTML. El desarrollo completo de este análisis se verá en las siguientes secciones.

4.2.1. Relación entre componentes Mockup e IFML

En primer lugar es necesario definir una relación entre los elementos más generales de la web (que van a formar parte de la paleta de opciones de la herramienta generadora de mockups) y los elementos disponibles IFML. En el análisis se vio que algunos de los elementos más comunes en el contexto web, no son representables mediante IFML. Por lo que se decidió utilizar como prueba de concepto el mecanismo de extensión provisto. Para ello se eligieron cuatro elementos web; Imagen, Video, Texto y Botón; los cuales se crearon extendiendo el componente ViewElement de IFML y agregándole nuevas propiedades necesarias para cada caso.

La Tabla 4.2 muestra una recopilación de los componentes web comúnmente utilizados, estableciendo una correspondencia con los elementos previamente vistos de IFML y los agregados mediante el mecanismo de extensión.

Componente Mockup	**Componente IFML**
Página Representa una pagina completa del mockup.	ViewContainer
Layout Representa el diseño de la página (2 columnas, 3 columnas, 4 columnas).	ViewContainer
Texto Texto simple de una página (Títulos, párrafos, textos simple).	TextField (Extensión de ViewElement)

Componente Mockup	Componente IFML
Formulario Sección de una página que contiene contenido normal, código, controles (checkbox, radiobuttons, etc.) para luego ser procesados por el sitio.	Form
Botón de Submit El cual envia la informacion ingresada en un formulario para ser procesada luego.	SubmitEvent
Barra de búsqueda Formulario que permite buscar determinado texto, normalmente ubicado en el header.	Form
Input Utilizados para crear controles interactivos para formularios basados en la web, que reciban datos del usuario.	SimpleField
Boton Representa el boton simple dentro del diagrama.	Button (Extensión de ViewElement)
Imagen Representa una imagen dentro del diagrama.	Image (Extensión de ViewElement)
Tabla Representa una tabla común HTML.	List
Modal Caja de diálogo que se muestra en la página actual.	Window
Pestañas Representando distinto contenido dependiendo de la que esté seleccionada.	XOR ViewContainer
Barra lateral	ViewContainer

Componente Mockup	**Componente IFML**
Muestra diversas formas de información al costado de una interfaz de usuario.	
Video Representa un video dentro del diagrama.	Video (Extensión de ViewElement)
Enlace Representa un enlace hacia otra página del diagrama.	TextField (Extensión de ViewElement)

Tabla 4.2: Relación entre componentes Mockup - IFML

4.2.2. Relación entre componentes IFML y HTML

Dada la decisión de restringir los mockups a plataformas web, fue necesario estudiar los componentes del lenguaje HTML para la elaboración de páginas web, y establecer una correspondencia entre los mismos con los elementos de IFML. En la Tabla 4.4 se muestra a grandes rasgos la relación establecida entre cada componente de dichos lenguajes.

Componente IFML	Tag HTML
ViewContainer	`<body>` `<div>`

Landmarks ViewContainer
Contenido que será alcanzables desde cualquier ViewContainer.
	`<header>`
	`<footer>`
	
	`<div>`

XOR ViewContainer Si un ViewContainer es marcado como XOR significa que los ViewContainer hijos podrán ser mostrados de uno a la vez.	`[<ul>, <div>]`
Events	Eventos de navegación JavaScript.

Componente IFML	Tag HTML
Se tomarán en cuenta los eventos Submit event, Select event y Click event.	Selección de un elemento de una lista. Acción de click sobre un enlace.
NavigationFlow Normalmente ocurren como consecuencia de un evento.	<a> <button> Eventos de navegacion JavaScript
List Las listas van a tener contenido estático, por lo que se mapean con tags específicos.	<form> [,]
Form	<form>
SimpleField	<input> <textarea> <select> <checkbox>
Video **(extension de ViewElement)**	<video>
Image **(extension de ViewElement)**	<img>
Button **(extension de ViewElement)**	<button>
TextField **(extension de ViewElement)**	<p> <a>

Tabla 4.4: Relación entre componentes IFML - HTML

4.3. Diseño y Arquitectura de la Solución

En base al análisis realizado, se propone un diseño que contiene tres componentes principales:

- Herramienta web de Mockups propia
- Backend encargado de generar el modelo IFML
- Aplicación Acceleo que realiza la generación de código

En la Figura 4.2 se ilustra un diagrama con la arquitectura del sistema. La misma muestra los componentes mencionados, y el flujo señalado parte desde un mockup generado por el usuario final hasta la generación de código HTML.

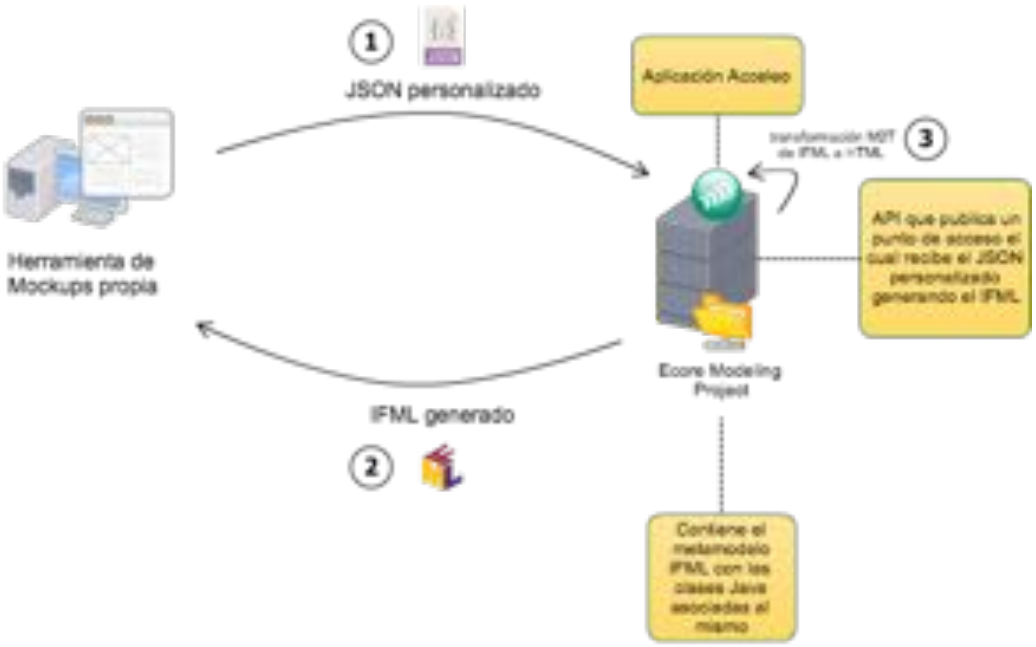


Figura 4.2: Arquitectura del sistema

Se optó por separar los componentes de diseño y procesamiento, por un lado para mantener modularizado el código del proyecto, así como también para aprovechar las mejores características de las tecnologías de cada área. Para el intercambio de datos se utiliza el formato JSON [22], ya que es un

formato muy usado y de fácil interpretación, para el cual es sencillo extraer y procesar los datos.

Herramienta web de Mockups propia

Como adelantamos en el Capítulo 3, se implementa una herramienta web que brinda la posibilidad de crear mockups utilizando los componentes más usuales de la web, permitiendo especificar la navegación entre ellos. El resultado es un diagrama persistido en formato JSON donde se especifica la estructura del mockup con su jerarquía e interacciones, el cual es enviado mediante un pedido HTTP [23] POST a la API del sistema, como indica el paso 1 de la Figura 4.2.

Backend encargado de generar el modelo IFML

Este componente contiene el metamodelo IFML con sus clases Java, asociadas a cada componente IFML. Posee una API que expone un punto de acceso, el cual recibe el JSON generado que representa el mockup y lo procesa utilizando las correspondencias vistas en la Subsección 4.2.1. Luego de dicho proceso, se transforman las clases generadas al formato XMI, almacenándolo en un archivo y se retornado a la herramienta para ser desplegado en pantalla, como indica el paso 2 de la Figura 4.2.

Aplicación Acceleo que realiza la generación de código

Se creó una aplicación Acceleo básica, la cual se encarga de ejecutar una transformación de modelo a texto (M2T) con el fin de generar código HTML a partir del modelo IFML, como indica el paso 3 de la Figura 4.2. Esta transformación aplica la relación establecida en la Subsección 4.2.1, logrando de esta forma generar la estructura básica de la web deseada.

Capítulo 5

Implementación del prototipo

En este capítulo se presentan los detalles de la implementación del prototipo que se realizó como prueba de concepto. Como se mencionó en el capítulo anterior en la Sección 4.3, se implementaron tres componentes, generando código HTML a partir de un mockup creado por el usuario. Se presentan las tecnologías utilizadas en cada componente, así como también una descripción general de su implementación, donde se presenta un ejemplo de ejecución de forma incremental, pasando por todos los componentes y mostrando resultados intermedios.

5.1. Herramienta de Mockups propia

Este componente permite al usuario crear mockups personalizados con el fin de generar el IFML correspondiente. De esta forma, una vez que el diagrama esté completo, la herramienta se comunica con una API mediante pedidos HTTP, para luego desplegar el resultado al usuario. Haciendo referencia al diagrama de la arquitectura, esta acción se puede ver en el paso 1 de la Figura 4.2.

Se tuvieron en cuenta varios aspectos de usabilidad web, diseñando un sistema que se adapte funcionalmente a las necesidades del usuario, brindándole así un sitio amigable, lo que permite disminuir el tiempo de aprendizaje de uso.

5.1.1. Tecnologías utilizadas

Para la implementación de este componente, se tomó la decisión de utilizar JavaScript [24] por varias razones; en particular porque es un lenguaje que es

ampliamente utilizado a nivel frontend, soportado por todos los navegadores que existen en el mercado y además es bastante rápido y sencillo implementar las funcionalidades que requiere este trabajo. Al utilizar JavaScript, se tiene el agregado de que hay miles de herramientas y librerías que brindan más características que facilitan el desarrollo. Algunas de esas tecnologías se utilizaron en esta implementación, entre las cuales se destacan las siguientes:

NodeJS - https://nodejs.org/

Node.js es un entorno de ejecución de código abierto, el cual permite ejecutar código JavaScript en un ámbito de backend. Es incluido en el proyecto ya que es una dependencia del gestor de tareas que detallaremos a continuación.

Grunt - http://gruntjs.com/

Grunt es una herramienta que permite ejecutar tareas utilizando JavaScript. Básicamente es una librería con la cual se pueden configurar y ejecutar tareas de manera automática, con lo que se ahorra tiempo al momento del desarrollo y despliegue de aplicaciones web. Las tareas a automatizar se especifican en un archivo de configuración llamado "Gruntfile", las cuales pueden ser por ejemplo: hacer un despliegue local de la aplicación, compilar hojas de estilo, ofuscar código etc. Además esta herramienta tiene la funcionalidad llamada "live reload", la cual observa el código buscando nuevos cambios y cuando detecta alguno refresca el navegador para que dichos cambios sean visibles.

AngularJS - https://angularjs.org/

AngularJS es un framework estructural, utilizado para crear aplicaciones web dinámicas. Implementa el patrón Model View Controller, y permite extender la syntaxis de HTML para expresar los componentes de una manera clara y concisa. Angular provee la funcionalidad de "data binding bidireccional" con el cual se sincroniza el modelo y la vista, de manera automática y en tiempo real, característica fundamental a la hora de realizar este trabajo. Otra característica importante que brinda este framework es que permite modularizar y separar el código, con el cual es mucho más cómodo desarrollar.

Bootstrap - https://getbootstrap.com/

Bootstrap es un framework frontend para diseñar sitios y aplicaciones web. Brinda un conjunto de hojas de estilo para los componentes más usuales de la web, con lo que facilita y agiliza el desarrollo. Bootstrap incluye también código JavaScript para ciertos componentes, como por ejemplo ventanas emergentes, páginas con pestañas, carruseles de imágenes etc. por lo que el desarrollador no tiene que preocuparse por desarrollarlos.

5.1.2. Descripción del componente

En esta sección se describen los detalles de la herramienta implementada en el marco de este proyecto. En primera instancia, se describen los aspectos visuales y funcionales de la herramienta, para luego presentar los aspectos de su implementación.

La herramienta de mockups es el frontend del proyecto, la parte visible donde el usuario realmente crea los diagramas que luego transformará en código. Es por esta razón que se hizo hincapié en brindar una herramienta amigable y fácil de utilizar, donde los elementos se explican por sí solos, sin requerir un manual de usuario o tour explicativo. La interfaz está dividida en 4 secciones,

las cuales son: *barra de acciones, paleta de componentes, lienzo y barra de propiedades*. La barra de acciones provee las funcionalidades de cargar o guardar un mockup, obtener su representación en formato JSON, así como también la funcionalidad principal que es la de obtener el modelo IFML en formato XMI.

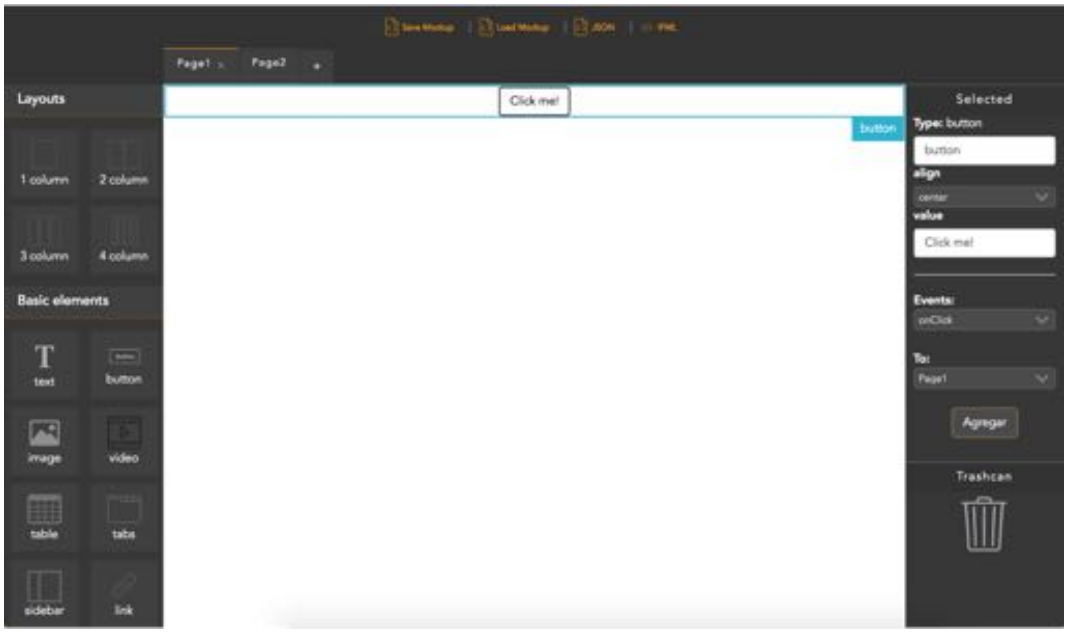


Figura 5.1: Editor de mockups

En la paleta de componentes, se lista el conjunto de componentes disponibles para utilizar en el diseño de un mockup. Ya que esta implementación se trata de una prueba de concepto, los componentes se limitaron a los más indispensables y utilizados en el desarrollo web, los cuales se detallan a continuación:

- columnas
- sidebar
- texto estático
- botones
- imagen
- video
- tabla
- tabs
- enlaces

- formularios

Los elementos de esta paleta son seleccionados y arrastrados hacia el lienzo, haciendo uso de la técnica "drag and drop" (arrastrar y soltar), que le permite al usuario construir su mockup de forma incremental. Al seleccionar un elemento en el lienzo, se listan las diferentes propiedades soportadas por el elemento seleccionado en la barra de propiedades, situada a la derecha. Estas propiedades van desde cambiar el identificador, alineación en la pantalla, hasta especificar eventos de interacciones con ellos.

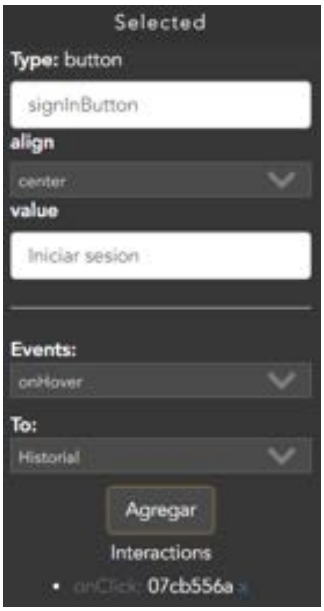


Figura 5.2: Barra de propiedades para un botón

El lienzo está dividido en pestañas, cada una de las cuales representa una página del mockup. El usuario puede agregar la cantidad de páginas que necesite para su diseño, simplemente agregando más pestañas, en donde se pueden agregar tanto interfaces completas como interfaces para ventanas emergentes.

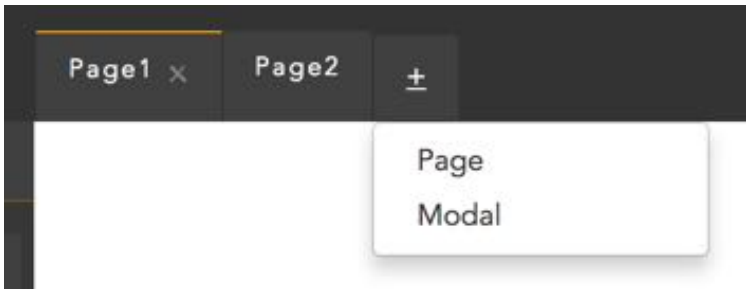


Figura 5.3: Agregar nueva página/modal

En lo que refiere a los detalles de implementación, podemos decir que la aplicación está construida en JavaScript, más precisamente utilizando el framework Angular, el cual fue utilizado tanto para el data binding, como para la importación de vistas parciales en la interfaz.

Angular permite implementar lo que se conoce como "Single Page Application" (SPA) [25], las cuales son aplicaciones web de una sola página, donde el contenido va siendo cargado a demanda, según las acciones del usuario y sin tener que recargar todo el sitio. La herramienta desarrollada en esta instancia, es una SPA, en donde se tienen vistas parciales, escritas en HTML, las cuales se van incluyendo a medida que el usuario crea el diagrama.

Además de las vistas, la aplicación consta también de un controlador JavaScript, el cual provee los manejadores para las diferentes acciones en la herramienta, ya sea para agregar una página, guardar y cargar mockups, etc. Para ir manteniendo la estructura del mockup se va creando un objeto JSON con la información de los elementos presentes en el lienzo. Esto es posible gracias al data binding provisto por Angular, el cual permite reflejar los cambios de la interfaz instantáneamente. Para poner todo en marcha, se utiliza la herramienta grunt que se mencionó en el apartado anterior, con el que se automatizan las tareas de inicialización y se levanta un servidor local, accesible desde la dirección `http://localhost:9000/` en el cual está disponible el editor de mockups para su uso.

Para ver más fácilmente la interacción de los componentes descritos en este Capítulo, se verá un ejemplo sencillo de mockup, que se irá completando durante todas las secciones del capítulo, mostrando los pasos realizados en cada componente.

Para comenzar con este ejemplo, se diseñará una interfaz sencilla en la cual hay dos páginas, una contiene un botón que al clickearlo va hacia la otra página, mostrando un texto estático.

El diseño en el editor se puede ver en la Figura 5.4; en la izquierda se aprecia el diseño de la primer página, y en la derecha el diseño de la página de destino.

En la barra de propiedades del botón, se especifica el evento *onClick*, con destino en la *Page 2*.

El siguiente paso en el flujo del mockup, es generar el modelo IFML correspondiente , para posteriormente generar el código HTML. Para lograr esto, se selecciona la opción *Get IFML* desde la barra de acciones de la herramienta,

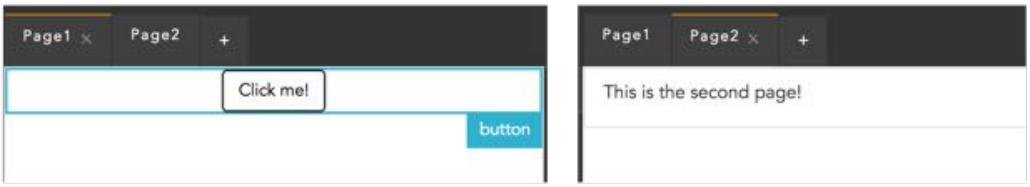


Figura 5.4: Mockup del ejemplo

el cual hace un pedido HTTP POST conteniendo el JSON generado, a la API publicada por el backend, donde se genera el modelo IFML, siendo retornado al usuario como respuesta al pedido.

5.2. Backend

El backend está encargado del manejo y procesamiento de la información recibida por la herramienta de mockups, exponiendo mediante una API los puntos de acceso para acceder sus las funcionalidades. En pocas palabras, este componente recibe el JSON que representa el mockup, lo procesa y genera el XMI correspondiente al modelo IFML, el cual es retornado a la herramienta de mockups para ser presentado al usuario. Haciendo referencia al diagrama de arquitectura de la Figura 4.2, el flujo anterior está dado por los puntos 1 y 2 respectivamente.

5.2.1. Tecnologías utilizadas

La implementación de este componente se realizó en Java [26] y se utilizó el IDE Eclipse [27]. A su vez, se utilizaron varias tecnologías asociadas a las anteriores, entre las cuales se destacan las siguientes:

Spark - http://sparkjava.com/

Micro framework Java que brinda la posibilidad de crear aplicaciones web de forma sencilla. Se enfoca en ser lo más simple y directo posible, sin la necesidad de realizar tareas de configuración tediosas. Spark es normalmente utilizado para crear APIs REST publicando diversos endpoint de forma directa.

Eclipse Modeling Framework - https://www.eclipse.org/modeling/emf/

Framework de modelado y de generación de código para construir herramientas y otras aplicaciones basadas en un modelo de datos estructurado. A partir de una especificación de modelo en XMI, EMF proporciona herramientas y soporte de ejecución para producir un conjunto de clases Java para el modelo, junto con un conjunto de clases de adaptadores que permiten la visualización y edición de comandos basado en el modelo, y un editor básico. Para el proyecto a implementar, se utilizó el metamodelo de IFML en formato ecore (EMF core), generando de esta forma las clases correspondientes al mismo para posterior utilización.

Gson - https://github.com/google/gson/

Gson es una librería Java que puede ser utilizada para convertir objetos Java en su representación JSON, asi como tambien puede ser utilizada para convertir un string JSON en un objeto Java equivalente. Gson puede trabajar con objetos propios u objetos preexistentes para los cuales no es necesario tener el código fuente.

5.2.2. Descripción del componente

La decisión de usar el IDE Eclipse e implementar el backend en Java, se basó principalmente en el hecho de poder aprovechar todos los beneficios que provee

el Eclipse Modeling Framework (EMF) en torno a MDE, en particular a la hora de trabajar con modelos. A su vez, dentro de la variedad de metamodelos de IFML que están disponibles desde el sitio web del lenguaje [28], existe un metamodelo expresado formato ecore, justamente el formato que utiliza EMF para manipular los modelos.

Los beneficios de EMF son provistos por los complementos (plugins) incorporados para el manejo de metamodelos en formato Ecore, junto con los complementos para el manejo de modelos en formato XMI, que es el utilizado por IFML para el intercambio y serialización de modelos.

La implementación del backend se dividió en dos proyectos Java: uno denominado "ifml" y otro denominado "api". En el proyecto "ifml" se encuentra el metamodelo IFML en formato ecore, en donde también se realizó la extensión del metamodelo, agregando los nuevos elementos que fueron detallados en la Subsección 4.2.1. En la Figura 5.5 se pueden apreciar (utilizando el visor de modelos provisto por el EMF) los elementos que pertenecen al paquete extensión de IFML, así como también los cuatro elementos agregados.

Cuando se tuvo lista la extensión, se procedió a generar las clases Java correspondientes a los elementos del metamodelo. El código generado consiste en interfaces e implementaciones concretas para cada elemento, tanto del paquete core como del paquete extensions. A su vez, se generan también dos clases, llamadas `CoreFactory` y `ExtensionsFactory`, que poseen métodos para crear cada uno de los objetos definidos, siendo el objeto principal el `IFMLModel` (raíz de IFML).

Tanto las interfaces generadas, como sus implementaciones, contienen métodos *getter* y *setter* para cada uno de sus atributos, lo cual te permite obtener y modificar sus valores.

En el proyecto "api" se publica un endpoint utilizando Spark, el cual recibe un archivo JSON del componente web con la estructura de un mockup. Este archivo se deserializa utilizando Gson, creando un objeto java de tipo `Mockup`, clase creada para este proyecto, la cual posee una estructura de listas recursivas para poder representar mockups de múltiples niveles.

Para generar el modelo IFML a retornar, se utilizan las clases generadas en el proyecto "ifml", por lo que es necesario que este proyecto se encuentre en el classpath del proyecto "api". De esta forma quedan disponibles todas las clases que representan el metamodelo IFML y sus componentes. Para lograr esto, es necesario crear un objeto `IFMLModel` y agregar los elementos de IFML

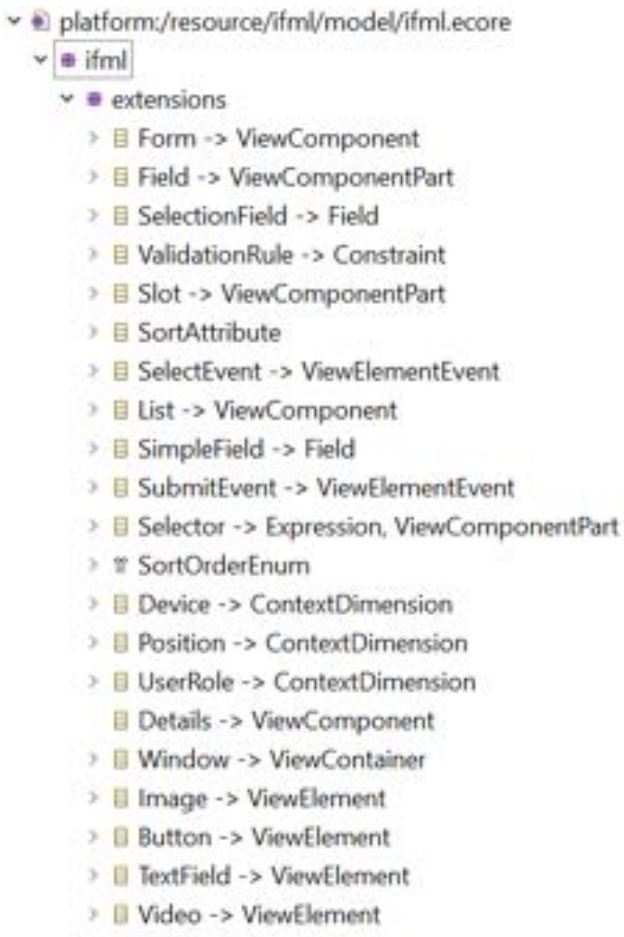


Figura 5.5: Vista de los elementos de extensión de IFML

correspondientes, cada uno con sus atributos, así como las relaciones entre ellos. El proceso está implementado como un algoritmo recursivo sobre el objeto `Mockup` obtenido, que va creando los objetos java de IFML utilizando las fábricas provistas, siguiendo las relaciones definidas en la Subsección 4.2.2.

Una vez obtenido el objeto `IFMLModel` con la estructura del mockup completa, se procede a generar el XMI a retornar al componente web, el cual es generado utilizando métodos propios de los plugins que provee EMF para el manejo de Ecore. Luego de generado, se envía en formato `String` como respuesta del servicio.

A continuación detalla el ejemplo introducido en la sección anterior, en relación a este componente. El JSON que es recibido por la API tiene la siguiente estructura principal:

```
{
  "name": "Mockup",
  "pages": [ ... ]
}
```

La key "name" provee el nombre del mockup, el cual será el nombre del modelo IFML resultante, representado por la clase `IFMLModel`. La key "pages", provee una lista con todas las páginas que componen el mockup, sobre la cual se realiza una recursión para generar todos los elementos que la componen.

En particular, se crea un `InteractionFlowModel` junto con una lista de `InteractionFlowModelElement`, la cual contendrá todos los elementos del modelo.

A continuación se muestra la estructura de la primer página definida en el JSON:

```
{
  "id": "8c74de2f",
  "type": "page",
  "name": "Page1",
  "properties": {
        "default": false,
        "landmark": false
  },
  "children": []
}
```

Las páginas serán creadas como `ViewContainer` y agregadas a la lista de `InteractionFlowModelElement`. A cada `ViewContainer` se le asigna su nombre y sus propiedades, utilizando los métodos provistos por las clases Java generadas. Se puede ver que la representación de la página en el objeto JSON tiene una lista bajo la key "children", la cual contiene los elementos a agregar dentro del `ViewContainer` generado, siendo posible que sean nuevos `ViewContainer`. En este caso particular, la primer página contiene solamente un botón, con la siguiente estructura en el JSON:

```
{
  "type": "button",
  "id": "2e998980",
  "name": "button",
  "properties": {
        "align": "center",
        "value": "Click me!"
  },
  "events": [
        {
          "type": "onClick",
          "link": "6476c239"
        }
  ]
}
```

El botón será creado como un `Button`, elemento que fue generado mediante extensión del elemento `ViewElement`, al cual se le asignarán las propiedades definidas en el mockup. A su vez en el JSON se especifica, bajo la key 'events', una lista de eventos asociados al botón. Por cada uno de ellos se creará un `ViewElementEvent`, asi como un `NavigationFlow` que se asocia al evento, al cual se le asigna como destino la página con el id seteado en la key 'link' del evento. La segunda página tiene la misma estructura, pero en vez de tener un botón dentro de sus hijos, tiene un texto, el cual será creado utilizando un `TextField`; elemento generado mediante el mecanismo de extensión que provee IFML.

5.3. Aplicación Acceleo

Este componente tiene la responsabilidad de generar código HTML a partir de un modelo IFML. Posee una transformación principal (paso 3 de la Figura 4.2), que por cada elemento soportado por nuestra herramienta genera un tag HTML correspondiente, logrando de esta forma generar la estructura general de la web diseñada.

5.3.1. Tecnologías utilizadas

Para la implementación de este componente, se utilizó Acceleo ya que es una herramienta que posee una implementación del estándar de transforma-

ciones Model to Text (M2T) del OMG.

Acceleo - https://www.acceleo.org/

Acceleo es una herramienta basada en Eclipse utilizada para la generación de código, la cual es una implementación del estándar de transformaciones Model to Text del OMG. Herramienta bastante fácil de utilizar, que posee un editor propio, con autocompletado, ayuda sintáctica, detección de errores, así como un debugger.

Bootstrap

En este componente también se utiliza el framework bootstrap, detallado en la Subsección 5.1.1, con el fin de representar ciertos componentes agregando unas pocas líneas de código, como por ejemplo para representar el modal.

5.3.2. Descripción del componente

Al igual que el backend, este componente se implementó en Eclipse. Para ello se creó un proyecto de Acceleo utilizando el wizard provisto, el cual solicita ingresar el metamodelo a utilizar; en este caso será el metamodelo de IFML utilizado en el backend. Este proyecto se nombró "ifml2html" y consta de dos elementos principales que permiten completar la generacion de codigo HTML:

- Clase de nombre `Generate.java`, la cual es generada por Acceleo y sirve como punto de entrada para el módulo de generación.
- Módulo de generación de nombre `generate.mtl`, en el cual se define la transformación M2T, la cual, a partir de un conjunto de reglas escritas utilizando la sintaxis propia de Acceleo, se describe como el IFML origen será transformado en código HTML.

Dentro del módulo que implementa la transformación, se define un template correspondiente a cada componente IFML, el cual determina el código HTML que representara al mismo, el cual está fuertemente vinculado al análisis realizado en la Subsección 4.2.2

Continuando con el ejemplo de las secciones anteriores, el XMI que representa al modelo IFML generado por el backend tiene la siguiente estructura:

```
<?xml version="1.0" encoding="ASCII"?>
<core:IFMLModel name="Mockup">
    <interactionFlowModel name="Pages">

        <interactionFlowModelElements xsi:type="core:ViewContainer" name="Page1">
            <viewElements xsi:type="ext:Button" name="button"
                          ButtonText="Click me!" lign="center">
                <viewElementEvents>
                    <navigationFlows
                    srcInteractionFlowElement=".../@viewElementEvents.0"
                    trgtInteractionFlowElement=".../@interactionFlowModelElements.1"/>
                </viewElementEvents>
            </viewElements>
        </interactionFlowModelElements>

        <interactionFlowModelElements xsi:type="core:ViewContainer" name="Page2"
                                      inInteractionFlows="../@navigationFlows.0">
            <viewElements xsi:type="ext:TextField" name="text"
                          FontSize="15" TextFieldText="This is the second page!" />
        </interactionFlowModelElements>

    </interactionFlowModel>
</core:IFMLModel>
```

Figura 5.6: XMI del Modelo IFML generado

Se puede ver que el modelo está conformado por un tag principal `IFMLModel`, el cual contiene un tag `InteractionFlowModel`. Bajo este último tag es que están contenidas las dos páginas que son representadas en el ejemplo:

- La primer página se especifica utilizando el tag `InteractionFlowModelElement` de tipo `ViewContainer`, en el cual se define el nombre de la página en cuestión, siendo en este caso 'Page1'. Siguiendo en la jerarquía de elementos, se encuentra un `ViewElement` de tipo `Button`, el cual posee dos propiedades personalizadas, ButtonText y Align, además de su nombre. Dentro de la lista de eventos asociados a este elemento, se distingue un `NavigationFlow`, donde el atributo trgtInteractionFlowElement indica la página de destino, con lo cual se puede determinar hacia que pagina redireccionar al momento de presionar el botón.
- La segunda página también se representa como un `InteractionFlowModelElement` de tipo `ViewContainer`. La diferencia es que en este caso se tiene un atributo que indica que el `InteractionFlow` de entrada es el `NavigationFlow` definido en la primer página. Además, bajo el `ViewContainer`, se tiene un `ViewElement`

de tipo `TextField` el cual posee dos propiedades personalizadas, FontSize y TextFieldText, además del nombre del elemento.

Con el objetivo de mostrar cómo Acceleo define las transformaciones, a modo de ejemplo se presenta cómo fueron implementadas para dos de los elementos principales que conforman el IFML del ejemplo. El primer template que vamos a analizar es el generador de HTML a partir de un elemento IFML Button de entrada.

```
[template public generate(btn: Button)]

  [if (btn.eAllContents()->isEmpty())]
    <button class="btn btn-primary button align-[ btn.Align /]" align="[ btn.Align /]">
      [ btn.ButtonText /]
    </button>
  [else]
    [for (content : InteractionFlowModelElement | btn.eAllContents())]
      [if (content.oclIsTypeOf(NavigationFlow))]
        [if (..trgtInteractionFlowElement.oclIsTypeOf(Window) and
             ..trgtInteractionFlowElement.oclAsType(Window).isModal)]
              <a data-toggle="modal" href="[ ..trgtInteractionFlowElement.name /].html"
                 class="btn btn-primary modal-trigger" align="[ btn.Align /]">
                  [ btn.ButtonText /]
              </a>
        [else]
            <a href="[..trgtInteractionFlowElement.name /].html"
                class="btn btn-primary" align="[ btn.Align /]">
                    [ btn.ButtonText /]
            </a>
        [/if]
      [/if]
    [/for]
  [/if]
[/template]
```

Figura 5.7: Template de la transformacion asociado a un Button

Esta transformación distingue tres casos fundamentales en base al elemento que recibe:

1. Si el Button de entrada no tiene una referencia de navegación, simplemente se genera un botón que no realiza ninguna acción.
2. Si el Button de entrada tiene un `NavigationFlow` con un destino específico, se genera un botón, el cual al ser clickeado, redirecciona hacia una nueva página.
3. Si el Button tiene un `NavigationFlow` cuyo destino es un elemento de tipo `Window` que representa un modal, se genera un botón HTML que al ser clickeado abre un modal sin redireccionar hacia otra página.

Este ejemplo recae en el caso dos, por lo que se genera un botón que redirecciona hacia una nueva página.

De manera similar, se define un generador de HTML a partir de un elemento IFML `TextField`:

```
[template public generate(tf: TextField)]
  [if (tf.eAllContents()->isEmpty())]
    <p style="font-size:[ tf.FontSize /]px">[ tf.TextFieldText /]</p>
  [else]
    [for (content : InteractionFlowModelElement | tf.eAllContents())]
      [if (content.oclIsTypeOf(NavigationFlow))]
        <a style="font-size:[ tf.FontSize /]px" href='[ ..trgtInteractionFlowElement.name /].html'>
          [tf.TextFieldText/]
        </a>
      [/if]
    [/for]
  [/if]
[/template]
```

Figura 5.8: Template de la transformacion asociado a un TextField

Esta transformación distingue dos casos fundamentales en base al elemento que recibe:

1. Si el `TextField` no contiene ningún elemento en su estructura, indica que es solamente texto, generando de esta forma una tag `<p>` que genera un párrafo con el texto indicado.
2. Si el `TextField` contiene un `NavigationFlow` con un destino específico, el texto generado será una tag `<a>` que representa un link con referencia a otra página.

Por último y una vez finalizada la ejecución de las transformaciones, se genera un directorio con el nombre del modelo IFML como se muestra en la Figura 5.9, el cual contendrá un archivo HTML por cada `InteractionFlowModelElement` de tipo `ViewContainer`, los cuales determinan una página completa. Cada archivo generado tiene el código HTML correspondiente a cada página, incluida la navegación entre las mismas, de esta forma se obtiene la interfaz completa correspondiente al mockup.

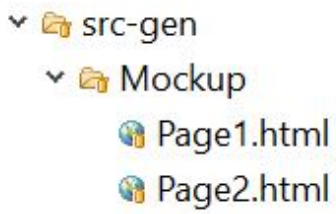


Figura 5.9: Archivos Generados

5.4. Pruebas del prototipo basados en ejemplos del libro

Se realizó una prueba de concepto en base a diversos ejemplos extraídos del libro "Interaction Flow Modeling Language" [7] con el fin de realizar un estudio comparativo entre los modelos IFML que genera el sistema de este proyecto, con los modelos presentados en el libro, analizando cada componente en base a su correspondencia, atributos, etc.

5.4.1. Ejemplo de ViewContainers

El primer ejemplo IFML seleccionado, muestra una estructura básica de distintos tipos de ViewContainers, como se aprecia en la Figura 5.10.

Figura 5.10: Ejemplo ViewContainers

A la izquierda se detalla el ejemplo extraído del libro, donde se pueden apreciar cuatro `ViewContainer`, el cual uno de ellos tiene la característica de ser XOR; esto significa que los componentes "View 1" y "View 2" no pueden aparecer a la vez en la interfaz. Para replicar este comportamiento en el editor de mockups desarrollado, se crea un mockup con 4 contenedores, donde uno es un componente "tab" con dos vistas. Esto nos permite especificar la naturaleza XOR entre las vistas "View 1" y "View 2" ya que en un componente "tab" las vistas que lo componen no aparecen a la vez.

El modelo IFML resultante en este editor de mockups se presenta en la Figura 5.11, donde se observa una clara correspondencia entre los elementos del modelo y los elementos del ejemplo.

Figura 5.11: Modelo IFML resultante

5.4.2. Multicriteria Search Pattern

El segundo ejemplo seleccionado, muestra un formulario de búsqueda con distintos tipos de filtro disponible. En la Figura 5.12 se puede ver el diagrama de Mockup y el modelo IFML que representa al mismo.

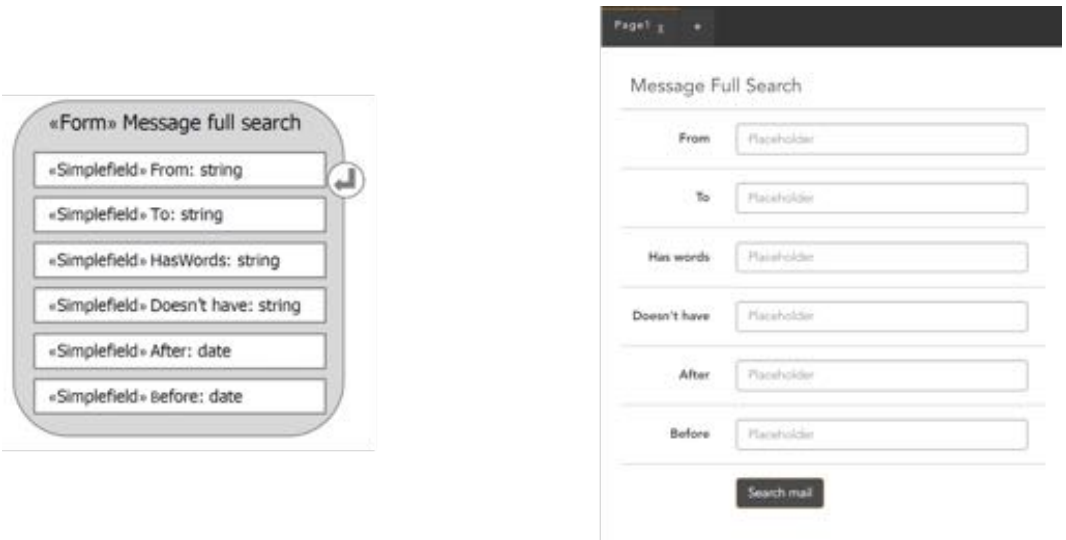


Figura 5.12: Mockup del formulario en la herramienta

A la izquierda se detalla el ejemplo extraído del libro, donde se pueden apreciar un `Form`, el cual contiene seis `SimpleFields` y un `SubmitEvent`. A la derecha se encuentra el mockup resultante de diseñar el ejemplo utilizando el editor de mockups.

El XMI del modelo IFML que es retornado por el backend se presenta en la Figura 5.13, donde se observa una clara correspondencia entre los elementos del modelo y los elementos del ejemplo.

Figura 5.13: Modelo IFML resultante

Capítulo 6

Caso de estudio

En este capítulo se presenta un caso de estudio para el cual se seleccionó un sitio web específico, con el fin de completar todos los pasos del ciclo de ejecución, desde crear el mockup, obtener el IFML con su estructura y generar el HTML correspondiente.

6.1. Introducción

Una vez definido el enfoque e implementado el prototipo, es necesario definir un caso de estudio completo, para poder validar la solución propuesta en un contexto real.

Antes de proseguir con el detalle del caso de estudio, es conveniente recordar que IFML se enfoca principalmente en la estructura y comportamiento de una interfaz. Dicho esto, no se toman en cuenta aspectos de estilos, tanto a la hora de realizar el mockup, como cuando se genera el código HTML, ya que no son representables en este lenguaje. Sin embargo, para este caso de estudio, se implementó una hoja de estilos, para ser aplicada luego de la transformación y por fuera del flujo de MDE, con el fin de que los resultados sean más cercanos a la interfaz final.

Otro punto a destacar es que, como consecuencia de extender el metamodelo de IFML (como se vio en el Capítulo 4), no fue posible representar las interfaces de este caso de estudio utilizando el modelo gráfico del IFML, ya que sería necesario extender también su sintaxis concreta y eso no aportaba valor a este proyecto. Es por esto que en su lugar se mostrará el árbol provisto por el editor de modelos de Eclipse.

6.2. YouTube

El sitio elegido para este caso de estudio es YouTube [29], el cual es un servicio gratuito de acceso compartido a videos en internet. Se seleccionó un subconjunto de páginas a ser recreadas en nuestra herramienta, a modo de cubrir diversos componentes de la web:

- **Inicio**: pagina principal del dominio YouTube. Contiene una vista general de todo el sitio.
- **Iniciar de Sesión**: refiere a la página de inicio de sesión del sitio.
- **Reproductor**: es la página en la que se muestra un video particular, en la cual se da la opción de reproducirlo, ver y realizar comentarios, así como también provee un conjunto de videos relacionados.
- **Menú de usuario**: es la pagina de inicio, pero una vez que el usuario ha iniciado sesión, por lo que está personalizada para dicho usuario.
- **Mi canal**: página que una perspectiva general del usuario, mostrando sus videos, suscriptores, listas de reproducción, entre otros.
- **Historial de videos**: es la página que muestra tanto el historial de reproducción, el historial de búsqueda y el historial de comentarios.

En la Figura 6.1 se muestra el árbol que representa el modelo IFML generado a partir del mockup de Youtube, utilizando el editor de modelos de Eclipse, donde se visualizan todas las páginas del mockup.

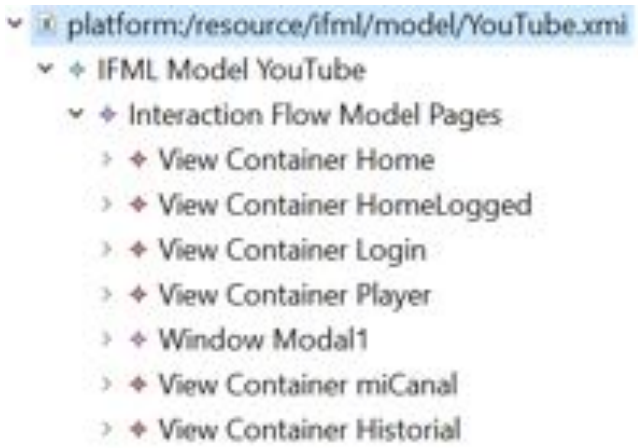


Figura 6.1: Arbol de elementos IFML generados

En esta figura se pueden observar siete páginas contenidas en la lista de `Interaction Flow Model Pages`. Seis de ellas son instancias de

`ViewContainer`, dentro de las cuales se definen los elementos propios de cada página. Además, está incluida una ventana modal, la cual es una instancia de `Window` ya que, dada su naturaleza de "modal", cuando se ejecuta abre una nueva ventana, privando de interacción a los elementos detrás de ella.

A modo de presentación, se detalla el proceso para dos de las páginas anteriores; la página del "Reproductor" y la página de "Mi Canal", ya que con ellas se cubren casi la totalidad de los componentes que utiliza el mockup.

6.2.1. Reproductor

La primer página seleccionada para el análisis es la pantalla de reproducción, la cual contiene diversas secciones y un conjunto de componentes variado. La Figura 6.2 muestra una captura de pantalla de la página web como lo es actualmente.

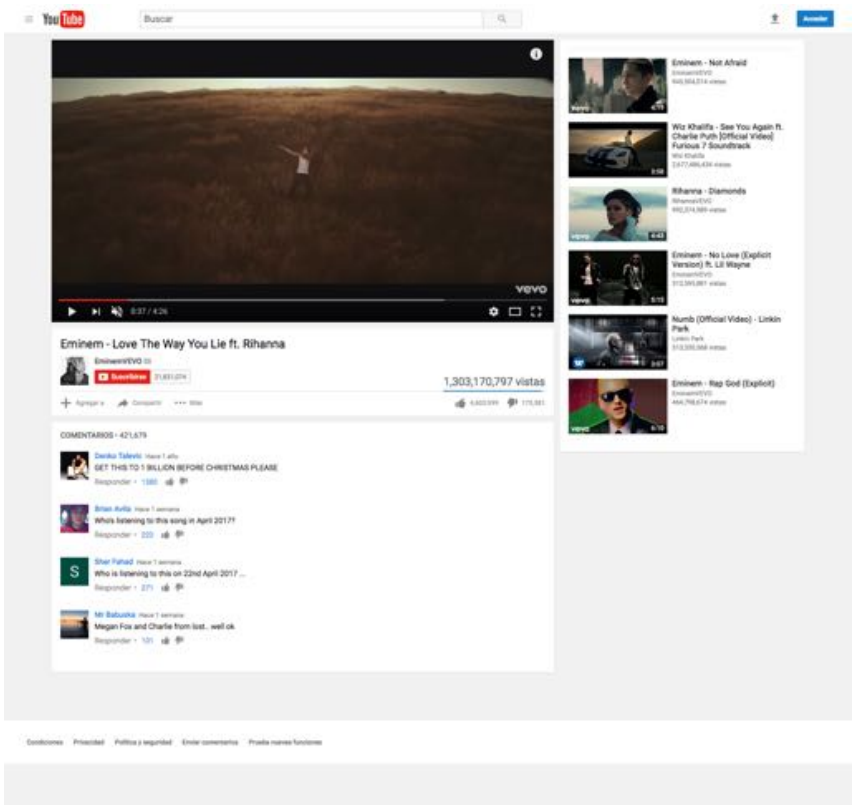


Figura 6.2: Página de Reproductor

Correspondencia de elementos

En la Figura 6.3 se muestra la representación de la página de reproducción en la herramienta de mockups desarrollada.

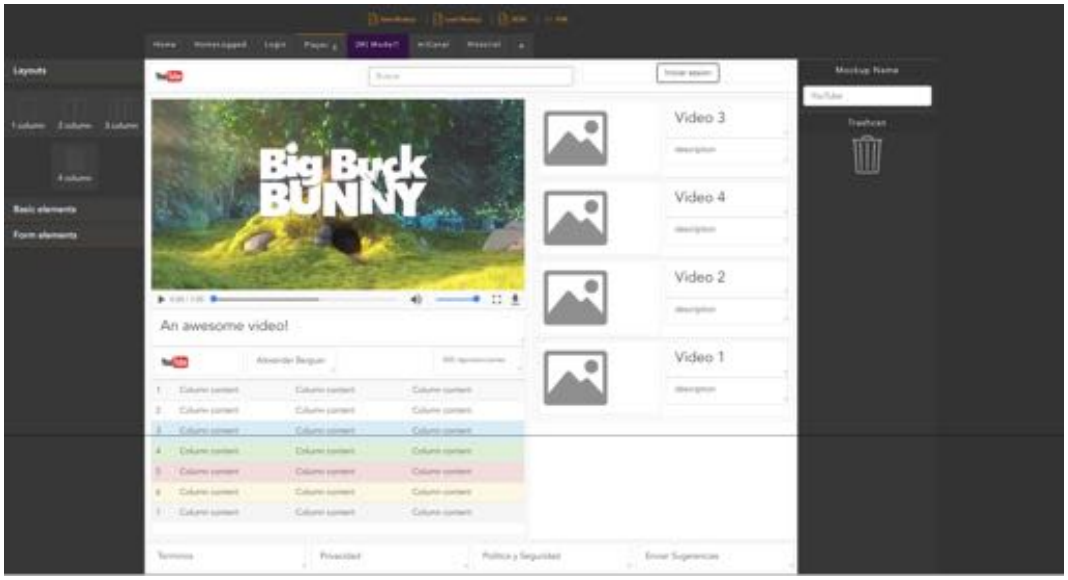


Figura 6.3: Mockup de la página de reproducción

Con el fin de tener claridad con los resultados obtenidos, se divide el mockup en secciones, mostrando la correspondencia entre el diagrama y el modelo IFML generado.

Header

Esta sección se encuentra en muchas de las páginas recreadas. Muestra el logo del sitio, un formulario de búsqueda, y un botón para iniciar sesión dentro del mismo. En la Figura 6.4 se muestra la representación del header en el mockup diseñado y su correspondencia con los elementos.

Cada elemento del header se representa de la siguiente forma:

- **Logo**: se representa como un elemento `Image`, el cual es una de las extensiones creadas, como se detalló en el Capítulo 4.
- **Formulario de búsqueda**: se corresponde con un elemento `Form` que contiene un `SimpleField`, el cual representa al cuadro de texto.
- **Botón de inicio de sesión**: representado por un elemento `Button`, otros de los elementos de la extensión creada. Contiene un `Event` con un `NavigationFlow`, el cual determina la acción del botón; en este caso particular el botón redirecciona a la página de login.

Figura 6.4: Correspondencia Header - IFML

A continuación se puede ver un extracto del JSON generado correspondiente al botón representado en el header.

```
{
    "type": "button",
    "id": "c6b30605",
    "name": "button",
    "properties": {
      "align": "center",
      "value": "Iniciar sesion"
    },
    "events": [
      {
        "type": "onClick",
        "link": "07cb556a"
      }
    ]
}
```

En su estructura se puede ver: el tipo de objeto, su identificador, el nombre y una lista de propiedades y eventos asociados al botón. Dentro de las propiedades se puede ver el texto del botón y su alineación dentro del mockup. También contiene una lista de eventos, que en este caso posee uno de tipo `onClick`, donde el link representa el id de la pagina destino.

Contenido

El contenido de la página está compuesto por el video que el usuario quiere reproducir, junto con información adicional debajo. Además contiene una tabla que representa los comentarios sobre el video. Del lado derecho se encuentra la sección de videos sugeridos, la cual se encarga de mostrar un conjunto de videos recomendados similares al que el usuario está accediendo.

Figura 6.5: Correspondencia Contenido - IFML

Cada elemento se representa de la siguiente forma:

- **Video**: se representa como un elemento `Video`, el cual es una extensión de un `ViewElement` con información adicional para poder representar un video en una página.
- **Información adicional**: se encuentra debajo del video y está conformado por un elemento `Image`, que muestra la foto del usuario que subió el video y dos elementos `TextField`, los cuales despliegan el nombre del usuario y la cantidad de reproducciones del video.
- **Comentarios**: representado por el elemento `List` de IFML.
- **Videos sugeridos**: se repite varias veces en el diagrama, y está definido por una imagen que se representa como un elemento `Image`, un título y una descripción que se representan con un `TextField`.

Footer

De la misma forma que el header, el footer se encuentra en varias páginas recreadas. El mismo muestra un conjunto de links dentro de la página como se puede ver en la Figura 6.6.

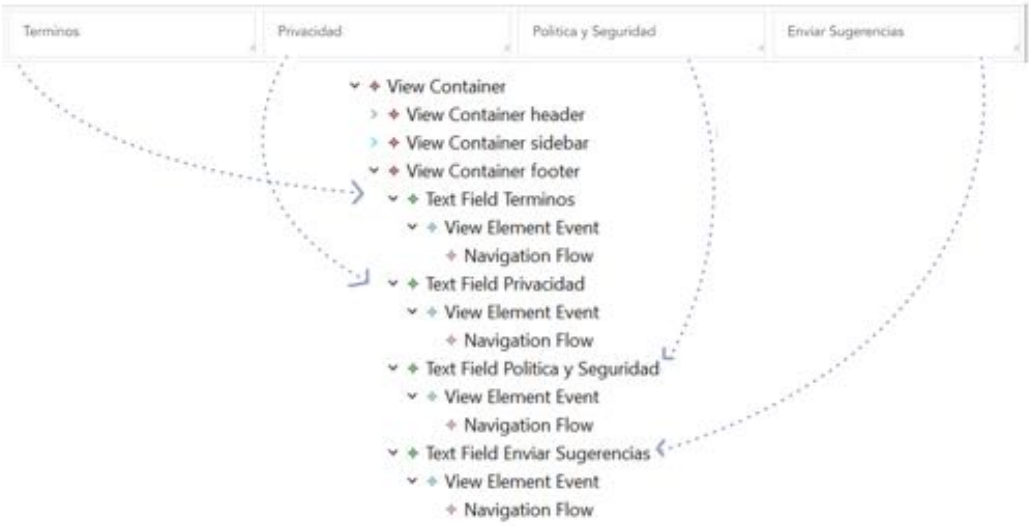


Figura 6.6: Correspondencia Footer - IFML

Cada link que se encuentra en el footer se representa en IFML como un `TextField` el cual contiene un `Event` con un `NavigationFlow`, determinando el destino hacia el cual se redirecciona luego de clickeado el link.

Generación de código

La fase final del ciclo se da al momento de ejecutar la transformación, generando el código HTML que representa la página de "Reproducción". El código generado representa la mayoría de la interfaz, define su estructura e interacciones ya que IFML sólo toma en cuenta estos aspectos. Por esta razón, los elementos HTML que conforman la página no tendrán propiedades de estilos, tamaños y posiciones. Para tener un sitio totalmente funcional se deberían especificar estas propiedades, de forma tal de brindarle al usuario una interfaz cómoda y amigable. Por esta razón, se creó una hoja de estilos que especifica esos aspectos de los componentes, y así validar que la estructura del código HTML generado se corresponde con la esperada por el mockup.

De esta forma, se muestran en la Figura 6.7 las interfaces generadas, en primera instancia la interfaz sin estilos aplicados, para luego mostrar la inter-

faz que se obtiene luego de incluir la hoja de estilos que se implementó.

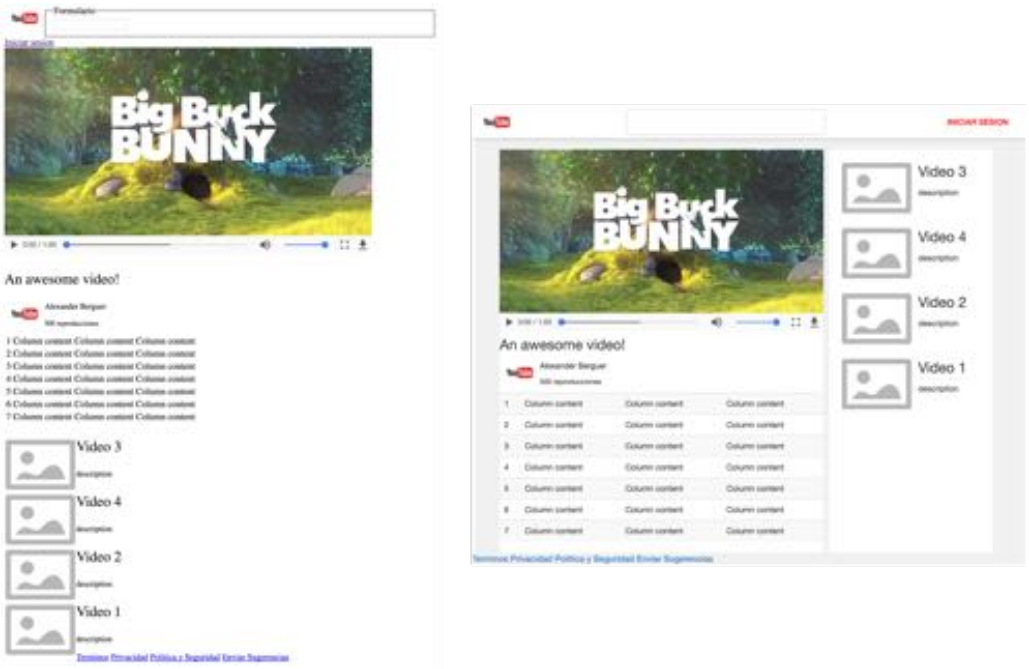


Figura 6.7: HTML generado de la página de reproducción sin estilos (izquierda) y con estilos (derecha)

6.2.2. Mi Canal

Siguiendo con el mockup de YouTube, se detalla la página "Mi Canal", ya que es bastante completa en contenido y utiliza varios componentes. En la figura Figura 6.8 se muestra una captura de la pantalla de esta sección en el sitio YouTube.

Correspondencia de elementos

Siguiendo la metodología de la parte anterior, en la figura Figura 6.9 se muestra el mockup generado que representa la página de "Mi Canal", y se lo analiza dividiéndolo en secciones con el fin de mostrar de forma más clara la correspondencia entre el mockup y el modelo IFML.

El header y el footer son similares a los descritos en la página de reproducción, con la excepción que en la página de "Mi Canal" hay un usuario logueado y en la otra no, lo que lleva a que en vez de tener el botón de acceder, se tiene la imagen de perfil del usuario y las notificaciones. El elemento "sidebar", es

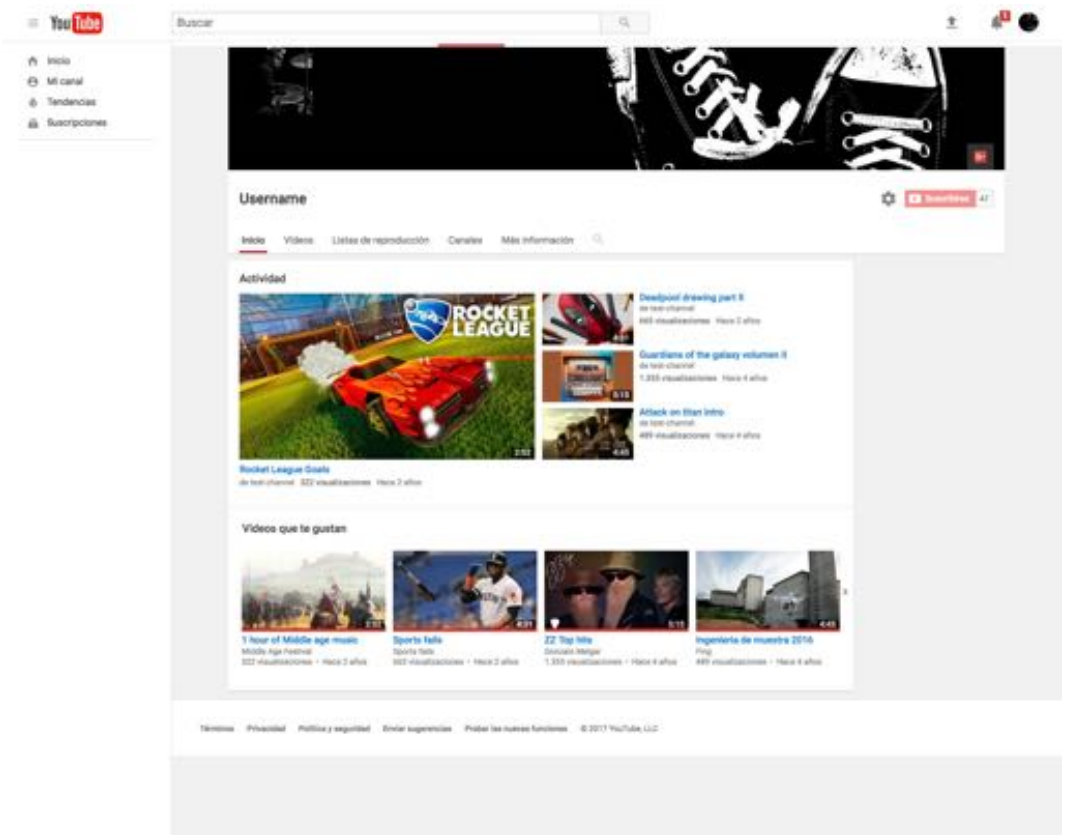


Figura 6.8: Página de 'Mi Canal'

la sección que se encuentra a la izquierda de la página y contiene links a otras páginas del sitio. La correspondencia de esta sección es análoga a la del footer, que se detalló en la parte anterior.

A la derecha del "sidebar", se encuentra el contenido de la sección, que está conformado por una imagen de portada, el nombre del usuario, un botón de suscripción y una sección de cinco pestañas, que comprenden en su interior varios componentes, como imágenes, textos, links, botones, etc. Por ejemplo en la pestaña "Inicio", se muestran los videos subidos por el usuario. La sección que contiene las cinco pestañas es la elegida para explicar en detalle. En la Figura 6.10 se muestran las correspondencias entre los elementos del mockup y el modelo IFML generado, utilizando la visualización de árbol provisto por el editor de modelos de eclipse.

Se puede ver que las pestañas están contenidas todas en un `ViewContainer` de nombre tabs con la propiedad de ser XOR, el cual tiene un `ViewContainer` por cada una de ellas, con su nombre correspondiente. se encuentra resaltada utilizando números rojos. Dentro de la tab `Inicio`, se puede ver que hay

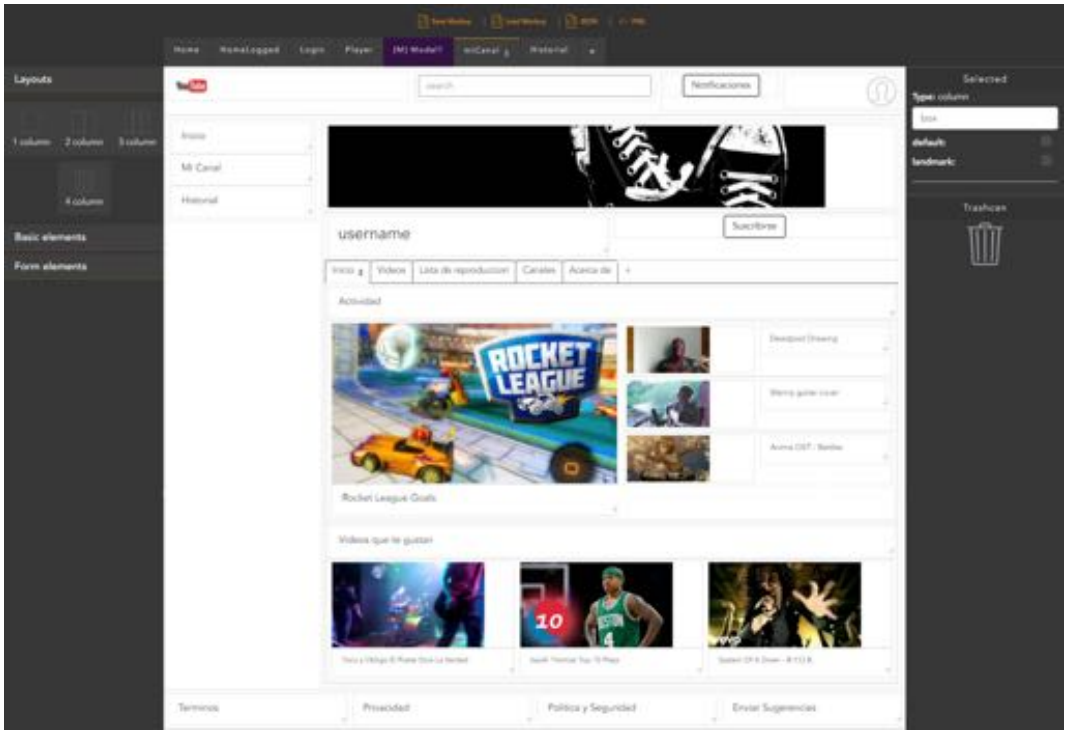


Figura 6.9: Mockup de la página de mi canal

Figura 6.10: Correspondencia sección tabs y modelo IFML

dos `ViewContainer`, donde cada uno contiene un `TextField` que determina el título de la sección y otro `ViewContainer` en el cual se encuentra el contenido, los cuales se encuentran encuadrados y serán explicados con más detalle a continuación, mediante la Figura 6.11.

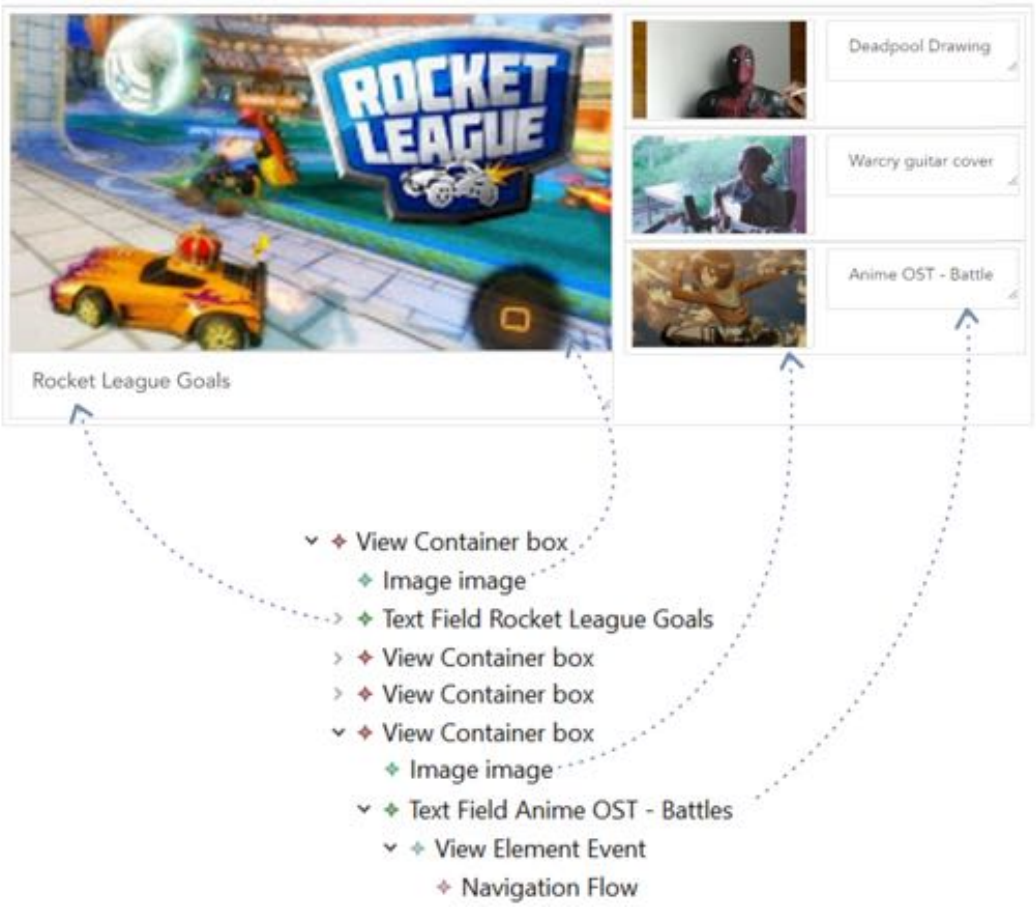


Figura 6.11: Correspondencia sección Actividad y modelo IFML

En la sección "Actividad", se muestran los últimos videos agregados por el usuario dueño del canal, por lo que en esta sección se ve que dentro del `ViewContainer` se encuentra una imagen de tipo `Image`, que muestra una captura del video y debajo de él se encuentra un `Text Field` con el nombre del video. Dicho `Text Field` a su vez es link, el cual está representado mediante un `ViewElementEvent` con un `NavigationFlow` cuyo destino es la pantalla de reproducción. Luego se encuentran a la derecha tres `ViewContainers` que tienen la misma estructura que el descrito anteriormente, al igual que los elementos descritos en la Figura 6.12, la cual muestra la comparación de la sección "Videos que te gustan".

Generación de código

Al igual que con el ejemplo de la página anterior, se muestra el código HTML generado para la pagina de "Mi Canal", en primera instancia la inter-

Figura 6.12: Correspondencia sección "Videos que te gustanz modelo IFML

faz sin estilos aplicados, para luego mostrar la interfaz que se obtiene luego de incluir la hoja de estilos que se implementó, donde se puede apreciar que la estructura final generada, se corresponde con la expresada mediante el mockup. Vale la pena destacar el uso del framework bootstrap, mencionado en el Capítulo 5, el cual luego de aplicado permite la representación de las pestañas de forma correcta.

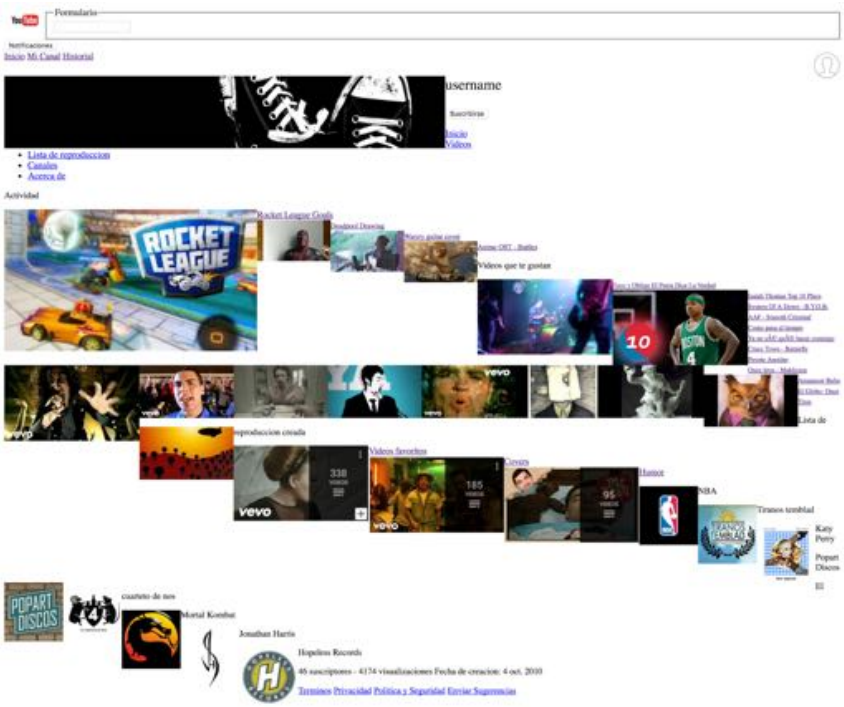


Figura 6.13: HTML generado de la página de "Mi canal"

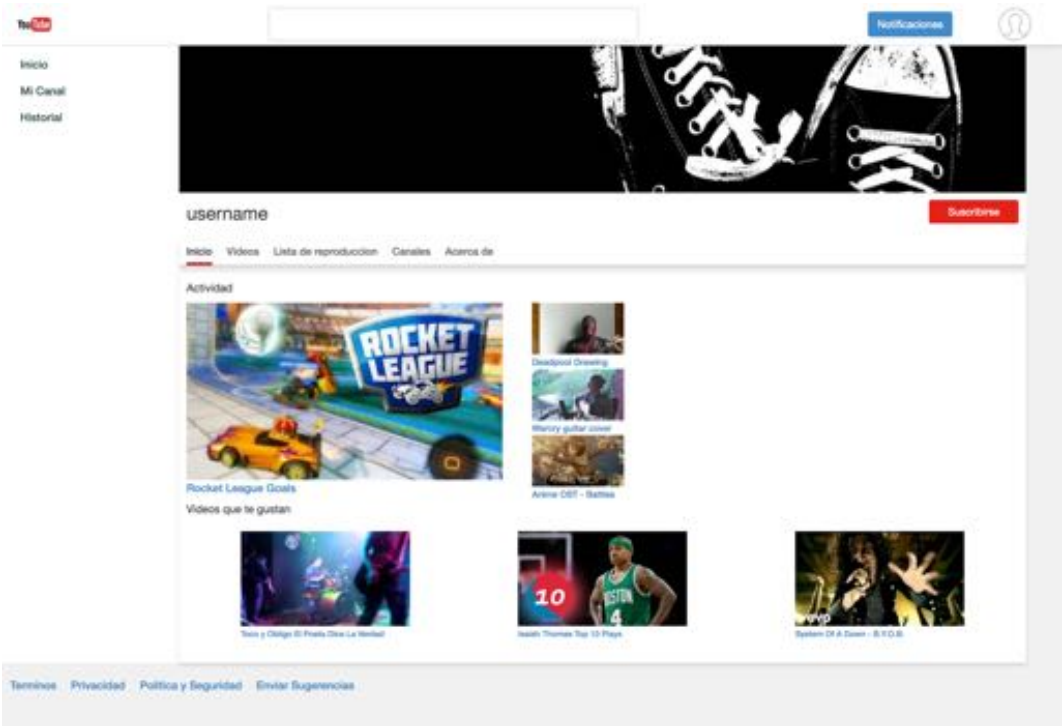


Figura 6.14: HTML generado de la página de "Mi canal" con estilos

Capítulo 7

Conclusiones y Trabajo Futuro

7.1. Conclusiones

Los mockups cada vez son más útiles en el desarrollo de software, ya que permiten validar el desarrollo con el cliente en etapas tempranas. Los desarrolladores de software utilizan cada vez más estos artefactos, pero luego en etapas de desarrollo se pierden o solo quedan para documentación. Por esta razón surge la necesidad de poder utilizar de alguna manera los mockups en la implementación del sistema en sí. En este proyecto se ha introducido un enfoque, basado en ingeniería dirigida por modelos para el reuso de mockups, utilizando los modelos IFML como capa de abstracción, para obtener el código ejecutable de las interfaces diseñadas.

La utilización de un lenguaje específico de dominio, como lo es IFML, permitió abstraer los conceptos, para luego realizar la transformación M2T fácilmente. Dada su naturaleza extensible, facilitó el hecho de definir algunos conceptos faltantes en el metamodelo, de manera de poder diseñar un mockup web más completo. De esta forma se evitan errores propios de la codificación, así como también permite que usuarios no técnicos participen en el proceso de desarrollo del sistema.

En resumen, se logró cumplir con los objetivos planteados al comienzo, comprender la realidad planteada, estudiar trabajos relacionados como lo fue el Mockup Driven Development, para luego definir un enfoque propio incluyendo los mejores aspectos de todos ellos, con el agregado de utilizar IFML que es considerado un estándar a la hora de representar las interfaces de usuario. Realizando un análisis pormenorizado, se presenta una lista con los resultados

alcanzados a lo largo del proyecto, los cuales se corresponden con los resultados esperados detallados en el Capítulo 1 respectivamente:

1. Se realizó un estudio del estado del arte de las herramientas generadoras de mockups, estudiando y documentando sus aspectos relevantes.
2. Se estudió el lenguaje IFML, detallando sus componentes y características principales, destacando los aspectos de estructura y navegación.
3. Se realizó un análisis de componentes, definiendo una relación entre los elementos más generales de la web y los elementos disponibles IFML. Como agregado, se realizó también un análisis de correspondencia entre IFML y HTML.
4. Se implementó un prototipo como prueba de concepto, desarrollando una herramienta capaz de generar y procesar un mockup, para luego terminar el flujo obteniendo el código ejecutable. Luego de esta etapa se presenta un caso de estudio que muestra sus funcionalidades y valida la idea.

7.2. Evaluación de la solución

A medida que el proyecto fue avanzando, se fue evaluando cada uno de los aspectos positivos y negativos de la solución. Entre los puntos más fuertesse destaca que el enfoque planteado es genérico e independiente de plataforma; si bien la prueba de concepto se realizó en un contexto web, podría ser utilizado para generar interfaces en cualquier tecnología, por ejemplo para dispositivos móviles, de escritorio, etc. Con respecto a aspectos de implementación, se hizo hincapié en desarrollar un sistema escalable, que se pueda extender y adaptar sin perder calidad de forma relativamente sencilla. Por ejemplo, agregar un nuevo componente, solamente implica agregar su representación en el frontend, agregar unas pocas líneas de código en el backend, y agregar un nuevo template en la transformación M2T. Otro aspecto a tener en cuenta en la implementación fue el desarrollo de una interfaz con la cual el usuario puede familiarizarse fácilmente, permitiéndole a un usuario inexperto utilizar la aplicación sin mayores inconvenientes.

Uno de los puntos débiles encontrados, es el hecho de tener que ejecutar la transformación implementada mediante el uso de acceleo de forma diferenciada. Se trabajó en integrarlo con las demás herramientas, pero luego de intentarlo y ver que no era posible de forma sencilla, se dejó de lado ya que no

afectaba el objetivo del proyecto.

Por otro lado la fuerte dependencia que se tiene con Eclipse y su framework EMF puede ser considerado una debilidad, ya que condiciona a que el backend tenga que utilizar Java como tecnología de desarrollo.

A su vez, el hecho de haber extendido IFML puede ser tomado como un punto débil en el caso de querer integrar los modelos generados con herramientas de terceros, las cuales se basen en el estándar de IFML. Los modelos no van a poder ser interpretados en su totalidad pudiendo perder información relevante.

7.3. Trabajo a futuro

En esta sección se presenta una serie de puntos que consideramos mejoras significativas del proyecto a futuro.

- La paleta de componentes del prototipo implementado contiene un conjunto básico de elementos, los cuales fueron necesarios para realizar la prueba de concepto. Por lo consideramos trabajo a futuro agregar nuevos elementos web a la paleta, lo que genera la necesidad de tener que agregar la correspondencia con elementos del modelo IFML y a su vez la correspondiente transformación a HTML.
- Agregar la posibilidad de que la herramienta provea un mecanismo para lograr la integración de datos, lo que implicaría la modificación de todos los componentes implementados. Por ejemplo en el editor de mockup se debe dar la posibilidad de crear conexiones a base de datos, o de importar archivos con datos a ser mostrados e intercambiados entre las distintas páginas. Con respecto al backend, se debe agregar el manejo de los elementos de `DataBindings` del lenguaje IFML y en la transformación M2T, se deben implementar dichos cambios en alguna tecnología particular.
- El prototipo solamente permite realizar mockups de páginas web, por lo que se podría extender el comportamiento para poder realizar mockups para otras tecnologías, como por ejemplo aplicaciones móviles o de escritorio. Este cambio implica tener que realizar implementaciones en los tres componentes desarrollados.
- El manejo de estilos en la herramienta no fue tomado en cuenta, ya que

como fue comentado a lo largo del documento, IFML no brinda soporte de estilos. Sería interesante en un futuro agregarlo, ya sea extendiendo IFML para que los soporte o por ejemplo que los estilos definidos en el mockup sean información de entrada de la transformación en paralelo con el IFML generado.

Referencias bibliográficas

[1] Douglas C. Schmidt. Guest editor's introduction: Model-driven engineering. *IEEE Computer*, 39(2):25–31, 2006.

[2] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012.

[3] Nishant Sinha and Rezwana Karim. Compiling mockups to flexible uis. In *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18-26, 2013*, pages 312–322, 2013.

[4] Axure. http://www.axure.com. Accedido: Marzo 2016.

[5] Balsamiq mockups. http://balsamiq.com. Accedido: Marzo 2016.

[6] Pencil project. http://pencil.evolus.vn. Accedido: Marzo 2016.

[7] Marco Brambilla and Piero Fraternali. *Interaction Flow Modeling Language*. Model-Driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kaufmann, 2015.

[8] Interaction flow modeling language (ifml) adopted as a standard by omg. https://marco-brambilla.com/2013/03/23/interaction-flow-modeling-language-ifml-adopted-as-a-standard-by-omg/. Accedido: Marzo 2016.

[9] Object management group. http://www.omg.org/. Accedido: Marzo 2016.

[10] Javier Troya Castilla Francisco Durán Muñoz and Antonio Vallecillo Moreno. Desarrollo de software dirigido por modelos. https://eva.fing.edu.uy/

pluginfile.php/123820/mod_resource/content/1/MDSDVallecillo.pdf. Accedido: Marzo 2016.

[11] Omg, model driven architecture – a technical perspective, 2001. http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01.

[12] Omg. meta object facility (mof) 2.0 query/view/transformation. final adopted specification version 1.1, object management group, 2009. http://www.omg.org/spec/QVT/. Accedido: Marzo 2016.

[13] Omg's metaobject facility. http://www.omg.org/mof/. Accedido: Marzo 2016.

[14] Acceleo - implementation of the object management group (omg) mof model to text language (mtl). https://www.eclipse.org/acceleo/. Accedido: Marzo 2016.

[15] Html - standard markup language used to create web pages. https://www.w3.org/html/. Accedido: Marzo 2016.

[16] Omg's unified modeling language. http://www.omg.org/UML/. Accedido: Marzo 2016.

[17] Xml metadata interchange (xmi). http://www.omg.org/spec/XMI/. Accedido: Marzo 2016.

[18] Extensible markup language (xml). https://www.w3.org/XML/. Accedido: Marzo 2016.

[19] José Matías Rivero, Gustavo Rossi, Julián Grigera, Juan Burella, Esteban Robles Luna, and Silvia E. Gordillo. From mockups to user interface models: An extensible model driven approach. In Florian Daniel and Federico Michele Facca, editors, *Current Trends in Web Engineering - 10th International Conference on Web Engineering, ICWE 2010 Workshops, Vienna, Austria, July 2010, Revised Selected Papers*, volume 6385 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2010.

[20] Indigo studio. http://www.infragistics.com/products/indigo-studio. Accedido: Marzo 2016.

[21] Ninja mock. https://ninjamock.com/. Accedido: Marzo 2016.

[22] Json - javascript object notation. http://www.json.org/. Accedido: Marzo 2016.

[23] Http - hypertext transfer protocol. https://www.w3.org/Protocols/. Accedido: Marzo 2016.

[24] Lenguaje javascript. https://developer.mozilla.org/es/docs/Web/JavaScript. Accedido: Agosto 2016.

[25] Single page applications. http://singlepageappbook.com/goal.html. Accedido: Agosto 2016.

[26] Java. https://www.java.com/es/about/whatis_java.jsp. Accedido: Julio 2016.

[27] Eclipse ide. https://www.eclipse.org/. Accedido: Marzo 2016.

[28] Metamodelos ifml. http://www.ifml.org/ifml-materials/. Accedido: Junio 2016.

[29] Youtube. https://www.youtube.com. Accedido: Junio 2016.

ANEXOS

Anexo 1

Modelos IFML

1.1. Modelo IFML

El primer componente que se debe mencionar es el modelo de IFML por sí mismo, el cual establece la estructura general del lenguaje y todos sus elementos. A grandes rasgos el metamodelo de IFML se muestra en la Figura 1.1.

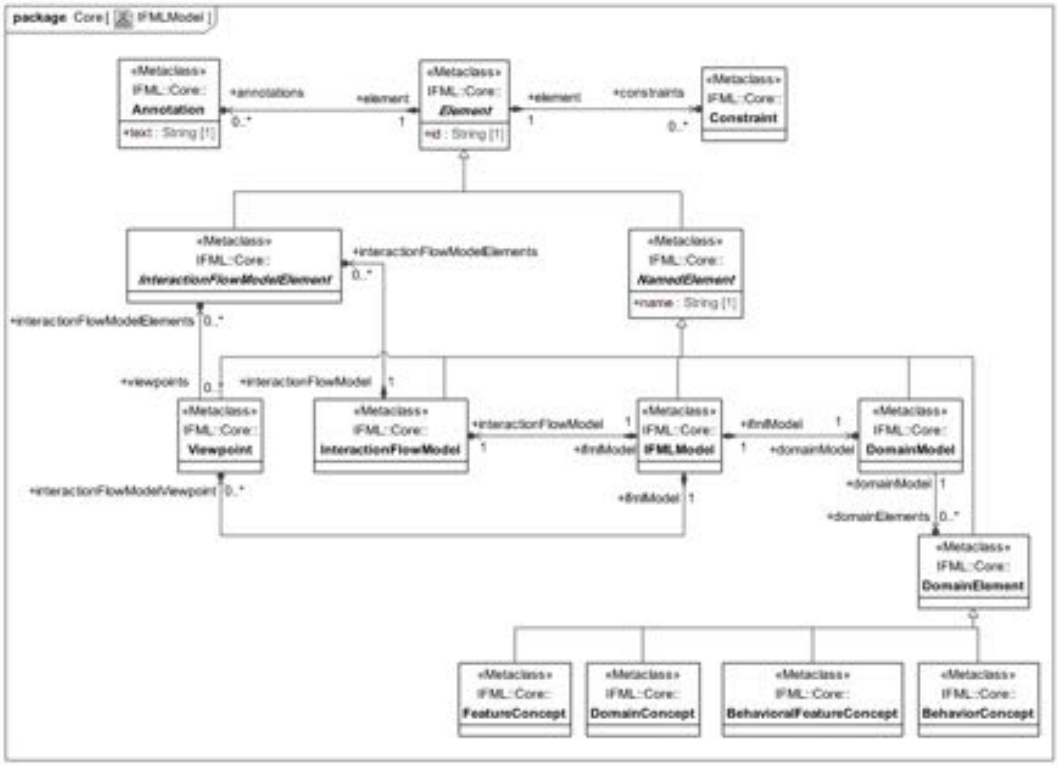


Figura 1.1: Metamodelo IFML

Dicho metamodelo es el contenedor de más alto nivel para el res-

to de los elementos previamente mencionados. Contiene principalmente un `InteractionFlowModel`, un `DomainModel` y opcionalmente podría contener `ViewPoints`. Como se puede apreciar en el metamodelo, un `ViewPoint` presenta un aspecto específico del sistema, mientras que un `InteractionFlowModel` es la vista que obtiene el usuario de toda la aplicación y todos sus componentes. El `ViewPoint` tiene una referencia a un conjunto de `InteractionFlowModelElements`, los cuales definen una parte funcional completa del sistema. Un `InteractionFlowModelElement` es una clase abstracta que es la generalización de cada elemento de un `InteractionFlowModel`. Por otra parte un `DomainModel` representa la descripción del contenido y el comportamiento dentro del `InteractionFlowModel`. El mismo `DomainModel` está compuesto de `DomainElements`, los cuales pueden ser conceptos, propiedades, comportamientos y métodos (en el diagrama se encuentran representados como `DomainConcept`, `FeatureConcept`, `BehaviorConcept` y `BehavioralFeatureConcept` respectivamente). Otro punto a destacar es que para cualquier elemento existente, se pueden definir `Annotations` o Restricciones.

1.2. InteractionFlowModel

Un `InteractionFlowModel` contiene todos los elementos relacionados a la vista que tiene el usuario de la aplicación, la cual es representada por un conjunto de `InteractionFlowModelElements`. El modelo de dicho el elemento se representa la figura 1.2.

Un `InteractionFlowModelElement` tiene 7 subtipos y son los encargados de las interacciones:

- InteractionFlowElement
- InteractionFlow
- ParameterBindingGroup
- ParameterBinding
- Parameter
- Expression
- Module

Un `InteractionFlow` es una directa entre dos `InteractionFlowElements` y pueden implicar la navegación a lo largo de la interfaz de usuario o sólo una

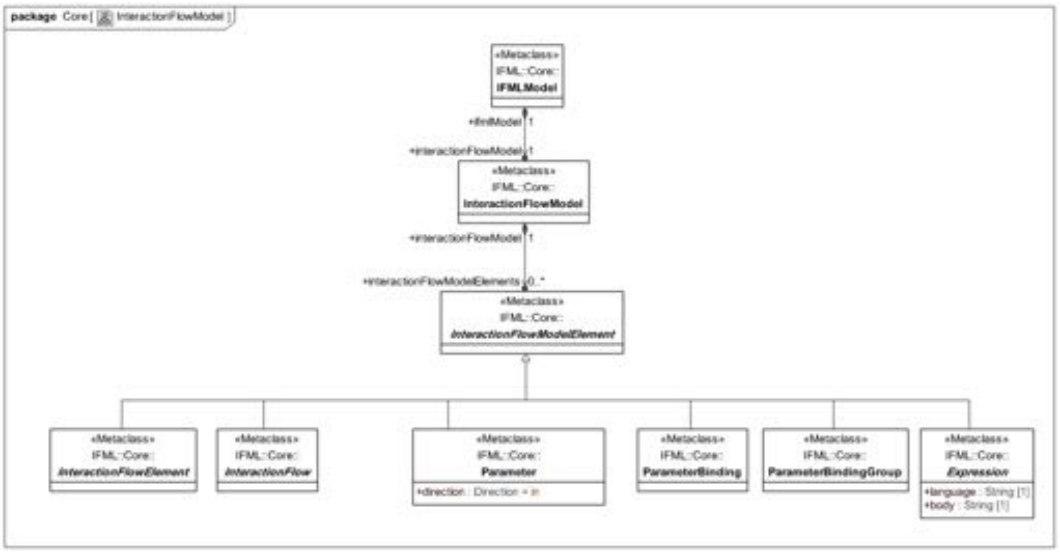


Figura 1.2: Metamodelo InteractionFlowModel

transferencia de información mediante la realización de valores de los parámetros de un `InteractionFlowElement` a otro.

Un `Parameter` es un nombre escrito que contiene valores. Los `Parameters` se llevan a cabo por `InteractionFlowElements` es decir, se llevan a cabo por `ViewElements`, `ViewComponentParts`, `Ports` y `Actions`.

Los `Parameters` corren entre `InteractionFlowElements` cuando los `Events` ocurren. Por ejemplo teniendo en cuenta el flujo de un `Parameter` P a partir de un `InteractionFlowElement` A a una `InteractionFlowElement` B, el parámetro P se considera como un parámetro de salida de `InteractionFlowElement` A y como un `Parameter` de entrada de `InteractionFlowElement` B.

Por otro lado los `ParameterBindings` determinan un `Parameter` de entrada de un `InteractionFlowElement` específico, en base a un `Parameter` de salida de un `InteractionFlowElement` origen.

Un módulo es un conjunto totalmente funcional de `InteractionFlowModelElements`, que puede ser reutilizado para mejorar el mantenimiento de IFML. Los módulos pueden ser sustituidos por otros módulos o `InteractionFlowElements` con la misma entrada y salida parámetros.

Una `Expression` define una declaración que evaluará en un contexto determinado a una sola instancia, un conjunto de instancias, o un resultado vacío. Tipos específicos de expresión, como expresiones booleanas, etc., se representan

como especializaciones de `Expression`.

1.3. InteractionFlowElements

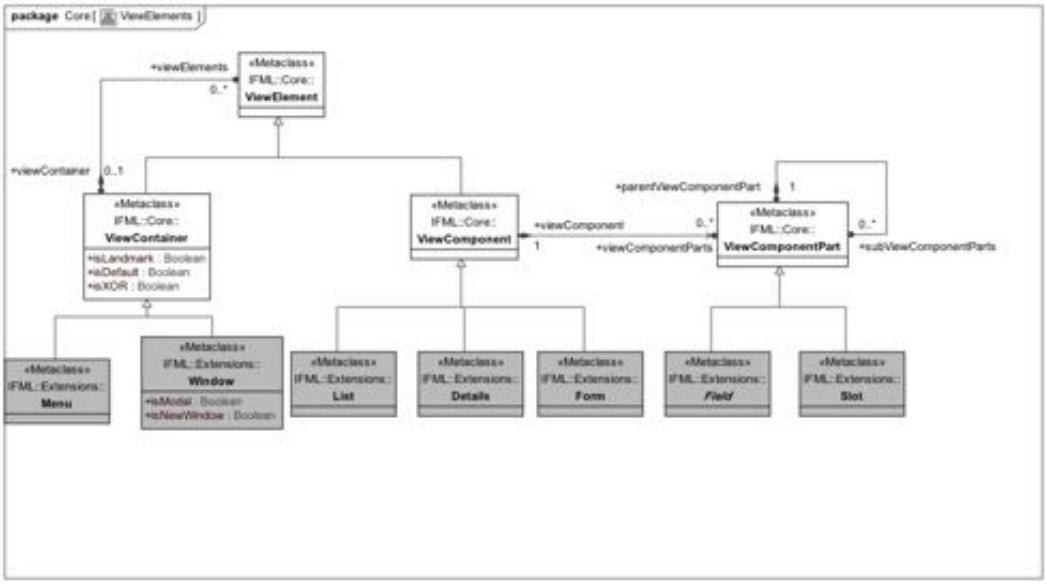


Figura 1.3: Metamodelo InteractionFlowModelElements

Es uno de los conceptos clave de IFML, el cual representa las partes del sistema como son los `ViewElements`, `ViewComponentsParts`, `Action`, `Events`, etc. que participan de las relaciones definidas por los `InteractionFlows`. Los `IneractionFlowElements` también contienen Parameters, que usualmente son transmitidos entre los `InteractionFlowElements` como consecuencia de un evento. Existen dos tipos de `InteractionFlows`, estos son los `NavigationFlows` y los `DataFlows`.

1.4. ViewElements

Son los elementos del metamodelo IFML que son visibles en la interfaz de usuario, estos pueden ser `ViewContainers` o `ViewComponents`. Los `ViewContainers`, como por ejemplo ventanas HTML o páginas, son contenedores de otros `ViewContainers` o `ViewComponents`, mientras que los `ViewComponents` son elementos de la interfaz que despliegan contenido o que permiten que el usuario interactúe con la aplicación.

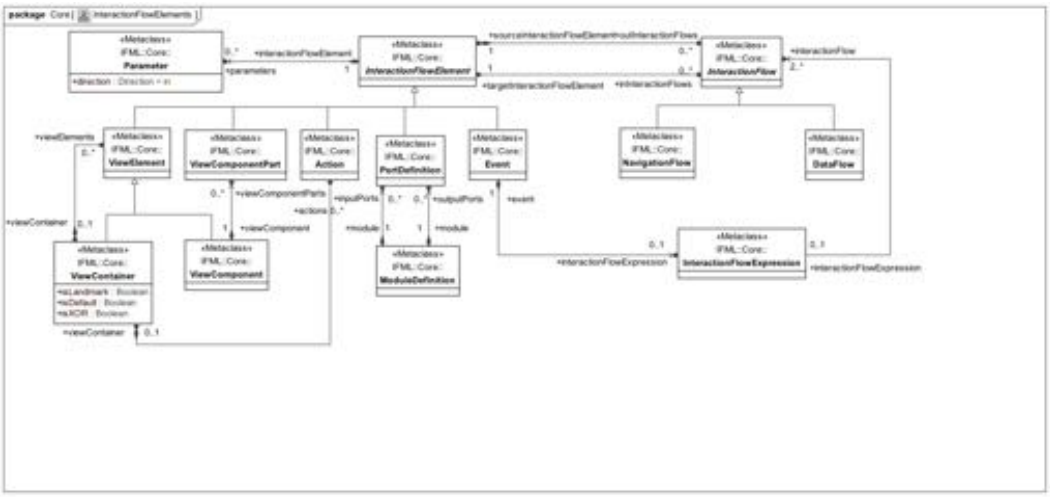


Figura 1.4: Metamodelo ViewElement

Un `ViewContainer` puede ser marcado como landmark, XOR y default. Si es landmark, significa que puede ser alcanzado desde cualquier otro `ViewElement` sin necesidad de definir un `InteractionFlow`, lo que hace que el modelo esté menos sobrecargado y sea más entendible para el usuario. Si un `ViewContainer` está marcado como default, significa que cuando se accede al `ViewContainer` padre, lo que se le presentará al usuario será el `ViewContainer` hijo marcado con default. Si tengo varios `ViewContainer` hijos, y estos están marcados como XOR, los `ViewContainer` se podrán mostrar al usuario de a uno a la vez.

Los `ViewComponents` existen solamente dentro de `ViewContainers`. Estos son elementos de la interfaz que pueden tener comportamiento dinámico, desplegar contenido, aceptar entradas de datos, etc. Un `ViewComponent` puede estar creado a partir de `ViewComponentParts`, por lo que una `ViewComponentPart` no puede existir fuera de un `ViewComponent`, pero sí puede tener Eventos o `InteractionFlows` asociados. En el metamodelo de la figura se puede ver que en gris hay elementos más concretos, estos son parte de un paquete de extensión de IFML.

1.5. Parameters

Un `Parameter` es un elemento tipado y con multiplicidad, cuyas instancias contienen valores. Los `Parameters` están contenidos en los `InteractionFlowElements` y son transportados por ellos cuando un `Event`

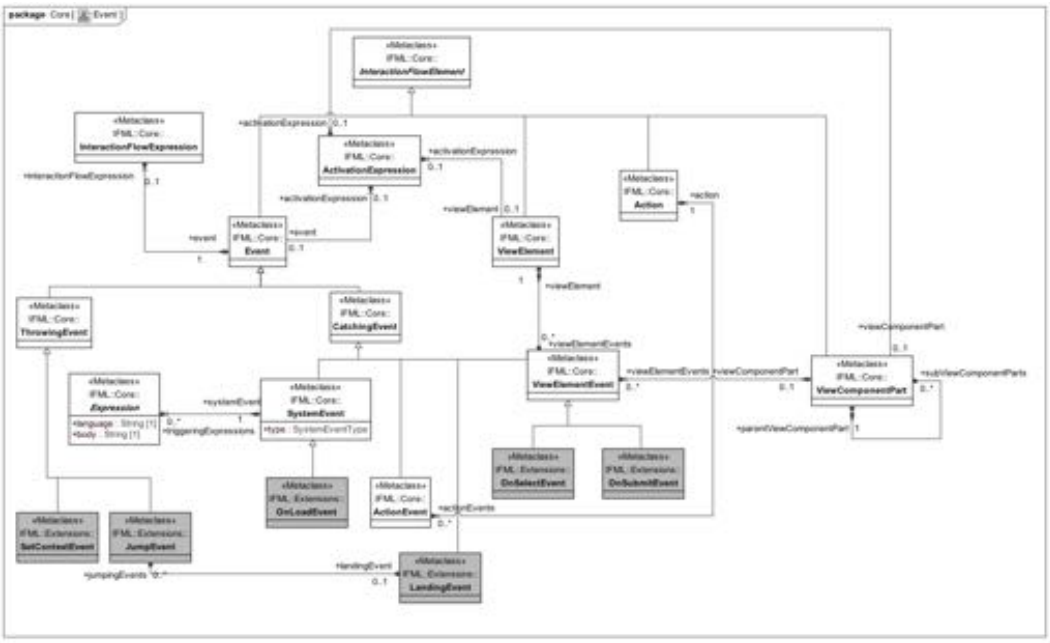


Figura 1.5: Metamodelo Parameter

es disparado. Los `Parameters` pueden ser de entrada, salida o ambas, por defecto estos son de entrada. Un `ParameterBinding` determina a cual `Parameter` de entrada de un `InteractionFlowElement` objetivo está conectado un `Parameter` de salida de un `InteractionFlowElement` de origen. Y además, determina como el valor del parámetro va a ser transferido cuando un `Event` sea disparado o cuando se siga un `InteractionFlow`.

1.6. Events

Los `Events` son ocurrencias que pueden afectar el estado de una aplicación, son un subtipo de los `InteractionFlowElement`. Estos `Events` se pueden clasificar en dos categorías principales:

- **CatchingEvents**: eventos que son capturados en la interfaz y que dispara un cambio en ella. Estos pueden ser de tres tipos.
    - **CatchingEvents**: eventos que son capturados en la interfaz y que dispara un cambio en ella. Estos pueden ser de tres tipos.
    - **ViewElementEvents**: Pertenecen al `ViewElement` con el que están relacionados, esto significa que los `ViewElements` contienen

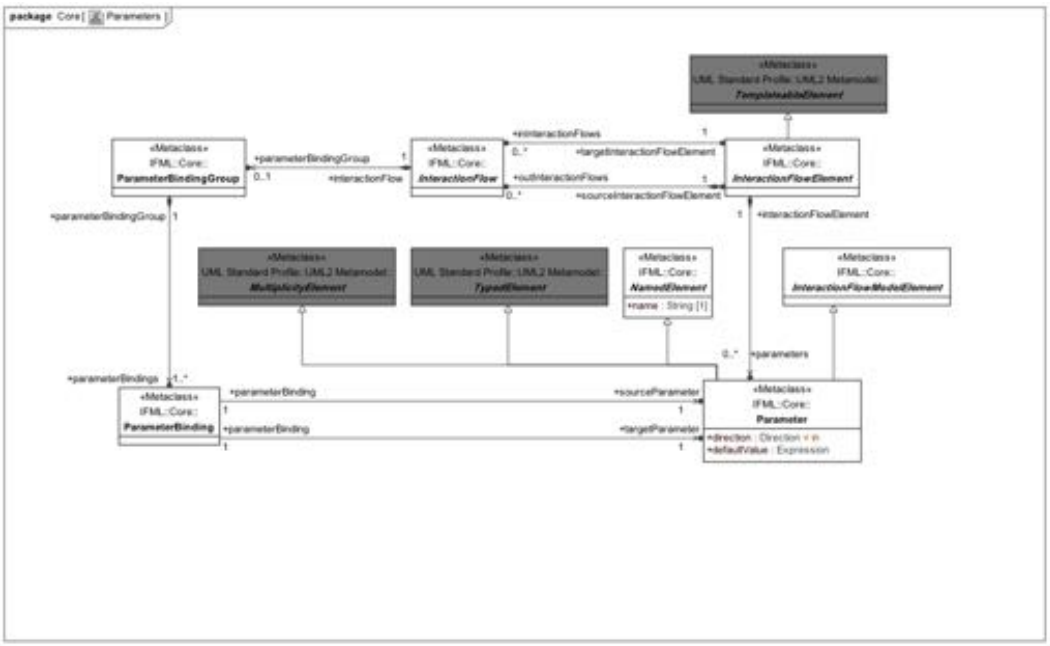


Figura 1.6: Metamodelo Event

`Events` que permiten al usuario activar interacciones en la aplicación.

- **ActionEvents**: Pertenecen al `Action` con el que están relacionados, una `Action` puede disparar un `ActionEvent` durante su ejecución o cuando termina, normalmente o con una excepción.
- **SystemEvents**: Estos tipos de `Events` son independientes, están al nivel del `InteractionFlowModel`. Estos eventos son por ejemplo, `OnLoadEvent`, se disparan cuando se cumple alguna condición especial, un problema en la red, etc.

- **ThrowingEvents**: eventos que generados por la interfaz.

Anexo 2

Estudio de Herramientas

La idea general de este anexo, es dar un vistazo general a cada una de las características de interés a evaluar en cada herramienta de desarrollo de mockups. Para cada una de las características se utiliza la metáfora del semáforo, indicando para cada caso, el criterio necesario para que represente un estado particular (verde, amarillo o rojo), siempre y cuando sea posible determinarlo.

En cada una de las herramientas estudiadas, se realizará una breve descripción de la misma, donde debajo del título de cada herramienta se presenta el modelo de semáforo para cada característica establecida.

Se estudiaron un total de sesenta herramientas, con el fin de reducir la lista de posibles candidatos y así poder encontrar cantidad más acotada a la hora de evaluar su utilización en el proyecto.

2.1. Características a evaluar

Se presenta un listado con cada una de las características de interés a evaluar en cada herramienta de desarrollo de mockups, con una breve descripción de la misma.

Plataforma
Indica el tipo de plataforma sobre la cual la herramienta trabaja, por ejemplo: Plataforma web, Windows, Linux, Mac OS, etc.

Tipo de aplicación

Indica el tipo de aplicación para el cual se pueden generar diagramas. Por ejemplo: Aplicación web, de escritorio, móvil, etc.

Código abierto
Indica si la herramienta es de código abierto o no. En caso de serlo se le asignará el estado verde, si tiene alguna porción del código como abierto y otra no sería amarillo y rojo sería en caso de no serlo.

Eventos
Indica si la herramienta soporta eventos de interacción. Dependiendo de la cantidad de eventos soportados (click, hover, drag, etc) será la clasificación resultante.

Licenciamiento
Indica el tipo de licencia de la herramienta, los valores posibles son: verde si es gratuito, amarillo si tiene diferentes niveles de licenciamiento y rojo si es una herramienta paga exclusivamente.

Documentación
Indica el nivel de la documentación provista por la herramienta para el entendimiento del funcionamiento de la misma. Se le asignará verde en caso de que la documentación sea completa, actualizada y de fácil entendimiento. Si no posee alguna de las características antes nombrada se le asignará el color amarillo y en caso de que la documentación sea nula se le asignará el color rojo.

Formatos
Indica el formato en el que se pueden exportar los diagramas diseñados en la herramienta. Dado que se necesita procesar dicho diagrama, es necesario obtener información interpretable al momento de exportar el mockup, siendo entonces verde si el formato es legible e interpretable luego de la exportación, amarillo si el formato por más que sea legible, no es muy común y es difícil de interpretar (puede ser un formato propietario para el cual no existen especificaciones), y rojo si no se puede exportar el mockup en un formato manipulable.

Navegación

Indica el nivel de navegación entre elementos de los mockups desarrollados por la herramienta. En caso de que soporte navegación clara entre cualquier tipo de elemento es verde, si no permite la navegación desde cualquier elemento del diagrama pero sí de algunos, entonces es amarillo, y en caso de no proveer manejo de navegación es rojo.

Componentes
Indica la variedad de componentes existentes dentro de la herramienta, como lo pueden ser, dropdowns, sliders, carouseles, canvas, mapas etc. Dependiendo de la variedad mencionada se clasificará usando el semáforo.

Usabilidad
Indica qué tan amigable es la herramienta con el usuario. Si la herramienta es muy clara y de fácil manejo es categorizada como verde. Si no es del todo intuitiva es amarilla, de lo contrario se categoriza como roja.

Asociación con datos
Indica si soporta asociación con fuentes de datos. En caso de poder asociar el diagrama con múltiples fuentes de datos, se categoriza como verde, sí solo se puede realizar dicha asociación con algunas pocas se considera amarillo y si no es posible se considera rojo.

Genera código
Indica si se puede generar código a partir del diagrama diseñado y en qué tecnología se genera dicho código. En caso de poder generar código en varias tecnologías, se listan dichas tecnologías y se categoriza como verde, sí solo se puede generar código para una tecnología se considera amarillo y si no es posible generar código se considera rojo.

Estilos personalizables
Indica si la herramienta permite personalizar estilos, ya sea colores de los elementos, fuentes de letras, tamaños de los componentes etc. Los valores posibles en esta clasificación serían: verde si permite una personalización total, amarillo si permite personalizar sólo algunos elementos y rojo si no permite ninguna personalización.

2.2. Estudio de herramientas

Se utilizó una notación específica para dicho análisis como se indica a continuación:

Tipos de diagrama soportados	Metáfora del Semáforo
Soporta diagramas de páginas web y aplicaciones de escritorio. Soporta diagramas para dispositivos móviles.	Rojo: indicando que cumple un criterio insatisfactorio. Amarillo: indicando que cumple un criterio regular. Verde: indicando que cumple un criterio satisfactorio. Gris: indicando que no se puede determinar el criterio.

Tabla 2.2: Notación específica del estudio

El listado de características es el previamente mencionado:

Tipo de aplicación que genera, Código abierto, Eventos, Licenciamiento, Documentación, Formatos que soporta, Navegación, Componentes, Usabilidad, Asociación con datos, Genera código, Estilos personalizables

En cada herramienta se detalla un resumen de todas las características estudiadas siguiendo dicho la notación indicada, donde en primer lugar se muestra los tipos de diagramas que soporta la herramienta, seguido del color de semáforo correspondiente a cada característica en su respectivo orden.

2.3. Evaluación de herramientas

Antetype http://antetype.com/

Antetype es una herramienta de escritorio disponible para equipos macintosh. Es de código cerrado y de pago, teniendo una versión trial de 30 días. La funcionalidad de la herramienta es la de hacer mockups para interfaces web y mobile. Tiene gran cantidad de componentes a utilizar, se pueden utilizar gestos y eventos asociados a dichos componentes y permite navegación entre vistas o pantallas de acuerdo a los triggers de los eventos. Tiene buena usabilidad y personalización, además de tener buena documentación, tutoriales y foros. No permite exportar el mockup a un archivo con formato conocido, que sea interpretable. Investigando bastante en la herramienta pudimos exportar a un formato el cual inspeccionando los paquetes de dicha generación, encontramos un archivo json. Dicho json es muy dificil de leer e interpretar, usa UUID para identificar los elementos y no hay una clara relación entre ellos. Puede exportar a html pero no mantiene la navegación.

Atomic https://atomic.io/

Plataforma web que permite realizar mockups de tipo Web y Mobile, muy personalizable pero de código cerrado, que permite guardar bocetos en formato html, png. Dentro de los aspectos positivos se destaca la navegación, los componentes, la usabilidad y los estilos personalizables. Sin embargo la generación de código en HTML resulta muy poco intuitiva, por lo que no resultará útil para el proyecto a realizar.

Axure http://axure.com/

Axure es una herramienta para hacer wireframes y mockups, disponible para PC y MAC. Permite diseñar interfaces de usuario tanto para web, mobile o aplicaciones de escritorio. La herramienta no es gratuita, posee una licencia gratis por 30 días que, luego de terminada hay que abonar la suma de 495 dólares anuales si se desea continuar su uso. La herramienta es muy completa, ofreciendo la manipulación de muchos eventos, una navegación clara, asociados a los eventos generados por el usuario y una buena usabilidad. No tiene una amplia variedad de componentes para utilizar, pero se pueden agregar librerías

desarrolladas por la comunidad, lo cual amplía considerablemente el espectro de componentes, además de ser completamente personalizables. Tiene una extensa Documentación, apoyada en la comunidad y foros en los cuales se pueden consultar. Además de esto, los desarrolladores brindan una API, que permite manipular los archivos generados por la herramienta, con lo cual da bastante libertad para trabajar. Por ejemplo con esta API se puede convertir los archivos en formato XML. Por último, la herramienta no permite asociación con tipos de datos y tiene la capacidad de exportar los mockups a HTML, y el árbol del sitio en formato word o csv.

Balsamiq http://Balsamiq.com

Balsamiq es una aplicación de pago disponible para PC y MAC. Permite hacer wireframes para interfaces web, mobile o para aplicaciones de escritorio de manera rápida y amigable al usuario. La herramienta no es open source, tiene 30 días de uso gratuito y tiene buena documentación. Tiene una amplia variedad de componentes disponibles para usar, pero son limitados en cuanto a su personalización y no brindan la posibilidad de asociarlos eventos de interacción a no ser el de çlick/tap". La navegación es solo por links entre páginas. Por último la herramienta permite exportar el mockup a formato JSON, no genera código ejecutable y tampoco tiene asociación con datos.

Easel https://www.easel.ly/

No es una herramienta de mockups, es para hacer infografías.

easyMocks http://easymock.org/

No es una herramienta de mockups, es para hacer mocks para test unitarios.

filesquare http://filesq.com/

Es una herramienta web para realizar mockups, pero la forma en la cual se realizan no es de nuestra utilidad. Ya que lo que te permite hacer es subir imágenes diseñadas en alguna otra herramienta y agregar links entre partes de ellas. Seleccionando áreas cuadrangulares en las cuales tendrá efecto dicho link. Los mockups realizados se pueden compartir y que distintos colaboradores agreguen comentarios, pero no se pueden exportar, por lo que quedó descartada.

Fireworks CS6 http://www.adobe.com/es/products/fireworks.html

La función principal de esta herramienta es diseñar las páginas con imágenes y gráficos finales más que para hacer mockups.

Flair Builder http://flairbuilder.com/

Flair Builder es una herramienta de código cerrado que permite hacer mockups para todas las plataformas. Como positivo, cuenta con una amplia variedad de componentes para utilizar, a los cuales se los puede personalizar completamente y asociar eventos predefinidos o generados por el usuario. Permite la navegación entre páginas de un mockup asociada a los triggers de los eventos y con animaciones de transiciones entre las pantallas. Otro aspecto positivo es que el archivo generado por el mockup está en formato JSON por lo cual es fácilmente manipulable. Lo Negativo de la herramienta es su limitada licencia gratuita, sólo brinda 15 días gratis. Luego hay que abonar 100 dólares para seguir utilizando. El otro aspecto negativo es relativo a la documentación y usabilidad de la herramienta. No brinda documentación, sólo brinda algunos pocos mockups de ejemplo. Además hay algunos elementos de la interfaz de usuario que no están claros y no se sabe cómo utilizarlos, que como consecuencia de no tener documentación se hace difícil deducir sus funcionalidades. Por último, la herramienta permite generar código HTML, pero éste es ilegible y muy difícil de utilizar para desarrollo.

Flinto https://www.flinto.com/

Flinto es una herramienta que tiene versión web y de escritorio para MAC. Permite subir imágenes de pantallas, hechas por ejemplo en photoshop o sketch y definir áreas en donde capturar eventos de usuario. Por ejemplo, se puede subir una imagen de un login de aplicación mobile y definir la zona en donde aparece el botón de "Log inçomo clickeable y realizar alguna acción como puede ser navegar a otra página. La herramienta no posee componentes propios y es imposible obtener una versión manipulable del mockup, por lo tanto no nos es de utilidad.

Fluid UI https://www.fluidui.com/

Es una herramienta web, la cual permite realizar mockups para Web, Móvil, Desktop así como para Wearables. Estos se crean realizando drag and drop desde una paleta de componentes ya diseñados, a los cuales se le pueden personalizar los estilos. El único evento que soporta es el click, sobre casi todos sus componentes, por lo tanto se puede realizar navegación entre páginas. Es de código cerrado y tiene licencia paga, pero ofrece un plan gratis el cual está muy limitado, tanto así que en dicho plan no se puede exportar a html. Por lo tanto esa característica no ha podido ser evaluada. Como no tiene un archivo que pueda llegar a ser interpretado, esta herramienta no resultará útil para el proyecto a realizar.

fluidIA http://www.fluidia.org/

El sitio web de dicha herramienta no se encuentra más en funcionamiento, se rastreó el repositorio de github que contiene el código pero sin documentación y sin siquiera ver algo gráfico como para poder probar algo dicha herramienta fue descartada.

ForeUI http://www.foreui.com/

Herramienta de mockups multiplataforma de escritorio personalizable de código cerrado, que permite guardar los diagramas en formato html, pdf, png. Dentro de los aspectos positivos se destaca la navegación, y los estilos personalizables. Si bien la generación de código HTML es relativamente parseable, no es posible obtener una versión gratuita con todas las prestaciones por más de 15 días.

Framebox http://framebox.org/

Framebox es una plataforma web que permite hacer mockups para aplicaciones web. No nos es de mucha utilidad ya que no es posible exportar ni guardar una versión del mockup. Tampoco tiene la posibilidad de definir múltiples pantallas, sino que es una herramienta para maquetar una parte o una página sola de un sitio, con lo cual no existe el concepto de Navegación. Tiene pocos componentes y muy básicos y no es posible asociar ningún tipo de eventos a ellos.

FrameLab25 http://frame.lab25.co.uk/

No es una herramienta de mockups, simplemente es un selector de Frames para una foto determinada, dejando al usuario poner una imagen como si estuviera en un dispositivo específico.

Gliffy https://www.gliffy.com/

Es una herramienta Web, que también se puede conseguir como plugin de Chrome o Attlasian. Permite realizar mockups para Web, Móvil y Desktop, pero estos son estáticos, no provee de eventos ni navegaciones. Por lo tanto lo que te ofrece es una página estática con los elementos que quieras agregarle, dichos elementos se agregan desde una paleta drag and drop. Es de código

cerrado y no es posible obtener una versión gratuita con todas las prestaciones por más de 15 días. La versión gratuita te permite generar 5 diagramas los cuales son públicos, y además no te permite exportar. Los formatos a los que permite exportar, son imágenes y un formato propio (Json) que te especifica los elementos y en qué ubicación se encuentran dentro de la página. Por lo que esta herramienta no resultará útil para el proyecto a realizar.

HotGloo http://www.hotgloo.com/

Es una herramienta web, la cual permite realizar mockups para Web, Móvil, Desktop así como para Wearables. Es de código cerrado y la versión gratuita de prueba que te brinda dura 15 días. Soporta varios eventos (click, init, over, out, change, focus in/out, keyboard enter, item click), los cuales te permiten definir navegaciones entre páginas. La desventaja radica en que solamente te permite exportar los diseños a imagen (PNG, PDF). Por lo que esta herramienta no resultará útil para el proyecto a realizar, ya que no tenemos cómo extraer lo diseñado.

InDesign CC http://www.adobe.com/products/indesign.html

Durante la exploración en busca de herramientas de maquetado nos topamos con InDesign, pero resultó no ser una herramienta de maquetado, sino que es una herramienta para diseñadores gráficos, similar a photoshop, cuya funcionalidad es crear folletos, documentos, diseñar sitios etc. Por esta razón la herramienta fue descartada.

Indigo studio http://www.infragistics.com/products/indigo-studio

Es una herramienta para realizar mockups que se puede utilizar tanto en sus versiones de escritorio para PC o para MAC. Se pueden realizar diseños para Web, Móvil, Desktop. Es de código cerrado y paga, te brinda un trial de 30 días. Se puede tramitar una licencia de estudiante solamente registrándose

con un mail .edu. Esta herramienta soporta una amplia variedad de eventos, tiene una documentación clara y concisa, y su usabilidad es muy buena. Por otro lado, permite exportar tanto a PDF, PNG, y a un formato propio el cual es un XML. En dichos archivos se pueden distinguir claramente los elementos diseñados y sus jerarquías, así como también las navegaciones entre páginas y elementos de ellas. Por lo que ésta herramienta es una de las posibles candidatas para la realización del proyecto.

inPresso Screens http://www.inpreso.com/

Esta aplicación no existe más.

InVision http://www.invisionapp.com/

Es una herramienta web, la cual permite realizar mockups para Web, Móvil, Desktop así como para Wearables. La forma en la cual se realizan no es de nuestra utilidad. Ya que lo que te permite hacer es subir imágenes diseñadas en alguna otra herramienta y agregar links entre partes de ellas. Seleccionando áreas cuadrangulares en las cuales tendrá efecto dicho link o hover. Los mockups realizados se pueden compartir y que distintos colaboradores agreguen comentarios y se pueden exportar a html, pero este es poco descriptivo, utiliza las imágenes y scripts propios para armar las navegaciones. Por lo tanto, es que descartamos esta herramienta.

iPlotz http://iplotz.com/

iPlotz es una herramienta de código cerrado que permite hacer maquetas tanto para web, como dispositivos móviles. Está disponible para todas las plataformas incluyendo web. Tiene una versión free con todas las funcionalidades pero limitada a un solo proyecto. La herramienta está bastante completa, Tiene múltiples componentes y templates predefinidos para utilizar, los cuales son totalmente personalizables y es posible manejar cualquier tipo de eventos en

esos componentes, además de eventos como .ºn page loadz gestos para dispositivos mobile. Permite una navegación asociada a los eventos de interacción, tiene buena usabilidad y es posible asociarlo con fuentes de datos a partir de archivos csv, los cuales quedan como templates para usos futuros. Lo negativo de la herramienta es que los archivos generados sin binarios y no hay posibilidad de manipularlos. Exporta el mockup sólo en HTML, pero usa sus propios js y no es muy parseable.

iRise http://www.irise.com/

Herramienta de plataforma web exclusiva, de código cerrado, que brinda soporte para la mayoría de los eventos del mouse (over, enter, click, double click, etc.). Dentro de los aspectos positivos se destaca la navegación, el manejo de eventos y la variedad de componentes disponibles. Como aspecto negativo tenemos que carece de documentación, no tiene una versión gratuita de más de 30 días y no exporta en un formato relativamente parseable para nuestras necesidades.

Jumpchart https://www.jumpchart.com/

Es una herramienta web, que te permite realizar mockups tanto para Web como para Desktop. La forma en que se realizan es escribiendo en texto plano, tiene una barra de herramientas que te permite agregar elementos, pero lo que se agrega es el texto que luego se traducirá a un elemento de html puro. No tiene estilos, solo te permite generar elementos de html con navegación entre páginas a través de links (hipervínculos). Es muy poco intuitiva, su usabilidad es mala. Tiene un a versión free la cual te permite tener un solo proyecto y luego hay versiones pagas. Se puede exportar a html, pero el costo de exportarlo es elevado, por lo que descartamos esta herramienta, además de que los botones, etc. no tienen eventos asociados como debería ser.

Just in mind Prototyper Pro http://www.justinmind.com/

Herramienta multiplataforma bastante completa de código cerrado. Dentro de sus características destacadas se encuentra el manejo de navegación entre componentes, la usabilidad, y la variedad de elementos existentes. Otra característica es que permite cargar determinados componentes agregando una estructura de datos denominada DataMaster, si bien esto es interesante ya que la destaca en comparación con el resto, no permite agregar datos desde una fuente externa. Tiene versión gratuita de 30 días de duración, dentro de la cual esta especificada la exportación en HTML, pero no tiene dicha opción disponible, por lo que quedaría descartada la utilización de dicha herramienta.

JustProto http://justproto.com/

Esta aplicación no existe más.

Kony http://www.kony.com/products/visualization/design

Esta aplicación es para desarrollar aplicaciones, no para realizar mockups.

Layout it! http://www.layoutit.com/

Layout it es una aplicación web gratuita que permite crear una estructura básica de un sitio a partir del framework css Bootstrap. Funciona con el método drag and drop de componentes bootstrap hacia la pantalla. No es de utilidad en relación a nuestro proyecto ya que no es una herramienta de mockups, sino para crear el layout inicial de un sitio y bajarse el HTML resultante.

Lumzy http://www.lumzy.com/

Es una herramienta web, que te permite realizar mockups tanto para Web como para Desktop. Es gratis, pero tiene un pop up que aparece cada cierto tiempo y si pagas te lo inhabilitan. Tanto la documentación como la variedad

de componentes y los eventos que soporta son muy pobres. Además, solamente exporta a PDF o PNG, por lo tanto esta herramienta no resultará útil para el proyecto a realizar.

MacAD and WinAD http://www.excelsoftware.com/

Esta aplicación no te permite descargarla ni probarla sin comprar la licencia, por lo tanto no nos fue posible evaluarla.

Maqetta http://maqetta.org/

Herramienta de Windows exclusiva la cual corre localmente un servidor web para poder hacer uso de la misma. Es gratis y de código abierto, lo que da lugar a inspección de código y modificación del mismo en caso de ser necesario. Se destaca en el manejo de navegación, usabilidad y variabilidad de componentes. La parte negativa es que el HTML generado por la herramienta no se corresponde mucho con la maqueta original, lo cual no brinda confianza en cuanto a la correspondencia de elementos.

Marvel https://marvelapp.com/

Herramienta web exclusiva de código cerrado que permite la realización de mockups muy básicos. Posee una cantidad limitada de componentes y en lo que refiere a eventos solo maneja la navegación de páginas. Básicamente permite armar un canvas libre de lo que resultaría siendo la aplicación a desarrollar, indicando la navegación entre elementos de la misma. Permite exportar en html y pdf entre otros, sin embargo el html generado no es de gran utilidad.

Mockabilly http://www.mockabilly.com/

Esta herramienta cesó su desarrollo y pronto se sacará del mercado, por lo que no estudiamos sus características.

Mockflow Wireframe Pro http://mockflow.com/apps/wireframepro/

Es una herramienta para realizar mockups que se puede utilizar tanto es su versión web, como en sus versiones de escritorio para PC o para MAC. La cual permite realizar mockups para Web, Móvil, Desktop. Es de código cerrado y es paga, pero tiene una versión free en la cual se puede tener un solo proyecto con un máximo de 4 páginas. Soporta solamente el evento del click, tiene una amplia variedad de componentes para elegir y una documentación un poco entreverada. La herramienta permite exportar a PDF, a imágenes y a HTML, pero el formato de dicho HTML no se pudo analizar porque para poder exportar en ese formato es necesario tener un a licencia.

MockingBird https://gomockingbird.com

MockingBird es una herramienta web, de pago y código cerrado. Es muy limitada en su versión free, permite tener solo 1 proyecto y luego de 6 días el mockups se vuelve read only. Tiene buena cantidad de componentes y buena usabilidad. Pero el mockup se guarda en la web y no es posible ningún tipo de exportación del mockup. Por lo que no es de utilidad para nuestro proyecto.

Mockplus http://www.mockplus.com/

Es una herramienta para realizar mockups que se puede utilizar tanto en sus versiones de escritorio para PC o para MAC o en versiones móviles para IOS y Android. La cual permite realizar mockups para Web, Móvil, Desktop. Es de código cerrado y paga, tiene una versión free en la cual no te permite exportar. Pero en el caso de poder exportar, los formatos en los que exporta no son de utilidad, ya que son imágenes o un ejecutable.

Mockup Builder http://mockupbuilder.com/

Herramienta multiplataforma de código cerrado, con una versión de prueba gratuita de 15 días. Se realizaron reiterados intentos para descargar esta aplicación pero la página de descarga tiene un error al redireccionar la página. Por lo visto posee una gran variedad de componentes y buen manejo de navegación, pero al no ser capaces de probar dicha herramienta siquiera queda descartada del conjunto.

Mockups Me http://mockups.me/

Herramienta multiplataforma que tiene versión para Windows, Mac, Linux, aplicación móvil y web. Es de código cerrado y maneja un escaso número de eventos (redirect to page, show popup, alerts, etc.). Posee una versión de prueba Web que no permite exportar los diagramas, de todas formas el formato de exportación más útil que posee son archivos de balsamiq, por lo que habría que realizar un doble mapeo si se opta por utilizar esta herramienta. Es decir realizar el proyecto utilizando la misma, exportarlo en el formato soportado por balsamiq y estudiar la forma en que mapea cada componente para poder determinar si es útil o no. Es por esto que nos parece excesivamente costoso el uso de esta herramienta, por lo que quedaría descartada para su posterior uso.

Moqups https://moqups.com/

Moqups es una herramienta web que en su versión web permite tener un proyecto activo. Al igual que la mayoría de las plataformas web los mockups son guardados en la web y no hay posibilidad de descargar una versión del mockup. La generación de código viene con la versión premium por lo que no pudimos probarla. La herramienta en sí es buena, tiene múltiples componentes y muy personalizables, pero no cumple con el perfil que estamos buscando para nuestro proyecto.

Napkee http://www.napkee.com/

Herramienta multiplataforma de código abierto y gratuita. Posee un manejo regular de eventos, así como los componentes básicos, usabilidad y navegación. Dicha herramienta permite exportar archivos con el fin de continuar su desarrollo utilizando balsamiq, por lo que se exporta un archivo y html. De todas formas dicha funcionalidad no parece funcionar correctamente ya que el balsamiq no fue capaz de reconocer muchos de los diagramas realizados, por lo que dicha herramienta no sería ideal para posterior uso.

NinjaMock https://ninjamock.com/

Herramienta web de código cerrado, con diversos niveles de licenciamiento que permite diseñar diagramas de aplicaciones web y móviles. Posee una versión gratuita que incluye todas las prestaciones de la herramienta pero con un límite máximo de tres proyectos lo cual no resulta una gran limitante. Dentro de los aspectos positivos se destaca la navegación, sus diversos componentes y la usabilidad. También se destaca por sobre todo su exportación en formato JSON la cual es muy legible e intuitiva a la hora de interpretar un diagrama realizado, por lo que ésta herramienta es una de las posibles candidatas para la realización del proyecto.

Omnigraffle https://www.omnigroup.com/omnigraffle/

No resultó ser una herramienta para construir mockups sino que es para crear cualquier tipo de gráficos, en los que están incluidos, por supuesto, las interfaces gráficas. Al no ser una herramienta específica para mockups, no tiene componentes, carece del concepto de navegación, eventos, etc y los productos que genera son imágenes.

Pencil Project http://pencil.evolus.vn/

Es una aplicación de escritorio para realizar mockups que se puede utilizar tanto en sus versiones para PC, MAC, Linux o como extensión de Firefox. Es de código abierto pero desde el 2013 que fue abandonado por los desarrolladores originales, un usuario de la herramienta creó un fork de github en el 2015 para continuar con el desarrollo, pero la última versión oficial del instalador es del 2013. La documentación es casi nula, aunque la herramienta es muy intuitiva para su uso. Soporta solamente el evento de click, lo que permite modelar la navegación entre distintas páginas. A la hora de exportar, se pueden generar imágenes en varios formatos, asi como tambien código HTML, pero en este código lo que se muestra son imágenes, y se especifican las coordenadas dentro de la pagina donde estan los link creados. Además los proyectos se guardan en un formato propio utilizando la sintaxis de XML, tiene una estructura que se puede llegar a entender pero no determina jerarquía entre los elementos.

Pidoco https://pidoco.com/

Pidoco es una herramienta de pago y código cerrado, que posee una versión trial por 31 días. Tiene una limitada cantidad de componentes, soporta eventos y la navegación se puede realizar a traves de éstos. Lamentablemente la herramienta no permite exportar de ninguna forma lo diseñado en un formato legible. Las opciones son imágenes o un código html muy dudoso.

Precursor https://precursorapp.com/

Herramienta web de código cerrado de diseño libre, por lo que un usuario experimentado puede diseñar lo que se proponga ya que es muy parecida a una herramienta de diseño gráfico que de mockups exclusivamente. Dentro de los aspectos destacados se encuentra la disponibilidad de una versión gratuita con todo lo necesario para el diseño de mockups. Su principal aspecto negativo es el hecho de que no exporta en ningún formato interpretable, solamente exporta los diseños en svg, png, pdf; por lo que quedaría descartada para el proyecto.

Proto.io https://proto.io/

Proto es una herramienta web para hacer prototipos de alta fidelidad. No es gratuita y el trial te permite trabajar por 15 días. Tiene buena cantidad de componentes, tanto web como mobile, eventos y gestos y navegación asociada a los eventos. Al igual que otras aplicaciones web, los diseños quedan en la nube, y solo es posible compartir mediante un share link o exportando el html.

Protoshare http://www.protoshare.com/

Protoshare permite hacer mockups para web y mobile. Es de código cerrado y es de pago, te permite trabajar gratuitamente durante 30 días. No tiene eventos ni navegación. Carece de documentación, la usabilidad no es buena, tiene componentes muy básicos y no exporta en ningún formato conocido. La exportación del HTML viene en la versión premium. No encontramos nada que pueda llegar a ser de utilidad en esta herramienta.

QuirkTools Wires http://quirktools.com/wires/

Herramienta web de código cerrado muy básica. Posee una versión gratuita y parece estar hecha utilizando bootstrap. El único tipo de eventos que posee es el de links de navegación. Dentro de las principales desventajas encontramos que no exporta diseños en otros formatos que no sean PDF, por lo que dicha herramienta quedaría descartada.

Savah http://www.savahapp.com/

Es una herramienta web para realizar mockups, pero la forma en la cual se realizan no es de nuestra utilidad. Ya que lo que te permite hacer es subir imágenes diseñadas en alguna otra herramienta y agregar links entre partes de ellas. Seleccionando áreas cuadrangulares en las cuales tendrá efecto dicho link. Los mockups realizados se pueden compartir y que distintos colaboradores agreguen comentarios, y se pueden exportar solamente las imagenes por

separado o todas juntas en un PDF, por lo que quedó descartada.

SnapUp http://www.quickfocus.com/

Herramienta web de código cerrado para diseñar diagramas de mockup web y móviles. Posee una versión beta gratuita y una diversa variedad de componentes. Dentro de las principales desventajas encontramos que no exporta diseños en otros formatos que no sean png, y no maneja eventos de navegación, por lo que dicha herramienta quedaría descartada.

Solidify http://www.solidifyapp.com/

Herramienta de plataforma web de código cerrado, con una buena variedad de componentes disponibles sin posibilidades de una versión gratuita. Si bien la suma de U$S 20 mensuales no resulta tan cara en comparación con otras, pero al no tener versión gratuita consideramos que no valía la pena suscribirse a una tarifa mensual para poder considerar si la herramienta era válida.

Uxpin https://uxpin.com/

Es una herramienta web, la cual permite realizar mockups para Web, Móvil y Desktop. Es de código cerrado y la versión gratuita de prueba que te brinda dura 35 hs. No brinda una gran cantidad de componentes para utilizar, pero si soporta una gran cantidad de eventos para generar interacciones entre páginas y elementos. Los diseños realizados se pueden exportar como PDF, PNG y HTML, pero el HTML generado es poco intuitivo y utiliza scripts propios. Por lo tanto esta herramienta no resultará útil para el proyecto a realizar.

Velositey http://dandkagency.com/extensions/velositey/

En el sitio de velocity anuncian que la herramienta ha sido discontinuada. Igual no nos era de utilidad ya que era un plugin para photoshop.

Visio https://products.office.com/es/Visio/flowchart-software

Herramienta de windows de escritorio, que pertenece a la gama de aplicaciones de Microsoft Office. Es de código cerrado y no posee versión gratuita. Dentro de los aspectos positivos de esta herramienta, se encuentra su variedad de componentes y su usabilidad. Sin embargo en cuanto al manejo de eventos, solamente soporta hipervínculos, lo cual si bien es válido para nuestro proyecto, no posee manejo para otros eventos requeridos. Otro de los aspectos a mencionar es que exporta los diagramas en un formato vsd con opción a xml, sin embargo dicho xml no resulta muy intuitivo por lo que no sería muy útil.

Visual Paradigm http://www.visual-paradigm.com/

Herramienta de windows que permite desarrollar mockups de interfaces web, de escritorio, y móviles. Es de código cerrado y posee una versión gratuita de 30 días de duración. Permite exportar los diagramas realizados en HTML y maneja una cantidad regular de componentes. No soporta navegación por lo que éste es un punto bajo para considerarla como buena candidata. Otro punto a destacar es que el HTML generado no genera elementos HTML, sino que inserta una imagen del diagrama realizado en una página web, sin ninguna posibilidad de manipular los componentes, razón por la cual quedaría descartada.

Weld https://www.weld.io/

Weld es una excelente aplicación para generar páginas web estáticas, como portfolios, sitios de presentación o landing pages. Permite generar los sitios mediante drag and drop de componentes predefinidos, y luego da la opción de publicarla bajo un dominio de la propiedad del usuario. Lamentablemente

para nuestro trabajo la herramienta no es de utilidad.

wireframe sketcher http://wireframesketcher.com/

Herramienta multiplataforma de código cerrado bastante libre a la hora de elegir un tipo de aplicación para maquetar. Posee una buena documentación, el único tipo de evento que soporta es la navegación entre páginas y posee distintos niveles de licenciamiento. Exporta los diagramas en una desconocida la cual inspeccionando la misma se ve claramente una estructura xml en cada página creada con una estructura y referencias a cada elemento bastante parseable, por lo cual si bien no tiene una versión gratuita permanente, podría ser considerada para posteriores etapas del proyecto.

wireframe.cc http://wireframe.cc/

Es una herramienta web, que te permite realizar mockups tanto para Web, Desktop y dispositivos móviles. Es de código cerrado y paga, te brinda una versión free, la cual te permite tener solamente una página, sin poder agregar links ni poder exportar. En el caso de poder exportar, esto se podría hacer a los formatos PDF y PNG, por lo tanto la descartamos como herramienta a tener en cuenta para el proyecto.

Wirefy http://getwirefy.com/

Es una herramienta que te permite realizar mockups escribiendo código, por lo cual se requieren conocimientos previos de HTML, CSS, Node, JS, etc. Por lo que la dificultad que implica realizar un diseño en esta herramienta no la consideramos apropiada para para nuestro proyecto.

Wiremagic http://wuwacorp.com/wiremagic/

Herramienta multiplataforma de código cerrado con diversos niveles de licenciamiento. A diferencia de otras herramientas ésta no provee una versión gratuita para poder evaluar la misma, es por esta razón que dicha herramienta fue descartada.