UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Desarrollo de herramienta para la prevención de Smells de UX en prototipos tempranos

AUTORES: Farinella Robertino - Fuentes Agustin

DIRECTOR/A: Dra. Alejandra Garrido

CODIRECTOR/A: Dr. Andrés Rodriguez

ASESOR/A PROFESIONAL: -

CARRERA: Licenciatura en Sistemas

Resumen

El desarrollo de software se destaca por la adopción generalizada de metodologías ágiles. Estas metodologías aceleran la entrega de valor al cliente, mejorando la eficiencia y velocidad del equipo de trabajo. Sin embargo, esta evolución constante a menudo lleva a la falta de tiempo para detallar el diseño propuesto, generando conflictos y "malos olores" de diseño. Para abordar estos desafíos, se realizó un plugin de Figma que brinde orientación a los diseñadores y mejore la especificación del prototipo, evitando ambigüedades y Smells de UX. El objetivo es facilitar la colaboración entre diseñadores y desarrolladores en un contexto ágil.

Palabras Clave

Diseño, usabilidad, experiencia de usuario, maquetado de interfaces, métodos ágiles, UX smells, UX Debt

Conclusiones

Creamos una Herramienta para Optimizar la Colaboración entre Diseñadores y Desarrolladores. Nuestra solución tecnológica innovadora supera desafíos comunes al destacar la importancia del diseño de interfaces y UX, al mismo tiempo que promueve una comunicación más efectiva entre ambos equipos. Este proyecto, de naturaleza libre y abierta, permite a cualquier persona extenderlo y mejorar la especificación del diseño interactivo, evitando ambigüedades y promoviendo las mejores prácticas en el desarrollo de interfaces.

Trabajos Realizados

Se realizaron las siguientes actividades:

- Estudio y entrevista preliminar para determinar los requerimientos y escenarios de uso de nuestra herramienta.

- Definición de un lenguaje simple de especificación de decisiones de diseño de interfaces en base a acciones, precondiciones, postcondiciones y restricciones.

- Desarrollo de un plugin de Figma que permite anotar diseños con el lenguaje definido.

Trabajos Futuros

- *Chequear automáticamente si hay UX smells.*
- *Importar/Exportar el diseño en otro entorno del trabajo.*
- *Exportar los tips a un documento de especificación de UX.*
- *Asociar la cuenta de Figma dentro de nuestra herramienta*
- *Comunidad de tips.*

Fecha de la presentación: Noviembre 2023

Agradecimientos

Queremos expresar nuestra sincera gratitud a todas las personas que hicieron posible nuestra tesina.

A nuestras familias, amigos y profesores, gracias por su apoyo incondicional y paciencia a lo largo de estos años de aprendizaje y crecimiento.

Alejandra Garrido y Andrés Rodríguez, nuestros directores y guías a lo largo de esta tesina, merecen un agradecimiento especial por su orientación constante.

A todos los que colaboraron en nuestra formación, su participación fue esencial en nuestra experiencia en la carrera.

CAPÍTULO 1

Introducción

1.1 Motivación

Hoy en día la mejor opción para el desarrollo de software es el uso de metodologías ágiles. Estas metodologías se caracterizan por mejorar la velocidad y eficacia del equipo de trabajo, y permitir entregas parciales con valor para el cliente. En cada entrega parcial, el producto se va modificando de acuerdo con el feedback obtenido, aumenta su funcionalidad y calidad en cada iteración. La calidad del diseño se refiere a las características que los diseñadores especifican para un producto [Pressman10]. En este trabajo se plantea un acercamiento ágil para el desarrollo de software donde se considera que tanto el diseño del producto como la implementación son actividades centrales en el proceso de software [Sommerville16].

A su vez, esta evolución que va dando forma al producto final requiere de muchas modificaciones y eso equivale a tiempo de trabajo; ese tiempo generalmente, se invierte en el área de desarrollo y se deja de lado el diseño propuesto, lo cual conlleva a “malos olores” de diseño (también denominados en la literatura como *Smells de UX* [GGRR17]) y eso puede generar conflictos entre el diseñador y el área de desarrollo, como también una demora en el desarrollo del producto [DaSilva11, Brhel15]. Los code smells son un conjunto de malas prácticas de codificación que pueden ser resueltos a través de refactorings. Los refactorings son transformaciones de código en el cual su objetivo es mejorar la calidad del mismo sin modificar la funcionalidad. Un smell de UX se puede definir como un indicio de una experiencia de usuario deficiente. [LGG]

El concepto de deuda técnica que se utiliza en el ámbito del desarrollo de software, establecido por Ward Cunningham, consiste en hacer compromisos técnicos que pueden ser convenientes a corto plazo pero que con el tiempo generan dificultades y son más costosos de arreglar [Cunningham92]. Sucede algo similar respecto en el diseño creándose deudas que en vez de ser técnicas son de usabilidad y UX, se realiza un diseño sencillo para que funcione en el corto plazo dejando esta deuda

para el largo plazo [DaFonseca19, Baltes21]. En casos severos, la deuda de UX puede conducir a la renovación de la infraestructura y la renovación completa de los productos [SRP19].

El prototipado incremental es utilizado por los diseñadores de UX como técnica para implementar sus propuestas. Comienzan con prototipos de baja fidelidad, a medida que se van aprobando las opciones del diseño de los prototipos de baja fidelidad, se van aumentando los detalles y así logran alcanzar prototipos de media y alta fidelidad (muy similares al producto final esperado).

Una de las herramientas más usadas por los diseñadores para el prototipado de media y alta fidelidad es Figma. Es una aplicación que brinda todas las herramientas necesarias para poder desarrollar la fase de diseño de un proyecto, donde se encuentran las herramientas vectoriales capaces de ilustrar como también aquellas para la creación de prototipos y la generación de código CSS, IOS y Android [Figma]. Utilizar esta herramienta, hace que el equipo de desarrollo se concentre en las funcionalidades del producto y el diseñador en la parte UX y UI. Los diseñadores de UI/UX a menudo tienen que diseñar y rediseñar conceptos hasta que se llega a un consenso. Las iteraciones de diseño de cada fidelidad y reelaboración involucradas en la transformación de una fidelidad a otra pueden hacer que todo este proceso sea extenuante [SPSJ19]. También existen otras herramientas de diseño, como Adobe XD y Sketch, entre las más populares.

Tomemos una situación en la cual un diseñador utiliza Figma, donde plasma todo el diseño del proyecto para que luego, el desarrollador, pueda tomar este diseño, y así aplicarlo en la fase de desarrollo. Este procedimiento hace que puedan quedar aspectos del diseño sin detallar y el desarrollador decida resolverlas sin consultar con el área de diseño. Por ejemplo, al completar un formulario, puede que el diseñador haya hecho el diseño para que se vayan habilitando o deshabilitando distintos campos de acuerdo al input del mismo.Otro caso podría ser que al presionar un botón se habilite otro botón, etc. Si esas restricciones no fueron indicadas explícitamente por el diseñador, esta situación generará un conflicto entre el área de diseño y desarrollo por lo que se deberá invertir más tiempo en corregir estas diferencias. Como el equipo de desarrollo estará más concentrado en seguir

desarrollando las funcionalidades del producto y aumentando su calidad, puede que este problema se extienda días, semanas, incluso meses. El diseñador es el encargado de especificar su diseño correctamente. Un aspecto clave es que los equipos de desarrollo ágil comprendan que el diseño de la experiencia del usuario no es un rol, sino una disciplina y una mentalidad [SRP19] [GS21].

Para encontrar más información sobre la interacción entre los diseñadores UX y los desarrolladores en la práctica, nos propusimos realizar una entrevista (detallada en el 3.1.1) a un diseñador con experiencia en el área y que a su vez, utilice la herramienta Figma.

Como posible solución al problema anteriormente planteado, pensamos en desarrollar una extensión de Figma que pueda guiar al diseñador para especificar su prototipo con el fin de evitar ambigüedades y la posible generación de Smells de UX en el ambiente de desarrollo, ya que no se encuentran herramientas que cumplan este objetivo. Esto ayudará tanto al diseñador que quizá no tiene la experiencia necesaria como así también al desarrollador para entender mejor el diseño. Dentro de la herramienta, se dispondrá de tips de uso genéricos como así también restricciones.

1.2 Objetivos

El objetivo principal de la propuesta es darle al diseñador la posibilidad de especificar las funcionalidades que conlleva el diseño, por ejemplo las pre y poscondiciones para una acción determinada (tips), las restricciones de un campo, etc. Cuando nombramos “tip” nos referimos a una decisión que el diseñador agrega en la interfaz, en las partes interactivas, con el fin de evitar malos olores de UX y ambigüedades en el diseño.
Con esto se intenta simplificar el trabajo del área de desarrollo y a su vez, evitar desacuerdos entre el diseñador y el grupo de desarrollo, ahorrando el tiempo de un posible rediseño de las interfaces por una mala especificación. Este objetivo busca alinear el trabajo de los diseñadores y desarrolladores con las reglas doradas para el diseño de interfaces de usuario propuestas por Theo Mandel, en su libro [Man97], especificó tres reglas doradas para el diseño de la interfaz de usuario:

1. Dejar el control al usuario.
2. Reducir la carga de memoria del usuario.
3. Hacer que la interfaz sea consistente.

De este objetivo principal, se desprenden dos objetivos secundarios:

- Uno es permitir al diseñador especificar todas las acciones posibles junto con todas las precondiciones y postcondiciones aplicables a un desarrollo de UX/UI, con el fin de brindarle al desarrollador la mejor especificación posible.
- El segundo es facilitar el trabajo centrado en la UX sin necesidad del retrabajo, evitando malgastar tiempo en el rediseño de las interfaces debido a una falta de especificación.

Luego de leer y analizar los diferentes trabajos científicos, investigar sobre las metodologías ágiles, los smells de UX, las causas y consecuencias de los mismos, y también analizando la entrevista realizada, concluímos que casi siempre había un denominador común: la falta de comunicación entre el área de desarrollo y el área de diseño. A su vez, en estas metodologías el tiempo está centrado en seguir mejorando o agregando funcionalidades, dejando de lado las buenas prácticas de diseño, como por ejemplo: tener una buena interfaz o una buena experiencia de usuario. Por ende se brinda muy poco tiempo para un rediseño con el fin de corregir los smells de UX.

Por estas razones, el fin de nuestra herramienta es evitar esos smells de UX por falta de comunicación, donde se tendrá un diseño más detallado para facilitar el trabajo al desarrollador y así, en caso de un posible rediseño, no llevaría el mismo tiempo que si no se hubiese utilizado el plugin, es decir, administraría mejor el tiempo de rediseño, se optimiza.

Buscamos comprender mejor el proceso de prototipado de UX y su transferencia al desarrollo, para poder brindar una herramienta para el prototipado de interfaces de

usuario que pueda facilitar la especificación de acciones del usuario con sus precondiciones y poscondiciones.

Trabajamos en una de las plataformas de prototipado de mayor difusión en la industria como Figma, la cuál la describimos en la motivación. Para facilitar la creación y edición de los prototipos, proponemos que todo el proceso se desarrolle dentro de Figma, para lo cual desarrollamos una extensión dentro del framework propuesta por Figma con la funcionalidad necesaria. Entonces, hemos desarrollado una herramienta que permite especificar la acción que puede tomar un elemento interactivo y, junto a ella, sus precondiciones y postcondiciones; el diseñador simplemente debe seleccionar la acción que le parezca más conveniente y en caso que quiera modificarla a su gusto, pueda hacerlo. Para ello, creamos una interfaz de usuario para que el diseñador pueda interactuar de una forma cómoda con la herramienta. A su vez, interactuamos con la API de Figma para poder recuperar el diseño y listar las acciones disponibles.

Un posible esquema de trabajo podría proveer al diseñador una librería de Bootstrap, que es una de las más utilizadas hoy en día en el mundo del desarrollo de software. Luego que determina qué componentes de la librería desea emplear en su prototipo, invoca nuestra extensión. Dependiendo del nombre del elemento, nuestro plugin listará acciones que tengan sentido para dicho elemento, por ejemplo, para un componente llamado “input” uno de los posibles tips sería la acción: Ingresar contenido (una opción que no aparecerá será la de acción: Click).
Una vez que el diseñador elige la acción general, se listarán distintas precondiciones y postcondiciones de acuerdo a la acción elegida, por ejemplo: Si se elige ingresar contenido habrá: precondiciones y postcondiciones para la opción email, precondiciones y postcondiciones para la opción nombre, etc. Con esto quedará especificado el elemento que tendrá interacción en el desarrollo. A su vez, en caso que no aparezca el tip deseado, el diseñador tendrá total libertad de poder editar cualquier precondición y postcondición de un tip para luego tener la opción de crear un comentario o copiarlo manualmente.

1.3 Logros alcanzados

Al finalizar el desarrollo se dispone de una herramienta en el entorno de Figma que permita una mejor especificación de cada elemento presente en un prototipo, siendo capaz de listar las acciones que pueda implementar el elemento junto con sus diversas precondiciones y postcondiciones, como así también se puede relacionar un elemento con otro, detallando por ejemplo, si el primer elemento debe estar habilitado o no, completo o no, para que el segundo se habilite. Cabe destacar, que también se puede especificar restricciones a cada elemento, como por ejemplo el tipo o la longitud del mismo. Se busca que este plugin brinde buenas prácticas de diseño, donde la idea sería especificar el funcionamiento de los elementos interactivos con el fin que cuando el desarrollador recibe el diseño, no quede lugar a las ambigüedades y por ende no habrá, o no tendría que haber, un malentendido entre el diseñador y el desarrollador. Con esto se evitan los conflictos y a su vez, se ahorraría tiempo de rediseño, es decir, se optimiza el proceso. Por último, se realizó una validación preliminar, con el objetivo de testear la herramienta con diseñadores reales y evaluar las mejoras generadas en el proceso de trabajo entre el diseño UX y su codificación.

1.4 Organización de la tesina

Capítulo 2

En este capítulo expondremos el marco teórico de esta tesina, donde presentamos trabajos relacionados que fueron estudiados y analizados para la realización de esta tesina. A su vez, se definen conceptos esenciales.

Capítulo 3

En este capítulo se detalla una entrevista realizada con un profesional del área de diseño, junto con la especificación de requerimientos y una explicación básica del funcionamiento de la herramienta con los elementos que la componen.

Capítulo 4

En este capítulo explicamos las tecnologías utilizadas para el desarrollo de la herramienta, junto con el funcionamiento relacionado a Figma y la obtención del token en Figma para realizar determinadas funcionalidades.

Capítulo 5

En el capítulo 5 se desarrolla la nomenclatura a respetar para obtener el máximo provecho al plugin, también se especifica cada funcionalidad junto a un ejemplo de uso.

Capítulo 6

En el capítulo 6 se especifica sobre la validación realizada de nuestra herramienta, realizada por un diseñador profesional. A su vez, se detallan las conclusiones de la misma.

Capítulo 7

En el capítulo final se habla sobre las conclusiones de esta tesina, las contribuciones que brinda este plugin, sus limitaciones y, además, posibles trabajos futuros.

CAPÍTULO 2

Marco Teórico y trabajos relacionados

2.1 Desarrollo de software con métodos ágiles

En los últimos 20 años, el uso de internet se convirtió en una necesidad, pudiendo así interactuar con aplicaciones que nos comenzaron a facilitar nuestra vida cotidiana, como es el ejemplo de Pedidos Ya o Rappi, pudiendo comprar cualquier cosa y que nos lo traigan a nuestros hogares, plataformas como Netflix que nos brindan una catálogo enorme de películas y series vía streaming, las redes sociales que nos acercan a las vidas de las personas, entre otras.

Las metodologías ágiles fueron ganando lugar y lograron predominar en el desarrollo de software, siendo la metodología que mejores resultados dió, superando a diferentes modelos como el modelo en espiral, el modelo de prototipos, el modelo en cascada, entre otros [Sommerville11].

Estas aplicaciones o páginas web son productos de software, en el que su proceso de desarrollo fue variando y evolucionando con el paso del tiempo, hay diferentes procesos de desarrollos debido a que cada equipo o grupo de trabajo adoptará su propio proceso de desarrollo adecuándose a sus objetivos y recursos que tengan disponibles, donde influyen varios factores como por ejemplo: la cantidad de personas que integran el equipo, el tiempo disponible para desarrollar el producto, el tipo de cliente, entre otros; llegando a predominar el uso de **metodologías ágiles** [Ashmore 14], debido a los buenos resultados que esta conlleva. Se destaca de estas metodologías su adaptación a los cambios, y principalmente, la entrega de pequeños productos de software funcionales en cortos períodos de tiempo. Estas incorporaciones pueden ser funcionalidades nuevas, errores corregidos y/o peticiones del cliente.

Algunas de las características de las metodologías ágiles por las que estas metodologías quedaron como la mejor opción al momento de realizar un desarrollo fueron:

- La metodología ágil es rápida, específica y dinámica.
- Estimula las actitudes y estructuras del equipo, pues hace más fácil la comunicación.
- Considera al cliente como parte del equipo de producción.
- Las entregas son tempranas y continuas.
- Su estructura cambia según la competencia.
- La mayoría de las conversaciones son cara a cara.
- La gente de negocios y desarrolladores trabajan siempre juntos.
- Sus acciones son ajustables y simples.

A pesar de que los tipos de metodologías ágiles se renuevan constantemente, a continuación se expone la más utilizada.

2.2 Scrum

Este método se dedica al desarrollo del producto con el objetivo de que progrese de manera constante. En vez de basarse en una planificación completa, su desarrollo se revisa periódicamente. Además, este método ágil permite al desarrollador trabajar en base a los resultados de su trabajo, por lo que este se mantiene comprometido con cumplir determinada tarea y tiene flexibilidad laboral, lo cual a su vez le causa motivación.

2.2.1 El proceso

En Scrum [Pressman10] un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos casos son de 3 y hasta 4 semanas, límite máximo de feedback de producto real y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente (Product Owner) prioriza los objetivos balanceando el valor que le aportan respecto a su coste (que el equipo estima considerando la Definición de Hecho) y quedan repartidos en iteraciones y entregas.

Las actividades que se llevan a cabo son las siguientes:

Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración. Consta de dos partes:

Selección de requisitos (2 horas):El cliente presenta al equipo la lista de requisitos priorizada del producto. El equipo consulta al cliente las dudas que surgen y establece los requisitos más prioritarios que prevé que podrá completar en la iteración, de manera que puedan ser entregados si se solicitan.

Planificación de la iteración (2 horas):El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se asignan las tareas, se organizan para trabajar en grupos con el fin de compartir conocimiento o para resolver juntos objetivos especialmente complejos.

Ejecución de la iteración

Cada día el equipo realiza una reunión de sincronización (15 minutos), generalmente delante de una pizarra (Scrum Taskboard). El equipo analiza el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con la previsión de objetivos a mostrar al final de la iteración. En la reunión cada miembro del equipo responde a tres preguntas:

1. ¿Qué he hecho desde la última reunión de sincronización para ayudar al equipo a cumplir su objetivo?
2. ¿Qué voy a hacer a partir de este momento para ayudar al equipo a cumplir su objetivo?
3. ¿Qué impedimentos tengo o voy a tener que nos impiden conseguir nuestro objetivo?

Durante la iteración el Scrum Master se encarga de que el equipo pueda mantener el foco para cumplir con sus objetivos.

- Elimina los obstáculos que el equipo no puede resolver por sí mismo.
- Protege al equipo de interrupciones externas que puedan afectar el objetivo de la iteración o su productividad.

Durante la iteración, el cliente junto con el equipo refinan la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian los objetivos del proyecto con el objetivo de maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración, la cual conlleva dos partes:

1. Revisión (demostración): El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente puede realizar las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.
2. Retrospectiva: El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Scrum Master se encargará de eliminar o escalar los obstáculos identificados que estén más allá del ámbito de acción del equipo.

2.3 Diseño de la interfaz de usuario y la experiencia de usuario

Un acercamiento ágil para el desarrollo de software es el desarrollo como la actividad central en el proceso de software [Sommerville16], teniendo solo en cuenta las funcionalidades del software. A su vez, el proceso que va dando forma al producto final, requiere de muchas modificaciones y eso equivale a tiempo de trabajo, ese tiempo generalmente se invierte en el área de desarrollo y se deja de

lado el diseño. Esto puede generar conflictos entre el diseñador y el área de desarrollo, como también un producto de menor calidad. [DaSilva11, Brhel15].

El diseñador juega un papel muy importante en el desarrollo de software, ya que es el que se encarga de diseñar la interfaz con la cuál el usuario se comunicará e interactuará con el sistema [Presman10], es en donde se implementa la usabilidad y experiencia de usuario.
Normalmente estos dos atributos de calidad del sistema se dejan de lado debido a que solo se invierte tiempo en la funcionalidad del sistema. Esto genera que en etapas más avanzadas del sistema, cuando se quiera mejorar la usabilidad y UX del producto, lleve más tiempo o requiera mucho más esfuerzo.

Por ende será muy importante que el diseñador que utiliza herramientas de diseño como es el ejemplo de Figma, Sketch o Adobe XD, entre otras, especifique aquellos elementos donde se mejoran la interacción que se requiere para cada elemento de la interfaz y las decisiones de diseño, por ejemplo: habilitar ciertos botones de acuerdo al ingreso de información en un campo, y el tipo de interacción permitida. Si estas condiciones no fueron indicadas explícitamente, esta situación generará un conflicto entre el área de diseño y desarrollo por lo que se deberá invertir más tiempo en corregir estas diferencias. Como el equipo de desarrollo estará más concentrado en seguir desarrollando las funcionalidades del producto y aumentando su calidad, puede que este problema se extienda días, semanas, incluso meses, cabe destacar que la calidad del diseño se refiere a las características que los diseñadores especifican para un producto [Pressman10]. Un aspecto clave es que los equipos de desarrollo ágil comprendan que el diseño de la experiencia del usuario no es un rol, sino una disciplina y una mentalidad [SRP19] [GS21].

Consideramos que la calidad y el tiempo son muy importantes en el desarrollo de software, por lo que planteamos una herramienta para utilizar dentro del entorno de Figma donde se utilice como guía al diseñador para especificar su prototipo con el objetivo de poner fin a las ambigüedades que puedan surgir entre el diseño pensado por el diseñador y la implementación del mismo en el área de desarrollo. Este sistema mejorará la comunicación entre el diseñador y el área de desarrollo, ahorrando tiempo y ganando armonía.

2.4 Trabajos relacionados

A continuación se describen trabajos de investigación relacionados que describen las problemáticas que se intentan resolver con nuestro plugin.

2.4.1 Deuda técnica de usabilidad

En un estudio [DaFonseca19] se demostró mediante el seguimiento de 5 proyectos diferentes que existen deudas técnicas, concepto establecido por Ward Cunningham [Cunningham92], en el proceso de desarrollo de un software. Estas deudas técnicas pueden ser de diferentes tipos, en este momento nos interesa la deuda técnica de usabilidad.

De la misma podemos describir que:

- Es frecuente
- Viola principios de interacción con el usuario, lo que conlleva a afectar negativamente la satisfacción con el usuario.
- Requiere un esfuerzo de baja resolución para resolverla.

A su vez, se demostró que la DT de usabilidad ocurrió en todos los proyectos que se estudiaron, donde en los elementos de los proyectos violó 8 de 10 heurísticas de Jacob Nielsen, destacándose “Flexibilidad y eficiencia de uso" y "Prevención de errores". Cabe aclarar que este tipo de violaciones tiende a tener un efecto negativo en la satisfacción con el usuario. Se especifica que los elementos de deuda técnica de usabilidad requieren un esfuerzo de resolución bajo, que oscila aproximadamente entre el 5,1 % y el 6,7 % del esfuerzo total estimado de resolución de elementos de TD dentro de los proyectos analizados. Se recomienda identificar y eliminar los ítems de deuda de usabilidad generando un posible impacto positivo en el usuario.

Se seleccionaron y clasificaron 255 ítems de deudas técnicas distintas pero al final de la etapa de evaluación solo quedaron 145.

En este paper se hacen y responden 3 preguntas claves:

1. ¿Las deudas técnicas de usabilidad son frecuentes en el proceso de desarrollo del software?

El análisis confirmó que la deuda de usabilidad ocurría con frecuencia (oscilando entre el 10,4% y el 20,8% del total de ítems de deudas técnicas de los cinco proyectos), siendo además uno de los únicos cuatro tipos de deuda que ocurría en todos los proyectos.

2. ¿Qué tipos de deudas técnicas de usabilidad son las más frecuentes en el proceso de desarrollo del software?

Los elementos de usabilidad se clasificaron de acuerdo con las heurísticas de usabilidad de Jakob Nielsen [Nielsen94]. Estas heurísticas representan 10 principios generales para el diseño de interacción, las cuales son:

(1) Visibilidad del estado del sistema.

(2) Coincidencia entre el sistema y el mundo real.

(3) Control y libertad del usuario.

(4) Consistencia y estándares.

(5) Prevención de errores.

(6) Reconocimiento en lugar de recuerdo.

(7) Flexibilidad y eficiencia de uso.

(8) Estética y diseño minimalista.

(9) Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores .

(10) Ayuda y documentación.

Se observó que se violan ocho de diez de las heurísticas de usabilidad de Nielsen, diferentes tipos de deuda de usabilidad tienden a ocurrir durante los proyectos de desarrollo de software.

3. ¿Cuánto esfuerzo es necesario para resolver un ítem de la deuda técnica de usabilidad en el proceso de desarrollo del software?

Se observa que los elementos de deuda de usabilidad generalmente requieren un esfuerzo de resolución bajo.

Por lo tanto, desde la perspectiva de la gestión de deudas técnicas, quitar este tipo de deuda generalmente tiene un balance positivo, ya que elimina una gran cantidad de elementos de la cartera de TD de los proyectos con poco esfuerzo y, al mismo tiempo, ayuda a mejorar la experiencia del usuario para los sistemas en desarrollo.

La deuda de UX se crea de manera similar a la deuda técnica, en el punto donde se realizan compensaciones para las necesidades, normalmente funcionalidades, del proyecto [DaFonseca 19][Rodriguez23]. Si no se utiliza una disciplina de refactorización para abordar las compensaciones, surgen dificultades significativas. Dos de estas situaciones interrelacionadas pueden derivar a una deuda significativa en la experiencia de usuario que posteriormente es más costoso corregir.

- Cuando las malas experiencias de usuarios persisten en los productos de una organización, donde el impulso de una buena experiencia de usuario es menor que para los productos comercializados externos que se enfrentan a una competencia.
- Cuando una organización posee una lista de productos, cada cuál con su diseño. Puede ser debido a fusiones entre empresas, cada una con su propia marca de UX, lo que conlleva en casos severos a una renovación de la infraestructura y de los productos.

AgileUX es la etiqueta otorgada a los esfuerzos que tienen como objetivo resolver estas preocupaciones mediante la integración de técnicas y procesos del diseño de interacción y los de métodos ágiles. Hay una reflexión [DaFonseca 19, Baltes 21] donde habla sobre la evolución de agileUX concluye que la integración de metodología ágil y UX requiere una comprensión mutua del equipo en tres dimensiones:

1. Proceso y práctica
2. Personal y social

3. Tecnología y artefactos, la cual utiliza la tecnología para coordinar las actividades del equipo y los artefactos para mediar en la comunicación en los equipos.

Se concluye especificando que lo ideal es mantener un equilibrio adecuado que mantenga la investigación como la reflexión para un buen diseño UX, así como la integración del feedback de los usuarios como alternativas técnicas. Esto quiere decir que se necesita una constante comunicación entre el equipo de desarrollo y el equipo de diseño.

2.4.2 Smells de UX

Nuestra herramienta no se concentra en detectar los smells de UX, sino más bien en evitarlos, donde además, se busca mejorar la comunicación entre el diseñador y el desarrollador para poder evitar ambigüedades entre las áreas anteriormente mencionadas, es decir, se ganaría calidad en el software y se ahorraría tiempo de refactoring. El poder de esta técnica, denominada refactoring, se establece en ayudar a identificar problemas potenciales en el aspecto objetivo, donde a través de una serie de pasos generan una buena solución a esos problemas [GGRR17]. Estos problemas potenciales se denominan bad smells, también existen los smells usabilidad, de manera similar a los bad smells de código, donde estos se relacionan con cualquier aspecto de la calidad en el uso de la web, efectividad, eficiencia y satisfacción, es decir es un indicio de un diseño defectuoso de la interfaz que entorpece las tareas que el usuario final realiza.

A continuación describiremos brevemente los smells de UX más comunes [GGRR17]:

Undescriptive Element: (elemento no descriptivo)

Todo componente interactivo de una interfaz necesita de un significante claro, es decir alguna pista que muestre al usuario lo que se puede hacer con él (Norman, 2013). Un comportamiento normal consiste en colocar el mouse sobre el control, dejarlo quieto unos instantes, y esperar que aparezca un tooltip.

Misleading Link: (vínculo confuso)

Este olor es similar al elemento no descriptivo pero específico para los enlaces. Además de los intentos de información sobre herramientas en los enlaces, este olor también captura secuencias de eventos donde muchos usuarios navegan por un enlace sólo para regresar poco después, lo que indica que lo más probable es que el los contenidos enlazados no están correctamente representados por el nombre del enlace.

No Processing Page: (ausencia de página de proceso)

Este olor se detecta cuando los usuarios tienen que esperar demasiado para que se cargue una página sin comentarios.

Free Input For Limited Values: (campo de texto libre para valores limitados)

Este olor se activa cuando se usa una entrada de texto estándar para ingresar datos de un conjunto limitado de valores, como ciudades o países. Los usuarios tienen la posibilidad de escribir un texto libre cuando en realidad las opciones son limitadas.

Unformatted Input: (campo sin formato)

Este olor indica que se está utilizando una entrada de texto estándar para ingresar datos que tienen un formato determinado, como fechas, números de teléfono o emails.

Unmatched Input Size: (tamaño de campo inadecuado)

Es frecuente encontrarse con campos de texto que tienen un tamaño muy diferente de los valores para los cuales está diseñado. Este problema trae confusión al usuario, como lo indican autores como Wroblewski (Wroblewski, 2008).

Forced Bulk Action: (acción en lote forzada)

Es común que las aplicaciones web que muestran una lista de elementos (como productos) ofrezcan capacidades de acción masiva. Los usuarios generalmente realizan estas acciones seleccionando un grupo de elementos mediante casillas de verificación y luego seleccionando la acción, “Eliminar”, o “Mover a...” y finalmente

aplican la acción. Aunque esta interacción funciona bien cuando se aplica a un grupo de elementos, lleva demasiado tiempo aplicarla a uno solo. Cuando el caso más frecuente es este último, y los usuarios no disponen de una forma alternativa y más rápida de aplicar estas acciones, se detecta el olor Forced Bulk Action.

No Client Validation: (ausencia de validación en el cliente)

Este olor detecta formularios que tienen una alta tasa de rechazo, donde la validación ocurre en el lado del servidor.

Late Validation: (validación tardía)

Este olor es el mismo que el anterior con la diferencia de que es específico para el escenario donde la validación ocurre después de hacer clic en el botón "Enviar", pero el envío no ocurre.

Wrong Default Value: (valor por defecto incorrecto)

Este olor detecta cuando la opción más común para un conjunto de botones de radio no coincide con el valor predeterminado.

Unresponsive Element: (elemento inmutable)

Este se detecta cuando los usuarios hacen click reiteradamente sobre un elemento, pero éste no desencadena ninguna acción. Esto sucede cuando los elementos dan un incentivo para que los usuarios los presionen, debido a su apariencia.

2.4.3 Refactorización de los sistemas

En el artículo Refactoring for usability in web applications [GRD11] se mencionó que William Opdyke y Ralph Johnson introdujeron la refactorización a principios de los años 90, principalmente para reestructurar la jerarquía de clases de un diseño orientado a objetos. Un tiempo después, Martin Fowler la popularizó, el cuál definió la refactorización como "un cambio realizado en la estructura interna del software para hacerlo más fácil de entender y más barato de modificar sin cambiar su comportamiento observable". Desde ese momento, la refactorización se aplicó a diversos artefactos de software, como por ejemplo, modelos del Lenguaje Unificado de Modelado (UML), bases de datos y documentos HTML. La filosofía básica de la

refactorización consiste en que cada refactorización es una leve transformación que preserva el comportamiento, sigue manteniéndose pero su intención cambio de acuerdo a la mejora de legibilidad, extensibilidad y mantenimiento del código fuente.

En el artículo se cree que es importante vincular cada refactorización no sólo a los "malos olores" que puede eliminar, tomando prestada la metáfora de Kent Beck y Fowler sobre los síntomas de los problemas del código, sino también a los atributos específicos de calidad que se pretende mejorar.

Por ejemplo, una refactorización HTML, como "Activar autocompletar", no mejora ninguna cualidad interna del código, pero hace que un formulario web sea más fácil de usar y, por lo tanto, cambia la intención de la refactorización hacia la mejora de la usabilidad del producto de software.

2.4.4 Granularidad de los prototipos de diseño y su carga de trabajo.

El artículo Eve: A Sketch-based Software Prototyping Workbench [SPSS J19] habla sobre la creación de prototipos, que es un método muy utilizado para diseñar y evaluar interfaces de usuario. Consta de tres fidelidades: baja (LoFi), media (MeFi) y alta (HiFi), cabe destacar que cada finalidad hace referencia al nivel de madurez del prototipo. Este proceso de creación posee varias etapas, pasando de un concepto en un borrador hasta convertirlo en un diseño que pueda ser evaluado por los usuarios. Este proceso es iterativo y, por tanto, lleva tiempo alcanzar un nivel de madurez.

Como en cualquier otro proceso creativo, los diseñadores de UI/UX a menudo tienen que diseñar y rediseñar conceptos hasta llegar a un consenso. Las iteraciones de diseño de cada fidelidad y el trabajo que supone transformar una fidelidad en otra pueden hacer que todo este proceso resulte agotador. Por ello, es importante que no haya ambigüedades a la hora de diseñar entre el área de diseño y el área de desarrollo.

En esta investigación proponen investigar la creación de prototipos de software desde la perspectiva de la carga de trabajo. El término carga de trabajo consiste no sólo en la cantidad de trabajo que hay que realizar durante la tarea, sino también al nivel de carga física y cognitiva que experimenta la persona.

En el artículo, se estudió la carga de trabajo subjetiva experimentada por los diseñadores de interfaz de usuario/UX durante la creación de prototipos de software. Para ello, realizaron un estudio de análisis de la carga de trabajo con 18 participantes utilizando el NASA Task Load Index (NASA-TLX). Los participantes crearon un prototipo de un concepto determinado y debían informar la carga de trabajo experimentada para cada fidelidad. La carga de trabajo media experimentada por los participantes aumentaba con cada fidelidad de prototipado.

Se deja constatado un aumento de la frustración, la demanda temporal y el esfuerzo, mientras que el rendimiento disminuye a medida que los participantes pasan de una fidelidad baja a una alta. Comparativamente, MeFi es físicamente más exigente y mentalmente menos exigente que LoFi y HiFi. Nuestra herramienta tiene el objetivo de disminuir la carga de trabajo en base a las horas “malgastadas” por un rediseño que surge de la mala comunicación o falta de la misma entre el diseñador y el desarrollador. De esta manera, ese tiempo perdido podría invertirse en mejorar otros aspectos funcionales y por lo tanto, el diseñador aumentaría su rendimiento.

2.4.5 Enfoques para el trabajo del diseño

En el artículo Integrating Agile and User-Centered Design [JHM14] se habla sobre la integración ágil y el diseño centrado en el usuario (DCU). Afirma que las numerosas ventajas de la ingeniería de software ágil la han convertido en una metodología de desarrollo de uso generalizado. Aunque, ágil por sí sola no se ocupa necesariamente de la usabilidad del producto de software. A su vez, la necesidad de una buena experiencia de usuario (UX) se ha hecho más evidente, por lo que se han realizado esfuerzos para integrar las prácticas de usabilidad del diseño UX en ágil.

Establece que un enfoque común de desarrollo de UX es el DCU. El objetivo del DCU es aumentar la usabilidad, la medida en que un usuario encuentra una interfaz fácil de navegar, y el término "usabilidad" también puede referirse a los métodos que pueden utilizarse para mejorar el diseño de la interfaz de una aplicación. La usabilidad es un factor importante a tener en cuenta en cualquier producto. Por ejemplo, cuando los usuarios ingresan a un sitio web y encuentran una dificultad para utilizarlo, es probable que lo terminen abandonando. Por lo tanto, muchos profesionales, desde el comienzo de la metodología ágil, se han sentido motivados para encontrar las mejores formas de integrar las prácticas de usabilidad en los productos que se desarrollan a través de ágil.

La prioridad de ágil es la satisfacción del cliente, que se logra mediante entregas iterativas de pequeños conjuntos de funciones al cliente. Este enfoque en la funcionalidad puede ir en deterioro de la usabilidad, ya que ágil hace hincapié en el mínimo trabajo de diseño previo, mientras que los diseñadores de UX lo consideran esencial. Sin embargo, dado que ágil hace énfasis en la entrega de fragmentos verticales de funcionalidad, esto significa que se pueden entregar características funcionales a los clientes para obtener comentarios frecuentes, que es un concepto clave en el DCU. Por lo que los clientes tienen la oportunidad de aceptar características o solicitar cambios en el diseño desde el principio. Por lo tanto, es importante reducir el tiempo de rediseño para poder lograr una satisfacción equilibrada tanto para los diseñadores UX como para los desarrolladores. Uno de los objetivos de la herramienta, sería invertir mejor el tiempo que se le otorga al proyecto con respecto al diseño, reduciendo el tiempo invertido en rediseñar.

Las prácticas sistemáticas del DCU diseñadas para mejorar la usabilidad del producto no se emplean durante el ciclo de desarrollo ágil. Una de las ventajas de realizar sprints para el cliente es que los comentarios de los usuarios se reciben antes, por lo que pueden utilizarse para corregir fallos de usabilidad.

Frente a las funcionalidades, la prioridad del DCU y la experiencia del usuario es la satisfacción del usuario. Para esto, al principio del proyecto se destinan importantes recursos a la investigación exhaustiva del usuario. Después le siguen iteraciones de diseño que consisten en la creación de prototipos y la evaluación, pero las

iteraciones son más largas que un típico sprint ágil. El diseño UX hace hincapié en métodos especializados de investigación del usuario final antes de fabricar el producto.

El diseño ágil y la UX comparten el objetivo de producir software de alta calidad, aunque lo abordan desde perspectivas diferentes. Uno de los problemas de la integración ágil y el diseño de la UX es la sincronización de sus actividades y prácticas. La colaboración entre los diseñadores de UX, los desarrolladores ágiles y otros equipos, por ejemplo el área de Marketing, debe mejorarse mediante una gran cantidad de comunicación. La comunicación entre los diseñadores de UX y los desarrolladores es muy importante porque cada grupo tendrá prioridades, objetivos y procesos diferentes aunque la integración de ambos grupos tiene sus ventajas. Por esta razón, nuestra herramienta brindará tips para crear un diseño más detallado y menos ambiguo.

Una revisión sistemática previa realizada por Silva da Silva en 2011 encontró algunas tendencias interesantes sobre los trabajos en este campo. El interés en ágil-UX ha aumentado en general con el tiempo desde la creación de ágil. Existen dos enfoques de trabajo para el diseño de UX: el especialista y el generalista.

Little Design Up Front (LDUF) fue la práctica más común utilizada en los estudios primarios de ágil-UX que Silva da Silva recopiló. El LDUF reduce, pero no elimina, la gran cantidad de trabajo de diseño realizado a través del DCU al principio del proyecto para poder dedicar más esfuerzo a la funcionalidad. Esta práctica posee una orientación similar a nuestro plugin, con el fin de que se pueda eliminar el tiempo de rediseño para poder concentrar esa “ganancia” en otra etapa del proyecto.

Otra práctica común es la estrecha colaboración entre el equipo ágil y el equipo de UX. La colaboración puede aumentar el éxito de un proyecto al mejorar la comprensión de lo que se supone que debe ser el proyecto entre los dos equipos.

También se suele sugerir que el diseño de UX debe realizarse en paralelo al desarrollo ágil. El problema de este enfoque es que el equipo debe garantizar

activamente la colaboración entre desarrolladores y diseñadores, ya que un proceso paralelo dificulta la comunicación entre las partes. Los mockups pueden llegar a ser una herramienta que brinda colaboración entre ambos equipos, y es importante que el mismo esté perfectamente diseñado y detallado para evitar errores de comprensión entre estas áreas. Nuestra herramienta serviría para aumentar la especificación del diseño. Los análisis que se hicieron sobre la integración de ágil y el DCU señalan que faltan estudios sólidos y controlados, así como de estudios en general.

Uno de los principales problemas que se identificaron con la integración ágil-UX entre los diferentes estudios fue que el diseñador UX estaba sobrecargado de trabajo y excesivamente distribuido entre los equipos ágil.

2.4.6 Utilizar artefactos como medio comunicacional

Artifact-Facilitated Communication in Agile User-Centered Design [GDSS19] es un artículo que habla sobre uno de los principales retos a los que se enfrenta la integración del diseño ágil y el diseño centrado en el usuario es cómo facilitar la comunicación entre los distintos profesionales implicados. Los artefactos facilitan la comunicación y los artefactos apoyan la colaboración.

El diseño ágil centrado en el usuario (DACU) ha evolucionado por distintos motivos. Por un lado, ágil quiere satisfacer a los clientes mediante lanzamientos puntuales a las peticiones de cambio sin comprometer la calidad del software. Por otro lado, el DCU pretende garantizar una usabilidad adecuada del software implementado y no tiene en cuenta los principios ágiles.

Existe una tensión inherente entre ambas escuelas de pensamiento, y por eso es una razón fundamental por la que los investigadores han estado investigando cómo integrar ambos enfoques.

Los primeros intentos concretos de integrar los enfoques ágil y DCU se publicaron hace aproximadamente una década. Uno de los principales problemas es cómo facilitar la comunicación entre los profesionales involucrados.

La dependencia de la comunicación dentro de los equipos ágiles es una característica fundamental. En DACU, diseñadores y desarrolladores deben estar preparados para comunicarse y colaborar. En este artículo se defiende la idea de la comunicación mediada por artefactos en este escenario. Con el objetivo de identificar y comprender los artefactos utilizados para facilitar la comunicación entre diseñadores y desarrolladores en un enfoque DACU, se llevó a cabo un estudio etnográfico en una comunidad en línea de profesionales ágiles distribuida por todo el mundo.

Como conclusión los métodos ágiles y de DCU pretenden diseñar y producir software de calidad desde perspectivas diferentes. El enfoque de DACU intenta cerrar las brechas entre estas áreas aportando las técnicas, métodos y artefactos más eficaces de cada una de ellas. Sin embargo, no sólo diferentes aspectos afectan a esta integración, sino que también la comunicación entre diferentes perfiles profesionales, como diseñadores y desarrolladores, tienen una gran influencia en ella.

A través del estudio de etnografía los hallazgos señalaron dos temas principales:

(1) Los artefactos facilitan la comunicación y (2) Los artefactos apoyan la colaboración. La interpretación y delimitación de los temas muestran que el uso de artefactos facilita la comunicación de los equipos. Por ello, en esta tesina, se desarrolla la herramienta "*Detección de smells de UX en prototipos tempranos*" que debería servir para mejorar la comunicación entre el diseñador y el desarrollador.

2.5 Herramientas de diseño:

Para el desarrollo de esta herramienta, nos propusimos investigar acerca de las diferentes plataformas de diseño más utilizadas por los diseñadores, donde nos encontramos con Figma, Sketch y Adobe XD. Cada una posee sus ventajas y desventajas.

2.5.1 Figma

Lanzada en 2016, Figma [Figma] es una aplicación basada en un navegador que promueve un enfoque de colaboración en el diseño. La ventaja es que se puede acceder fácilmente en cualquier momento, independientemente del sistema operativo. Esta herramienta ha sido construida para soportar la colaboración en tiempo real, los miembros del equipo pueden comunicarse entre sí y gestionar sus propias tareas del proyecto, a su vez, el historial de versiones te permite retroceder los cambios, en caso que sea necesario. Figma tiene un plan gratuito que permite 3 proyectos, 2 editores y un historial de versiones de 30 días.

Entre otras características se destaca:

- La capacidad de crear estilos consistentes y aplicarlos a través de los proyectos.
- Copiar CSS directamente de los archivos de diseño.
- Una biblioteca de recursos de búsqueda.
- Permisos de usuario.
- Crear prototipos animados e interactivos.
- Función de autodiseño para diseños sensibles.
- Una biblioteca de plugins, y la posibilidad de crear tus propios plugins.

2.5.2 Sketch

Lanzada en 2010, es una herramienta de escritorio Sketch que solo está disponible para MacOS. Posee una biblioteca de varias extensiones que traen nuevas capacidades y mejoran el flujo de trabajo, además ofrece una prueba gratuita de 30 días.

Entre otras características se destaca:

- Edición de imágenes vectoriales.
- Diseños sensibles a través de Smart Layouts.
- Soporte para fuentes variables.
- Colaboración con colegas y clientes.

- Añade fácilmente datos basados en texto e imágenes a tu demo.
- Servicio Sketch Cloud para compartir sus creaciones.
- Bibliotecas para compartir recursos (símbolos, imágenes, texto, estilos) en los documentos.
- La posibilidad de crear y utilizar plantillas.
- Un gran número de plugins disponibles.

2.5.3 Adobe XD

Es una herramienta de edición gráfica que sirve para la creación de interfaces. Se trata de un programa dedicado de manera íntegra al diseño y prototipado de aplicaciones web.
Permite a los diseñadores trabajar en la experiencia de los usuarios al navegar, con un rango de error mínimo y un tiempo reducido, al reunir todas las herramientas que se incluyen en otros programas de Adobe y las concentra en una sola.

Se centra en el diseño de interfaces digitales para aplicaciones móviles, ya sean para PC o para Mac, o sitios web. Y a través de este programa es mucho más sencillo crear imágenes preliminares de la interfaz de un proyecto.

XD tiene un nivel gratuito sin límite de proyectos, sin embargo los archivos no pueden ser guardados localmente y deben ser guardados en la nube, con una capacidad de almacenamiento de 2 GB.

2.5.4 Elección de Figma

Procedimos a seleccionar Figma [Figma] debido a que permite la creación de plugins de la comunidad, por lo que a través de su API, podremos acceder a los diseños y a su información. Por lo que nuestra herramienta estaría basada en un plugin de Figma, el cuál se ejecutaría dentro del entorno de la misma herramienta de diseño.

Otro punto por el que elegimos Figma es porque Sketch solo se puede usar en Mac OS, Adobe XD solo en windows y Mac, pero Figma es un Browser-based, con lo que te permite usarlo en cualquier sistema operativo. Además Figma es gratis, en

cambio Sketch permite una prueba de una semana y después es pago y Adobe XD brinda también una prueba gratis pero que solo lo puede editar un usuario. Figma permite una versión gratis en la que pueden colaborar dos editores y la creación de máximo 3 proyectos.

- La colaboración es la funcionalidad central en torno a la cual se construye una gran parte de Figma, y se muestra. Estar basado en la web significa que las características de colaboración son muy rápidas y de fácil acceso, el modo multijugador es actualmente más avanzado que la beta de edición de XD, incluye un modo de observación, bibliotecas de equipo, y un proceso fluido para sincronizar y actualizar los cambios en los recursos compartidos.

- El hecho de que Figma tenga la capacidad de guardar efectos como estilos es muy útil para un sistema de diseño, así como la capacidad de elegir individualmente si guardar el trazo y los colores de relleno. La clasificación automática en grupos también es una característica útil.

- Figma puede generar código para iOS y Android, así como CSS. Ese código es mucho más fácil de acceder en línea en la aplicación Figma que el proceso de XD de generar un enlace y luego visitarlo en el navegador. El código de Figma se actualiza en tiempo real, y el CSS y el SVG pueden ser copiados directamente del lienzo. Estas características se suman para hacer de Figma un claro ganador en la generación de código.

- XD no tiene ningún tipo de API. Figma sí.

- Tanto Figma como XD tienen un amplio ecosistema de plugins con muchas herramientas útiles hechas por la comunidad.

2.5.5 Integración con Figma

La integración a Figma se hará en la etapa de desarrollo del diseño.
Nuestra herramienta se centrará en evitar ambigüedades por parte del equipo de desarrollo sobre los diseños propuestos por el diseñador, las especificaciones a los

elementos que brindará nuestro plugin facilitará la comunicación entre el diseñador y el desarrollador como así también le otorgará la responsabilidad al diseñador sobre las tareas que tengan que ver con los cambios de UX, lo que conlleva a disminuir las tareas al equipo de desarrollo.

A su vez, este plugin se ejecuta en la misma aplicación de Figma por lo que no hay que salirse del entorno de diseño para interactuar con la misma.

CAPÍTULO 3

Especificaciones de la herramienta

3.1 Requerimientos

Descubrir los requerimientos se centra en explorar el espacio del problema y definir lo que se va a desarrollar, lo cual incluye comprender a los usuarios y sus capacidades, cómo un nuevo producto podría ayudar a los usuarios finales en su vida diaria, las tareas, objetivos y contextos actuales de los usuarios. [SRP19-2].
Como requerimiento para la creación de este plugin se tomó lo documentado que se especificó en la sección “2.4 Trabajos Relacionados” junto con la entrevista detallada en la siguiente sección.

3.1.1 Entrevista a un diseñador

Para tener una visión realista sobre el diseño UX y su vínculo con el área de desarrollo, tuvimos la oportunidad de entrevistar a un profesional del área de diseño con 6 años de experiencia.
A partir de la investigación y trabajos relacionados, disponemos de algunos aspectos del proceso que aclarar, por lo tanto creamos un primer boceto de herramienta. A partir de allí, realizamos la entrevista para que nos explique cómo trabaja y para ello, utilizamos el boceto para tratar de encontrar si esa forma de encarar el problema podría ser de ayuda.

Introducción:
Consideramos que la calidad y el tiempo son muy importantes en el desarrollo de software, por lo que planteamos una herramienta para utilizar dentro del entorno de Figma donde se utilice como guía al diseñador para especificar su prototipo con el objetivo de poner fin a las ambigüedades que puedan surgir entre el diseño pensado por el diseñador y la implementación del mismo en el área de desarrollo. Este sistema mejorará la comunicación entre el diseñador y el área de desarrollo, ahorrando tiempo y ganando armonía.

La idea central de nuestra tesina es la de crear una herramienta dentro del entorno

de Figma que se encargue de recomendar acciones o requisitos a través de pre y post condiciones que se listan de acuerdo con el diseño realizado, es decir, dependiendo del componente del diseño donde se ejecute el plugin, este listará una serie de acciones que están relacionadas. Por ejemplo: Un botón, la acción será “click”. Esto generará comentarios para el desarrollador, con el fin de eliminar ambigüedades entre el diseñador y desarrollador, mejorando así la comunicación y evitando el surgimiento de malos olores de diseño, llegando a ahorrar tiempo de rediseño o de revisión del código y a su vez, estableciendo una relación de armonía entre el área de diseño y desarrollo.

- **¿Qué opinas de esta idea e investigación?**

  Por lo que vi, me gusto y siento que esta buena la estructura de investigación, los tópicos y herramientas que estudiaron, las competencias de Figma, los puntos de olor de UX más comunes en el desarrollo en conjunto entre el equipo de desarrolladores y diseñadores.

- **¿Qué situaciones parecidas recordás en tu trabajo?**

  Trabajando en la metodología ágil, me pasó que había que sacar una función rápido en un menú, entonces yo lo diseñé ya desplegado, con el color/estado ya estaba definido. Lo que pasó fue que el desarrollador utilizó ese color para el estado del componente sin estar activado. Fue una falta de comunicación porque yo no aclaré que ese color era sólo para cuando estaba activado. Yo me imaginé que cuando el mouse estaba por encima del componente iba a ponerse de ese color, y el desarrollador se imaginó cuando se hacía click.

- **Entorno de trabajo como UX:**
  - **¿Hace cuánto tiempo trabajas de diseñador de UX?**

    Hace aproximadamente 5 años que trabajo como diseñador de UX

  - **¿En tu trabajo, utilizan algún tipo de metodología de desarrollo?¿Cuál?**

Si, usamos SCRUM.

- **¿Cuáles son los roles que se incluyen y que hacen cada uno?**

  Yo trabajo en el equipo de marketing, soy el diseñador, me plantean la situación a resolver, cuál es el contenido que tiene que haber, y a partir de eso pienso cómo diseñar cada pantalla. Depende el tiempo que haya, hago primero Wireframes en alta (se piensa más la estética y la forma, se lo lleva a lo real, se saltean etapas). Hay un empleado que hace el empaquetado, otro hace la conexión entre la base de datos con otros servidores para recopilar información, como por ejemplo las imágenes de las portadas, hay otra persona que es el que dirige el equipo de desarrollo y está presente para brindar ayuda, a su vez, hay un scrum master y en mi caso tengo una jefa de marketing que es la que tiene la "última palabra".

- **¿Podrías describir cómo se inserta tu trabajo de diseñador dentro de esa metodología? ¿Te encuentras con algún problema interactuando de esa forma con el resto del equipo?**

  Al principio se me hizo muy difícil ya que va muy en contra con la metodología que estamos acostumbrado a trabajar los diseñadores UX, el problema es que la metodología brinda escasa ayuda, más allá de hablar de diseño, sino que para llegar a un producto que sea fácil de usar e intuitivo, así también como armónico con la marca, más allá que no todo es lo visual, al final del producto sí. Parte de la usabilidad es que la marca de la empresa se vea reflejada en la interfaz, entonces también da un poco de pertenencia a la app, y da a entender que si sigo dentro de Facebook o Instagram, no es una página, sino que son procesos que hacés mentalmente. Esta metodología (ágil) no deja madurar esos procesos que debería tener para que ya cuando salga al menos no haya que hacer muchas actualizaciones, ya que lo que hace es sacar el producto lo antes posible e ir iterando

constantemente. Pero lo que pasa, es que no se hace el arreglo del diseño, sino que el tiempo está entre agregar funcionalidades o mejorar la parte de diseño de UX y siempre se elige agregar funcionalidades nuevas. Es decir, se deja mucho de lado la parte de diseño. Además, para mi, se necesitan más personas para este tipo de metodologías, un equipo más grande de diseño en todo sentido, para evitar esta situación.

Hay momentos en los que sí se puede realizar un rediseño pero no son los deseables para la metodología ágil que se usa, no se brinda mucho tiempo para mejorar un diseño.

- **¿Piensas que se necesita más comunicación con el desarrollador? ¿Y de qué forma se te ocurre que podría ser esa comunicación?**

  Siento que con algunos tengo más comunicación que con otros, pero porque tambien del otro lado, siendo programador, hay muchos que no les importa, en vez de preguntarte a vos deciden ellos, por ejemplo la búsqueda de un ícono que no encuentran y ponen otro, luego lo veo y no es el mismo, son detalles que se les pasan por no preguntar sus dudas. Sí, hay problemas de comunicación pero no con todos, depende el desarrollador.

- **Con respecto a nuestra idea preliminar:**
  - **¿Qué herramienta de diseño utilizas para tu trabajo?**

    Uso todo el paquete de Adobe, menos la parte de prototipos que para ello utilizó Figma.

    Para diseñar uso Figma pero a veces necesito hacer cosas específicas y uso Illustrator. Uso más Figma sobre todo.

  - **¿Cómo se insertan en el proceso de diseño y desarrollo general?**

    Se inserta como algo que siento que nos une con los desarrolladores, es que Figma está hecho para desarrolladores y diseñadores, entonces los incluye de cierta forma, ya que es muy útil para su pipeline y su trabajo. Porque ahí tienen todos los datos de lo que yo

hago exactamente: color, espacios, tipografía, como se ve, etc.

- **Cuando se descubren errores de diseño o cosas para mejorar, ¿Cómo se registra el pedido de cambio?**
  En mi caso me comunican: "Ya está desarrollada la web, revisala y hacé una lista de los cambios" en ese momento, tengo que pasar la lista de los cambios a un grupo de team que tenemos. (entregás el diseño, el desarrollador lo hace y después te mandan a revisarlo y de ahí listo los cambios) y de ahí nuevamente reviso, porque a veces lo cambian mal.

- **¿Quién hace la corrección? ¿Se realiza una nueva versión del diseño? ¿Le comunicás al desarrollador qué tiene que modificar?**
  La respuesta en parte, se encuentra en la pregunta anterior.
  Sí, porque a veces por ahí el programador no pudo hacer un menú y lo hizo de otra forma, pero por ahí hay otras formas mejores de las que pensó, entonces yo le digo que ese menú no es, y si el desarrollador me dice que ese menú no pudo hacerlo entonces se analiza y en caso que quede como en la web, yo voy a figma a dejar el diseño igual que como está en la web. Figma no es visualmente fiel a como se ve luego en la web.
  Sí, le comunico al desarrollador con capturas, etc.

- **Muestra de la idea del plugin:**
  - **¿Usarías una herramienta que te ayude a especificar restricciones?**
    Yo creo que la usaría en proyectos de 0 que tiene que ser muy rápido o para empaquetados, cosas básicas. O si tengo un equipo con diseñadores juniors. A parte depende, por ejemplo yo tengo que crear cosas que no sean estandarizadas sino innovación, entonces en ese momento no me serviría.

  - **¿Qué le agregarías?**
    Por ejemplo: 'Te sugerimos que pongas "esto" en figma y decirle

sentite libre de editarlo'. No dejar sólo las pre y post condiciones, sino aclararle que puede editarlas y que deje esas pre y post condiciones en un comentario, sugerirle obviamente. Creo que lo básico lo cumple, después obvio siempre se puede agregar más cosas.

3.2 Ejecución de los escenarios de usos con el plugin

A continuación se detalla el procedimiento utilizando nuestra herramienta.

Luego de que el diseñador realice el diseño que le habían especificado, deberá respetar la nomenclatura que indicamos para aprovechar el máximo de nuestra herramienta. Una vez que haya corregido la nomenclatura de los componentes, procederá a ejecutar nuestro plugin con el fin de comentar la interacción que posee la interfaz.**Fig.1**

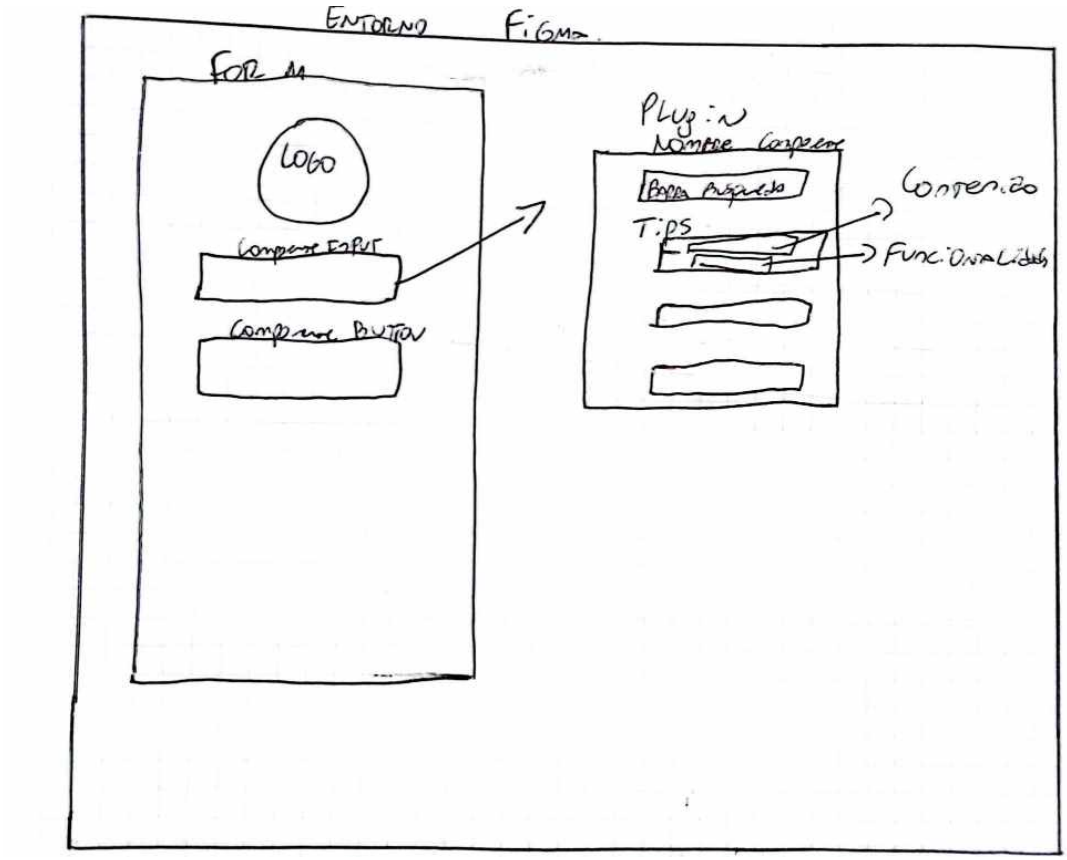


Fig.1 Primer diseño

Para el caso de ir habilitando campo por campo, se podrá utilizar la función "relacionar componentes" donde elige el segundo componente y lo relacionará con

el primero, este indicará que se habilitará una vez que el primero esté completo y así sucesivamente con cada componente.

Para especificar algún tip en especial, tendrá la lista de tips que ofrecemos por defecto, y a su vez estos podrán modificarlos libremente. Una vez modificado el tip a gusto, podrá utilizar la funcionalidad “Crear Comentario” el cual crea un comentario con la información del tip en el elemento donde se está trabajando. En caso que quiera hacerlo manualmente, podrá utilizar la funcionalidad “Copiar al portapapeles” donde copiará el tip completo y podrá pegarlo a gusto donde desee.

Para especificar restricciones, deberá ejecutar la funcionalidad “Agregar restricciones” donde se listará una serie de restricciones que le podrá agregar al diseño.**Fig.2**

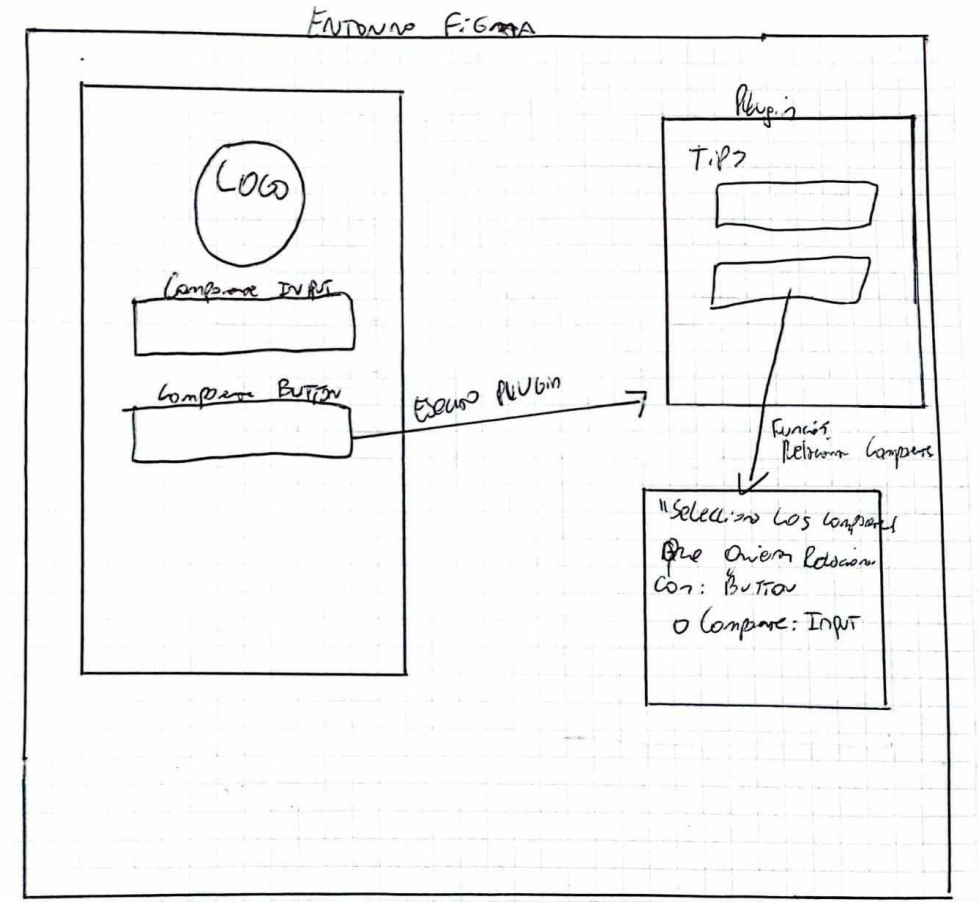


Fig.2 Diseño más detallado

De esta manera, obtendrá un diseño mejor detallado, el cual evitará la previa coordinación con el desarrollador (en caso que el mismo esté de acuerdo), ganando

así más tiempo para utilizarlo en otro tipo de tarea. Cabe destacar, que se evitarán malos olores de UX y se generará una comunicación mucho más clara y específica con el desarrollador, como así también, se deja de lado ambigüedades porque se deja constatado en el diseño lo que el diseñador especificó.

3.3 Escenario de uso

3.3.1 ESCENARIO 1: DISEÑO

En este escenario de uso, se necesita diseñar un formulario de registro para la aplicación. El diseñador decide realizar un mock up con el siguiente diseño preliminar:**Fig.3**

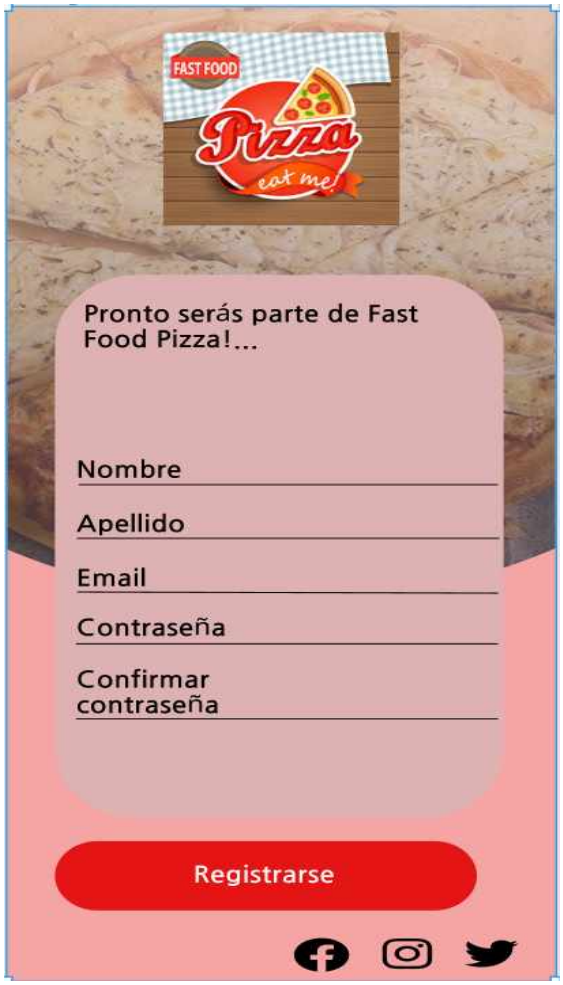


Fig.3 Formulario básico

El formulario está pensado para llenarse de manera secuencial. Por lo tanto, para mejorar la usabilidad, el diseñador decide que cada campo deberá habilitarse

cuando el anterior en la secuencia prevista esté completo. Una vez que todos los campos estén completos, se habilitará el botón de Registro. El diseñador también necesita especificar la longitud y el tipo de cada campo (restricciones). Sin embargo, las herramientas con las que trabaja no le permiten especificar esta información. Utiliza para ello una coordinación informal con el equipo de desarrollo. Esto puede generar actividades extras y refuerza la necesidad de buena comunicación en el equipo de trabajo.

Todas estas situaciones generan una serie de alternativas donde se da lugar a la libre interpretación y al surgimiento de malos olores.

Este es uno de los escenarios habituales de especificación de diseño de IU en el que la herramienta desarrollada en nuestra tesina, puede facilitar la tarea facilitando la especificación de las decisiones detalladas de diseño.

3.3.2 ESCENARIO 2: REDISEÑO

En este escenario de uso, el diseñador debe realizar un rediseño debido a una nueva funcionalidad, en este caso, se le pidió agregar un nuevo método de pago que sería el pago con tarjeta. Para ello, debe modificar el mockup que realizó en otro momento.

El mockup modificado sería el siguiente**Fig.4**:

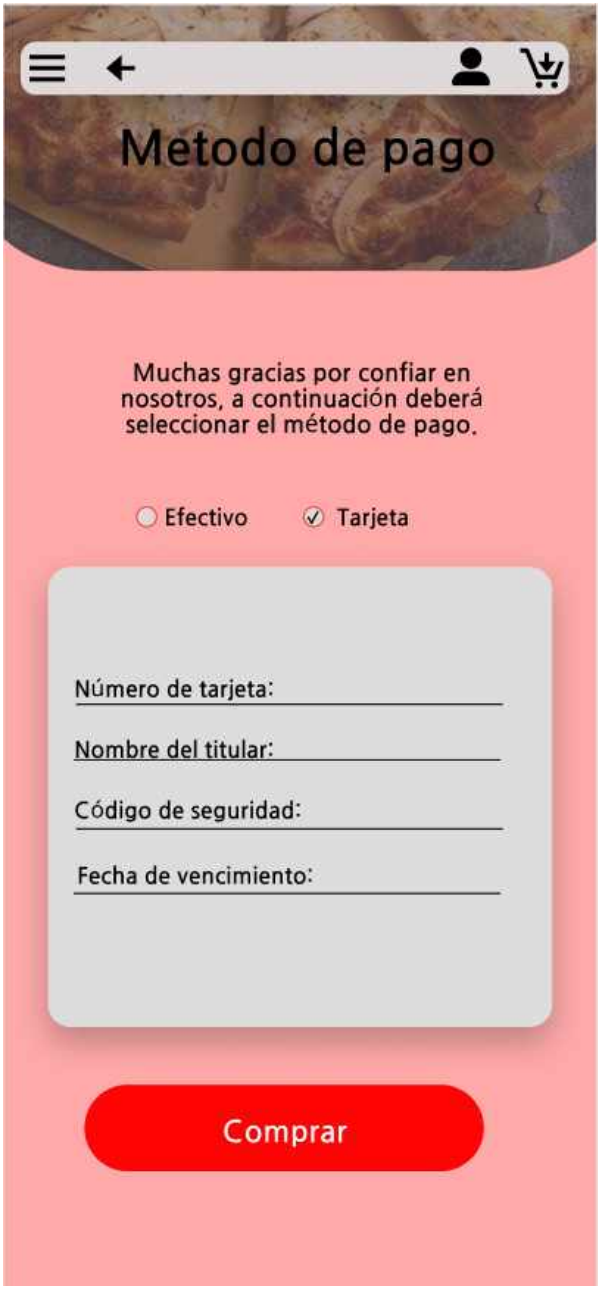


Fig.4 Formulario con Checkbox.

En este caso, el diseñador necesita poner un checkbox para poder seleccionar un método de pago. Seleccionando tarjeta, se deberá listar los campos que poseen cierta restricción:

Número de Tarjeta: Tendrá como longitud 16 caracteres de tipo entero.

Nombre del titular: Tendrá como longitud 20 caracteres de tipo string.

Código de seguridad: Tendrá como longitud 3 caracteres de tipo entero.

Fecha de vencimiento: Tendrá el formato de fecha, con longitud de 10 caracteres de tipo string.

A su vez, los campos se irán habilitando a medida que se vaya llenando el campo anterior y luego de completar los mismos, se habilita el botón “Comprar”.

De igual manera que planteamos en el escenario de uso 1, el diseñador es el encargado de especificar la interacción de la interfaz, pero no puede hacerlo con las herramientas existentes. Este es otro caso donde nuestra extensión a Figma cobraría mucha utilidad.

En una sesión normal de trabajo utilizando Figma, el diseñador tendrá a su disposición un complemento que le permite especificar estos requerimientos. Nuestro plugin estará disponible desde el momento que el diseñador comienza a trabajar en Figma, el flujo que soportará el mismo será similar al especificado en los escenarios de uso.

3.4 Funcionamiento básico del plugin

Nuestra herramienta consta de un funcionamiento sencillo, se deberá seleccionar 1 o más componentes y en base a eso, nosotros brindamos una serie de tips junto con diferentes funcionalidades. Cabe destacar que se debe respetar una nomenclatura, desarrollada en secciones posteriores, para sacar el máximo provecho de la misma, teniendo así más funcionalidades. En cambio, si no se respeta la nomenclatura de los componentes, se tendrá un plugin más “libre” pero limitado a ciertas funcionalidades.

A continuación se describen dos elementos esenciales de la herramienta, los cuáles se denominan tip y restricción.

3.4.1 Tips

Un tip es un dato de especificación que el diseñador agrega a su diseño, en las partes interactivas, dirigido al desarrollador, con el fin de evitar errores de interpretación por parte del área de desarrollo y así administrar mejor el tiempo de trabajo. Un tip está compuesto por:

- Nombre: Hace referencia al nombre del tip.

- Descripción: Se utiliza para describir con más detalle el tip en cuestión.

- Precondición: Es la condición que debe cumplirse antes de interactuar con el elemento implementado.

- Postcondición: Es la condición que debe cumplirse después de interactuar con el elemento implementado.

A continuación se listaran los tips que se encuentran cargados en la herramienta, los mismos surgen de la bibliografía analizada, de la entrevista realizada al diseñador, y de las pruebas y casos de uso que nosotros mismos realizamos.

Los inputs que dispone la herramienta por defecto poseen como precondición haber completado el campo anterior y como postcondición la habilitación de un componente de tipo input o un componente de tipo botón. Cabe destacar que además de los tips provistos por la herramienta, el diseñador cuenta con la opción de agregar un texto libre con la especificación que desea transmitir.

Button		
Tarea/Acción	**Precondición**	**Postcondición**
Form registro genérico	Se deberá haber llenado los campos anteriores, se habilita el botón registrarse	Se registra al usuario, si la carga es lenta, se deberá agregar una barra de progreso
Fallo envío de formulario	-	Si no se llenaron los campos requeridos correctamente, se marcarán los bordes de los campos con rojo que falten llenar o seleccionar. En caso de haberse llenado correctamente, se registra al usuario, si la carga

		es lenta, se deberá agregar una barra de progreso
Posar puntero del mouse	El mouse deberá estar posicionado sobre un elemento	Se mostrará un pequeño mensaje que describe el elemento
Botón iniciar sesión genérico.	Se deberá haber llenado los campos requeridos para la autenticación. Se habilita el botón iniciar sesión	-Se mostrará una barra de progreso, de acorde al tiempo de carga de la validación. -Luego se mostrará una alerta donde se inició sesión correctamente o las credenciales son incorrectas
Click con barra de progreso genérica	-	Por cada vez que se cambia de página o sección, deberá haber una barra de progreso
Feedback en la culminación de un evento	-	Se mostrará una barra de progreso hasta finalizar la acción, seguido de una alerta especificando la culminación del evento
Carrito. Click +: aumenta la cantidad del producto, Click -: Disminuye la cantidad del producto, en caso de llegar a 0 se elimina.	Haber agregado anteriormente un producto al carrito	Se habilita el botón de Finalizar compra si se tiene al menos 1 producto

Cambio de Estado/color	-	Cuando clickeamos ese botón, el estado/color cambia
Posición del mouse	-	Cuando ubicamos el mouse sobre el botón, el estado/color cambia

Checkbox		
Tarea/Acción	**Precondición**	**Postcondición**
Opción de pago genérico	-	Si está marcado tarjeta, desplegará los campos a llenar para finalizar la compra, una vez completados, se habilitará el botón comprar. Si se selecciona efectivo simplemente se habilitará el botón comprar
Uso genérico checkbox	Se mostrará múltiples opciones, donde se podrá elegir una o más	Se deberá haber seleccionado al menos un elemento de esas opciones
Uso genérico checkbox	Seleccionar al menos una opción del checkbox	Se cambiará a un color determinado el fondo de la opción elegida

RadioButton		
Tarea/Acción	**Precondición**	**Postcondición**
Uso genérico	Seleccionar la opción que	Se cambiará a un color

radio button	necesita del radio	determinado el fondo de la opción elegida

Select		
Tarea/Acción	**Precondición**	**Postcondición**
Uso genérico select	Desplegar las opciones del select	Al posar el cursor sobre las opciones del select se irá cambiando al color:

Link		
Tarea/Acción	**Precondición**	**Postcondición**
Link que despliega una nueva ventana	-	Se desplegará un pop-up
Link que te redirige a un sitio externo	-	Se redirigirá a una url externa
Link que te redirige a un sitio interno	-	Se redirigirá a una url interna
Link que antes de redirigir visualiza un “cargando”	-	Se desplegará una barra de progreso mientras se redirige a otro sitio.

3.4.2 Restricciones

Una restricción es similar a una regla de negocio que el diseñador agrega a su diseño, en las partes interactivas, dirigido al desarrollador, con el fin de evitar errores de interpretación por parte del área de desarrollo y así administrar mejor el tiempo de trabajo. A diferencia del tip, está más concentrado en aquellos componentes que

sean de tipo input, es decir, restringen la información que se carga. A continuación se listaran las restricciones que se encuentran en la herramienta.

Restricciones	
Nombre	**Descripción**
longitud_mail	El campo tendrá como máximo 50 caracteres de longitud
longitud_num_tarjeta	El campo tendrá 16 caracteres de longitud
longitud_input	El campo tendrá 30 caracteres de longitud
longitud_fecha	El campo tendrá 10 caracteres de longitud
longitud_cod_tarjeta	El campo tendrá 3 caracteres de longitud

Restricciones	
Nombre	**Descripción**
formato_mail	El campo deberá poseer un @
formato_num_tarjeta	El campo será de tipo numérico
formato_input	El campo será de tipo alfanumérico
formato_fecha	El campo será del tipo date
formato_cod_tarjeta	El campo será de tipo numérico
peso_archivo	El tamaño máximo del archivo será de 50 MB.

3.4.3 Relaciones

La funcionalidad “Relacionar componentes” permite relacionar los componentes seleccionados estableciendo alguna condición que se debe cumplir. Por ejemplo, si tenemos 3 componentes de tipo input y 1 de tipo botón, cuando en el componente “botón” presionamos “Relacionar” se listan los componentes restantes donde se podrán seleccionar los que queramos y estableceremos la condición que se debe cumplir, en este caso son: Lleno, vacío, habilitado, deshabilitado. Cabe destacar que para utilizar esta funcionalidad se debe respetar la nomenclatura especificada en la sección 5.1.2. Volviendo al ejemplo, una vez que seleccionamos la condición y los componentes se debe presionar “Relacionar” y se especificará en el tip de la siguiente manera:*RELACIÓN: Los siguientes componentes deben estar llenos: input-email, input-password, para que este componente esté habilitado.*

CAPÍTULO 4

Integración con Figma

4.1 Desarrollo del plugin en Figma

Figma posee una librería que brinda un objeto global y subobjetos [globalObjects], los cuáles disponen de propiedades y funciones que sirven para crear, visualizar y modificar el contenido de los diseños. A su vez, cuenta con una API REST [ApiFigma], la cual posee varios endpoints que permite realizar más acciones que utilizando la librería. Nuestra herramienta se vincula con Figma utilizando ambas variantes, dependiendo la funcionalidad que desarrollamos.

4.1.1 Tecnologías utilizadas

Figma [DocumentacionFigma] permite escribir con Javascript los plugins y utiliza HTML para la creación de la IU. También recomienda utilizar Typescript para la escritura de la herramienta porque permite crear plugins más robustos, además sugiere la utilización de web pack para agrupar grandes proyectos de varios archivos e importar bibliotecas, y usar React para crear interfaces de usuario más complejas. Por lo mencionado anteriormente utilizamos un template [template] que incluía React y web pack, a su vez, esto permite optimizar la herramienta y así, obtener el máximo rendimiento.

Para la recuperación de las precondiciones y postcondiciones utilizamos un archivo json.

4.1.2 Funcionamiento relacionado a Figma

La herramienta “Mockup Spec” consiste, básicamente, en brindar tips con precondiciones y postcondiciones, para aplicar a un componente, ya sea un input, un botón, un checkbox, un link, un radio button, etc, especificando su comportamiento antes y después de una acción. El objetivo principal de la misma es evitar rediseños por falta de especificación de los elementos que poseen interacción con el área de desarrollo.

4.1.3 Token personal

Para poder usar la funcionalidad crear comentario, se debe ingresar un token, el cuál permite publicar comentarios desde la herramienta al diseño. A continuación, se detalla la serie de pasos que se debe realizar para obtener el token:

1. Para generar un token de acceso personal debe iniciar sesión en su cuenta de Figma y dirigirse al home.
2. Luego debe dirigirse a la configuración de la cuenta desde el menú superior derecho dentro de Figma y presionar settings. **Fig.5**

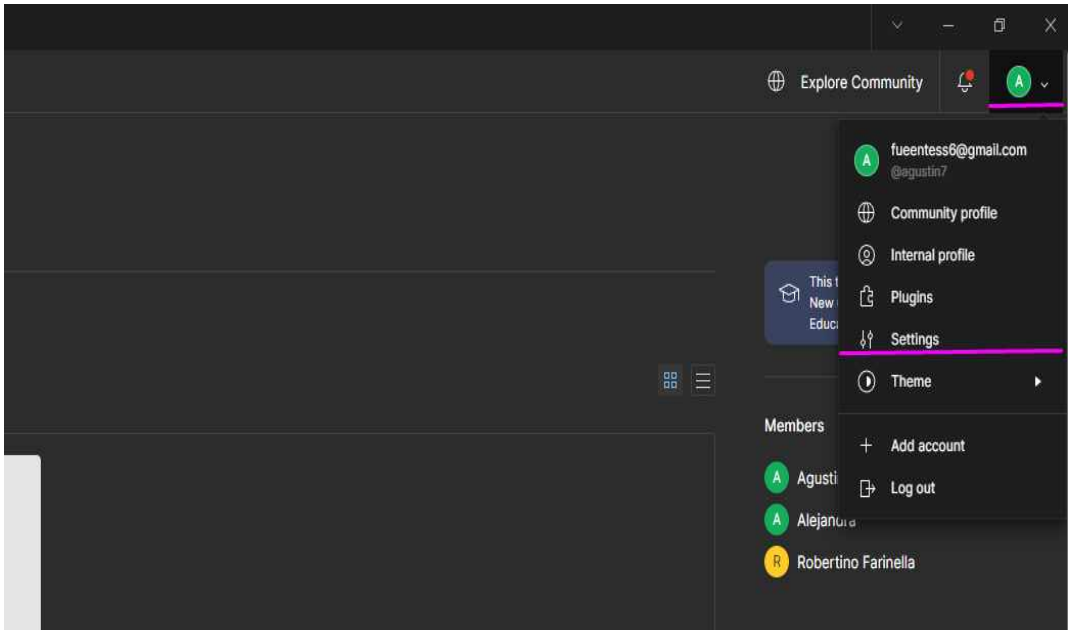


Fig.5. Opciones en Figma

3. Debe buscar la sección de tokens, ingresar un nombre para el token en el campo "*Create a new personal access token*" y presionar enter. **Fig.6**

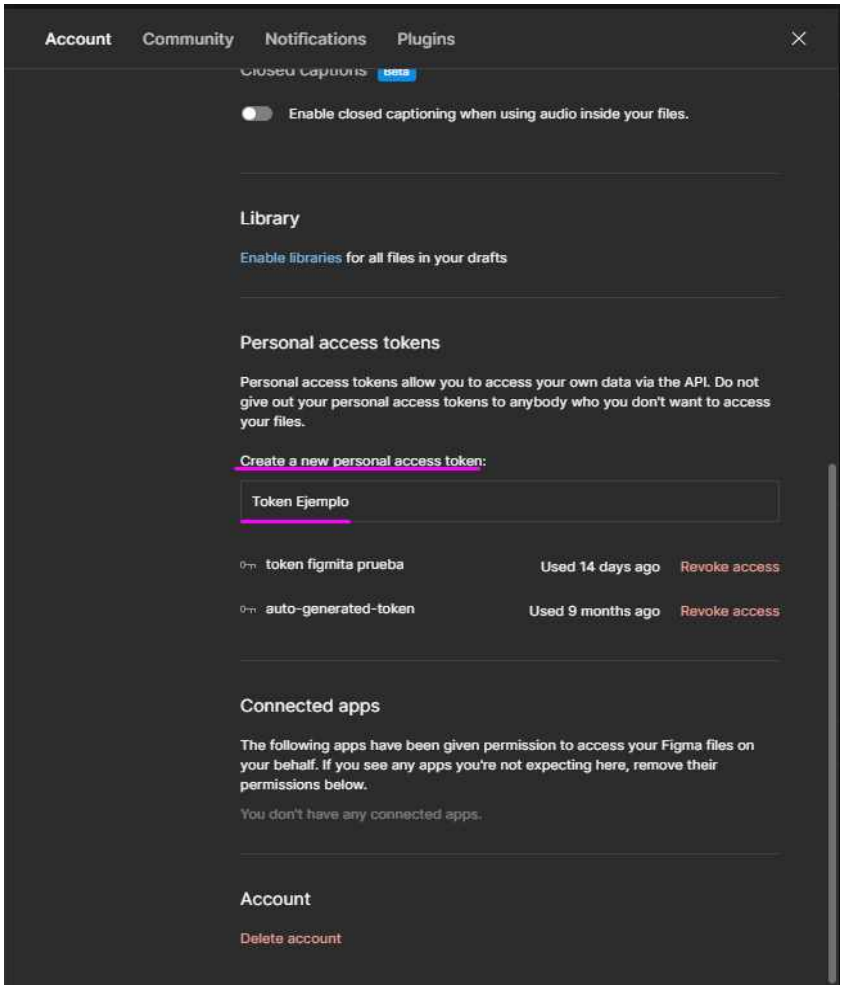


Fig.6 Creación de token personal

Luego aparecerá el token de acceso, se debe copiar y posteriormente pegar en el home de la herramienta. **Fig.7 y Fig.8.**

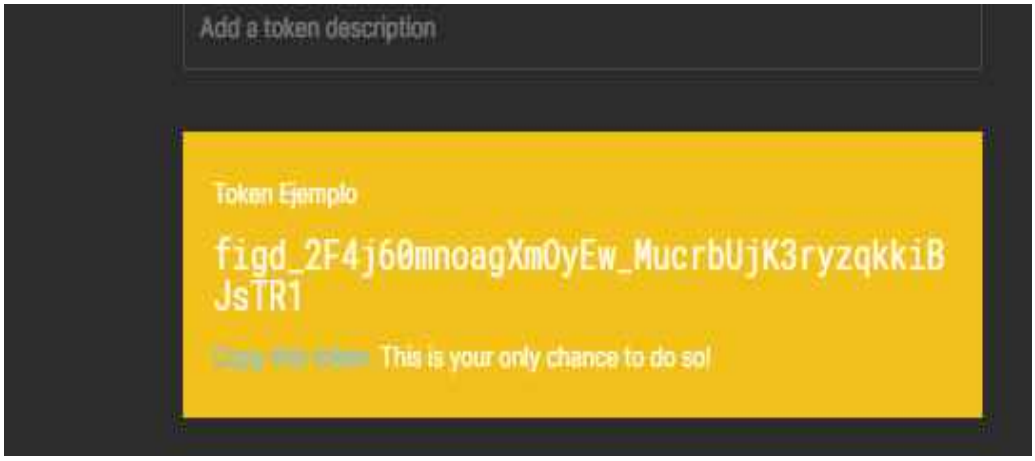


Fig.7 Token

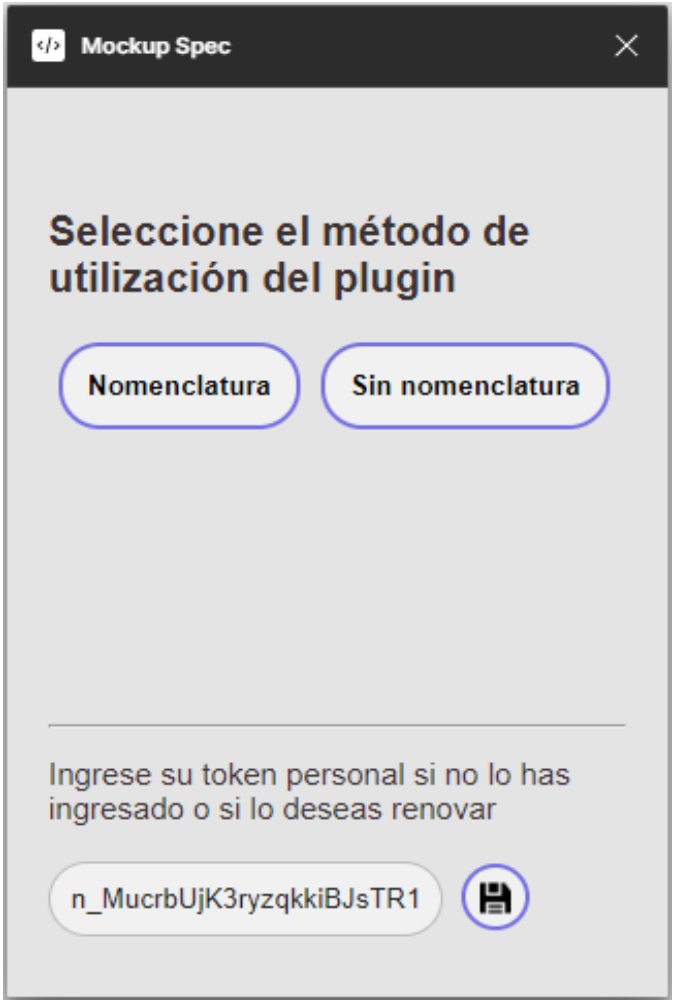


Fig.8 Ingreso de token

En caso de no ingresar el token, la funcionalidad crear comentario no se podrá llevar a cabo teniendo que realizar el comentario manualmente, para ello se dispone de la función “copiar al portapapeles” la cuál copia el contenido del tip para que luego el diseñador cree un comentario y pegue desde el portapapeles.

4.2 Endpoints y librería de Figma

Para el desarrollo del plugin utilizamos una librería de Figma para poder obtener el árbol de componentes que tiene el diseño, al igual que para almacenar el token de acceso en el client storage.

El método*figma.currentPage.selection*devuelve un árbol de todos los componentes que se seleccionaron a la hora de ejecutar la herramienta, donde cada componente tiene distintos atributos tales como el id, el nombre, la posición x e y, entre otros.

El único endpoint de la api rest de Figma que utilizamos es el de postear un comentario. Para crear el plugin y la interfaz de usuario, utilizamos la librería que provee Figma anteriormente mencionada.

En base a esto, tuvimos ciertas limitaciones con la librería y los endpoint que provee Figma. Una de las más destacadas, es el no saber el tipo de componente que el diseñador crea, frente a ello, no podríamos realizar alguna clasificación para determinar si un componente es un texto, un botón, un link, un checkbox, entre otros. Para solucionar esta limitación, se nos ocurrió crear una nomenclatura, utilizando el nombre del componente, donde se podrá filtrar por el nombre y en base a ello, obtener los distintos tips relacionados.

Otra limitación muy importante que nos encontramos y no pudimos solucionar, es el posteo del comentario mediante la funcionalidad “Crear comentario”. Cuando el diseñador presiona el botón dentro del plugin para ejecutar la funcionalidad “Crear comentario”, internamente, se recupera mediante la librería de Figma, las coordenadas (x,y) del componente donde se quiere crear el comentario, y con esas coordenadas y otros datos más como por ejemplo el tip, se utiliza el endpoint *https://api.figma.com/v1/files/' + fileKey + '/comments'* [RESTAPI], para subir el comentario al proyecto. La limitación surge en el momento de la creación del comentario, ya que a veces lo pega en las coordenadas correctas y otras veces no.

Sin embargo, el comentario queda en una ubicación cercana a las coordenadas donde debería haberse creado.

CAPÍTULO 5

Manual de usuario

En las siguientes secciones se explicará todo lo necesario para poder utilizar el plugin, asemejándose a un manual de usuario. Cabe destacar que un manual de usuario es un documento de comunicación técnica destinado a dar asistencia a las personas que utilizan un sistema en particular. El mismo estará compuesto por la especificación de convenciones a tener en cuenta a la hora de nombrar los componentes que el diseñador va a utilizar y las funcionalidades que brindará la herramienta para aumentar la especificación del diseño y la comunicación entre el diseñador y el desarrollador.

5.1 Importación del plugin

Para poder utilizar esta herramienta en el entorno de Figma se deberá seguir los siguientes pasos:

1. Dirigirse a la siguiente url: https://github.com/agu17/plugin-figma

2. Presionar en el botón “Code” y luego “Download ZIP”.**Fig.9**

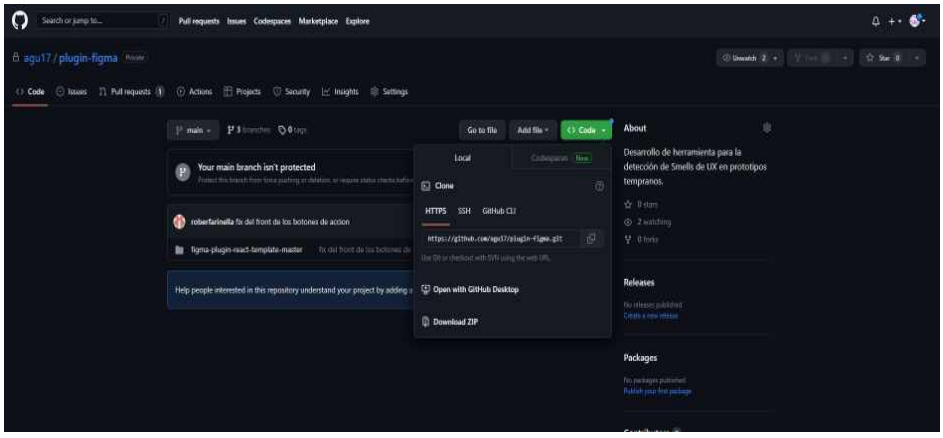


Fig.9 Repositorio

3. Una vez descargado el ZIP, descomprimir el archivo y luego se deben dirigir a Figma, ir a Menú→Plugins→Development→Import plugin from manifest, seleccionar la carpeta descomprimida y buscar el archivo manifest.json. **Fig.10**

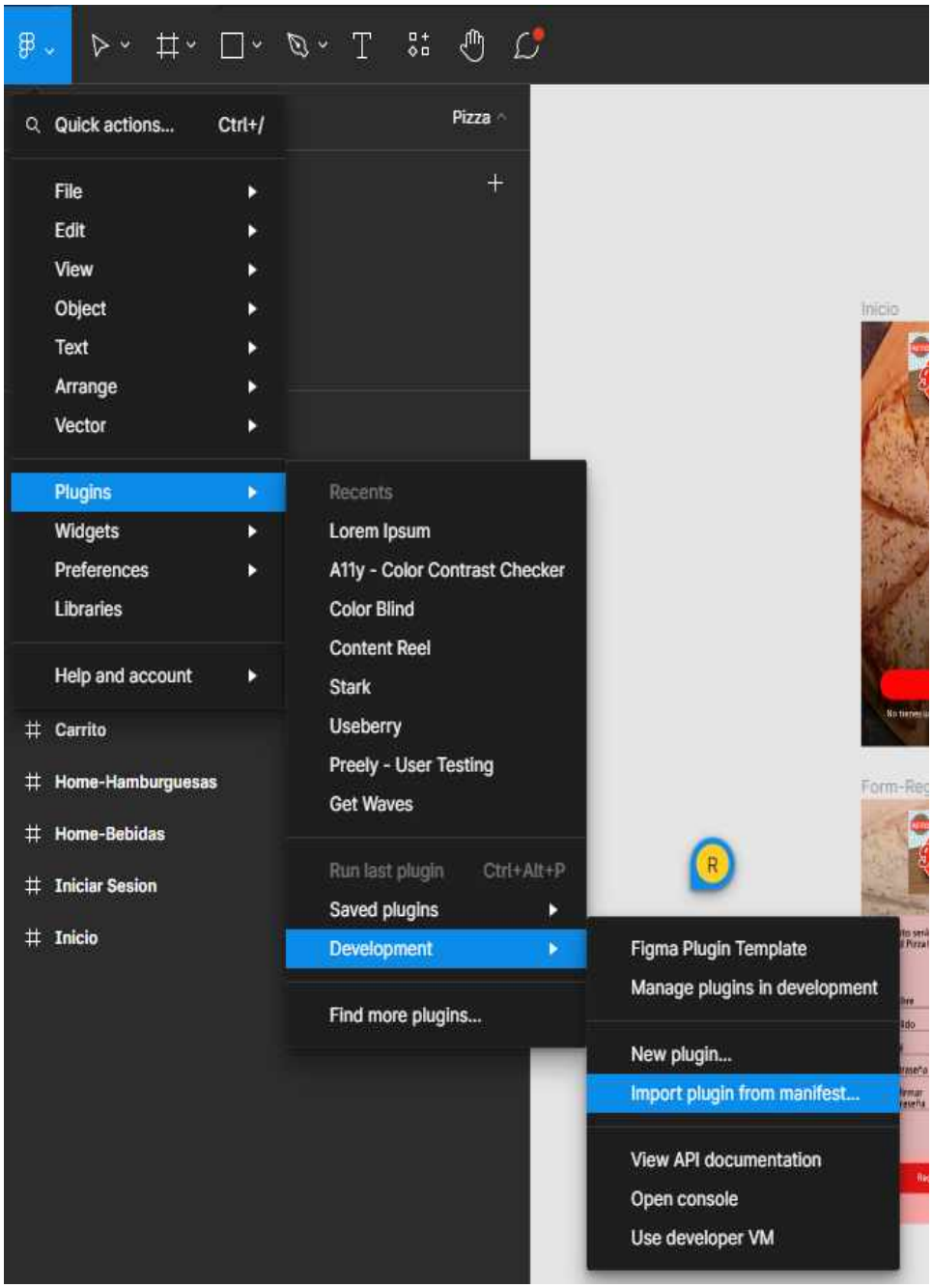


Fig.10 Importación del plugin

4. Una vez que lo encontramos, presionaremos “Run” y se ejecutará.

Estos pasos deben hacerse una única vez, luego para ejecutar el plugin de una manera más sencilla se debe hacer: seleccionar los componentes→click derecho →plugins→development→ Mockup Spec **Fig.11**

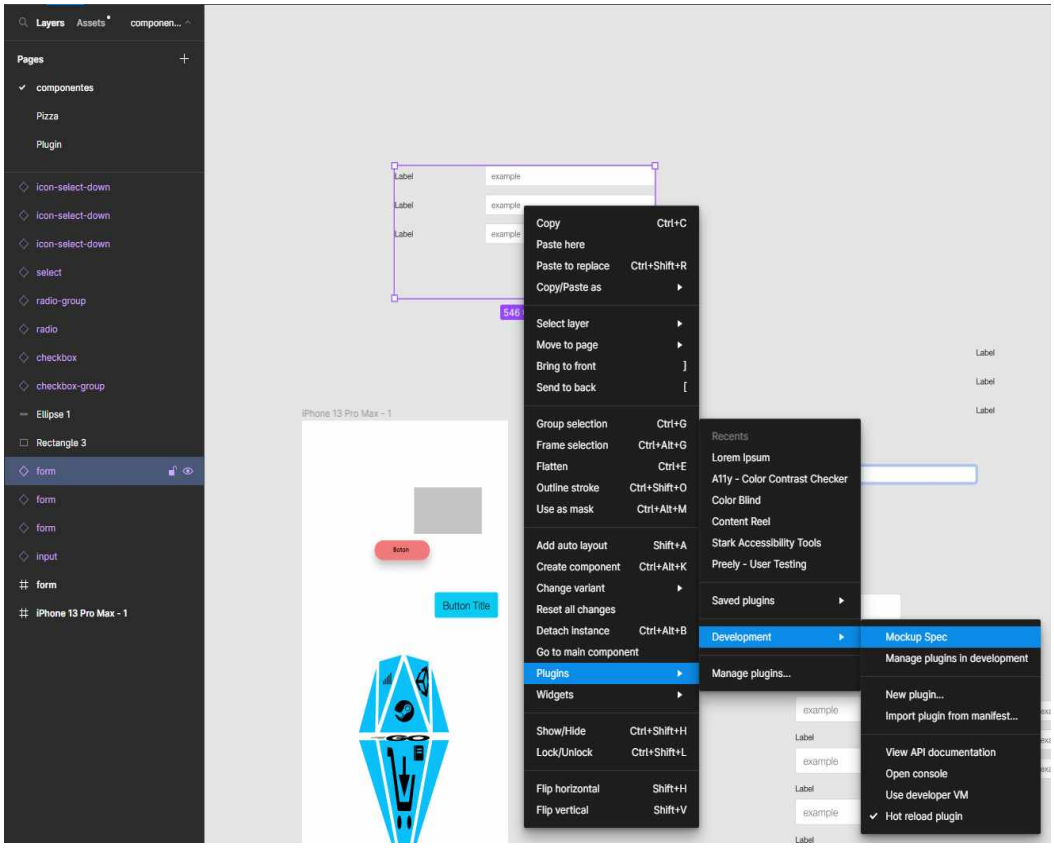


Fig.11 Ejecución del plugin

5.2 Especificación de convenciones

Al momento de ejecutar la herramienta en un diseño, establecimos una serie de reglas que debe seguir el diseñador para utilizar de manera óptima el plugin. En caso de no seguir estas reglas, la herramienta funcionará pero no se aprovechará su potencial.

Estás reglas brindarán una cierta facilidad a la hora de elegir qué tip corresponde al componente de figma, ya que esto se logra realizando un filtrado por el nombre del componente de figma. A continuación detallaremos los diversos tipos junto con sus tips.

5.2.1 Input

Aquellos componentes que su nombre contenga la palabra “input” en su nombre en el diseño, será contemplado como un input, por lo que se listará de manera automática aquellos tips donde se deba ingresar contenido. **Fig.12.**

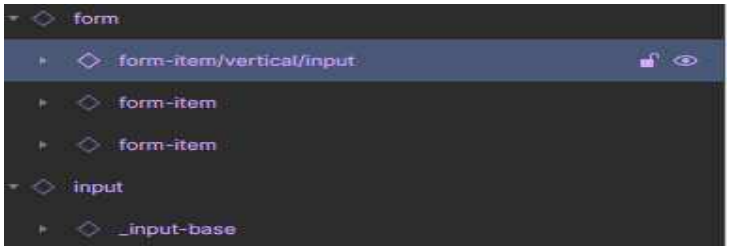


Fig.12. Nomenclatura Input

5.2.2 Button

Aquellos componentes que su nombre contenga la palabra “button” en su nombre en el diseño, será contemplado como un button, por lo que se listará de manera automática aquellos tips donde se deba presionar un botón.**Fig.13.**

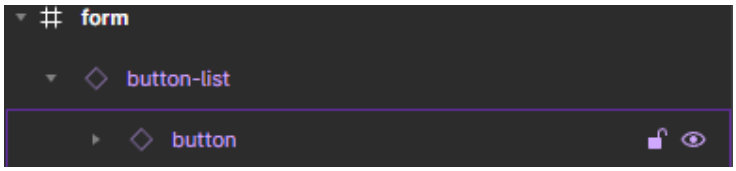


Fig.13. Nomenclatura button

5.2.3 CheckBox

Aquellos componentes que su nombre contenga la palabra “button”, será contemplado como un checkbox, por lo que se listará de manera automática aquellos tips donde se deba seleccionar al menos un elemento de un checkbox. **Fig.14**

Fig.14. Nomenclatura Checkbox

5.2.4 RadioButton

Aquellos componentes que su nombre contenga la palabra “Radiobutton”será contemplado como un Radiobutton, por lo que se listará de manera automática aquellos tips donde se deba seleccionar un único elemento.**Fig.15**

Fig.15. Nomenclatura RadioButton

5.2.5 Select

Aquellos componentes que su nombre contenga la palabra “Select”, será contemplado como un Select, por lo que se listará de manera automática aquellos tips donde se deba seleccionar un único elemento de una lista desplegable.**Fig.16.**

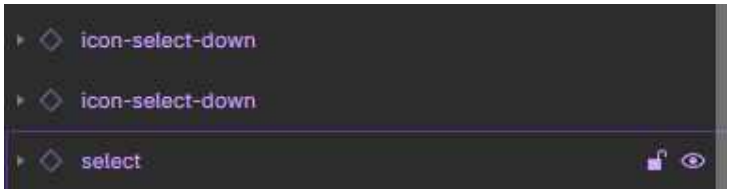


Fig.16 Nomenclatura Select

5.2.6 Link

Aquellos componentes que su nombre contenga la palabra “Link”, será contemplado como un link, por lo que se listará de manera automática los tips relacionados a un enlace.**Fig.17**

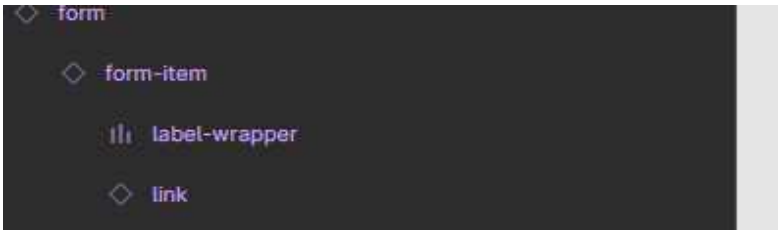


Fig.17 Nomenclatura Link

5.3 Funcionalidades

A continuación se detallarán las funcionalidades de la herramienta y lo necesario para poder llevarlas a cabo. Para ello, utilizaremos un diseño real, en este caso elegimos una app similar a la de Rappi.

Se procede a elegir el mockup orientado al registro de un usuario **Fig.18.**

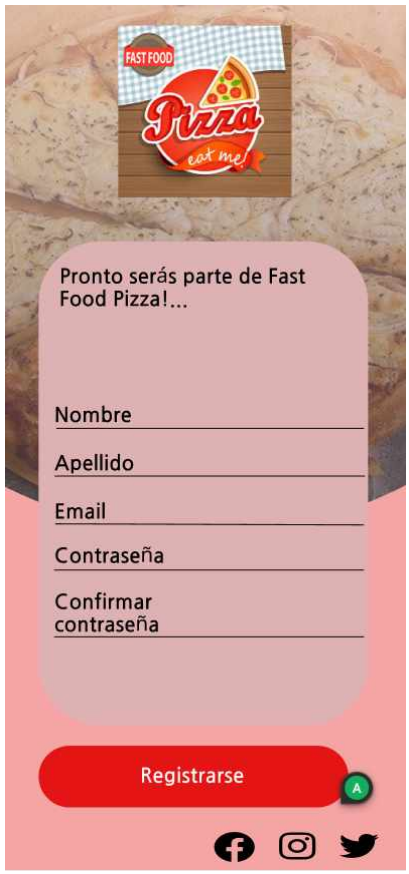


Fig.18. Formulario de ejemplo

Se procede a seleccionar el grupo de componentes donde se va a ejecutar el plugin.**Fig.19**

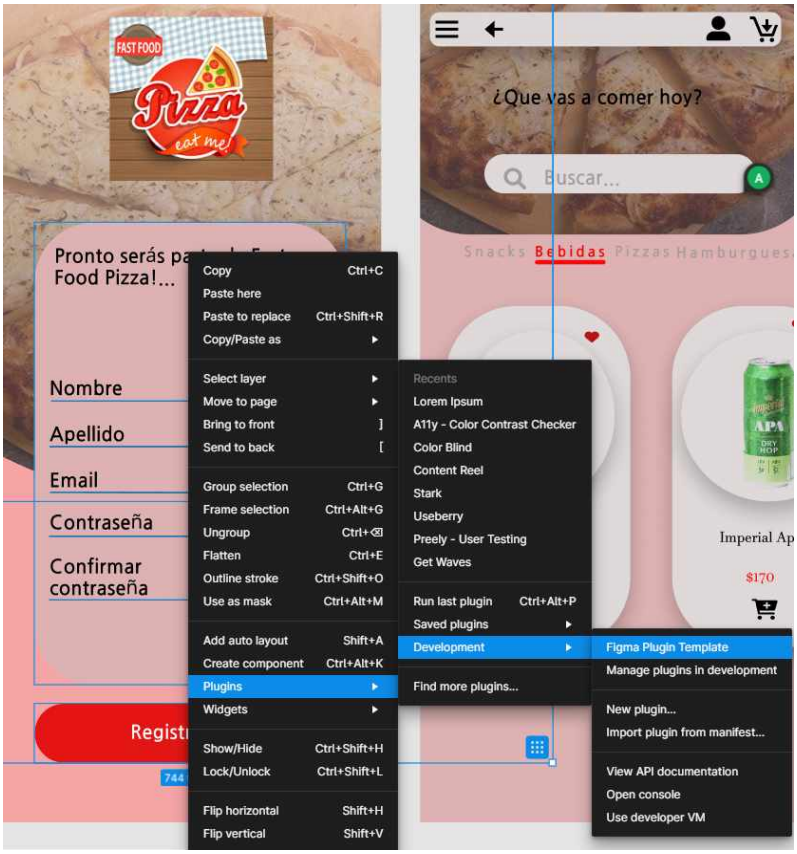


Fig.19 Selección de componentes

Una vez seleccionado “Detección de smells de UX en prototipos tempranos”, se desplegará el plugin.**Fig.20**

5.3.1 Selección de método de utilización:

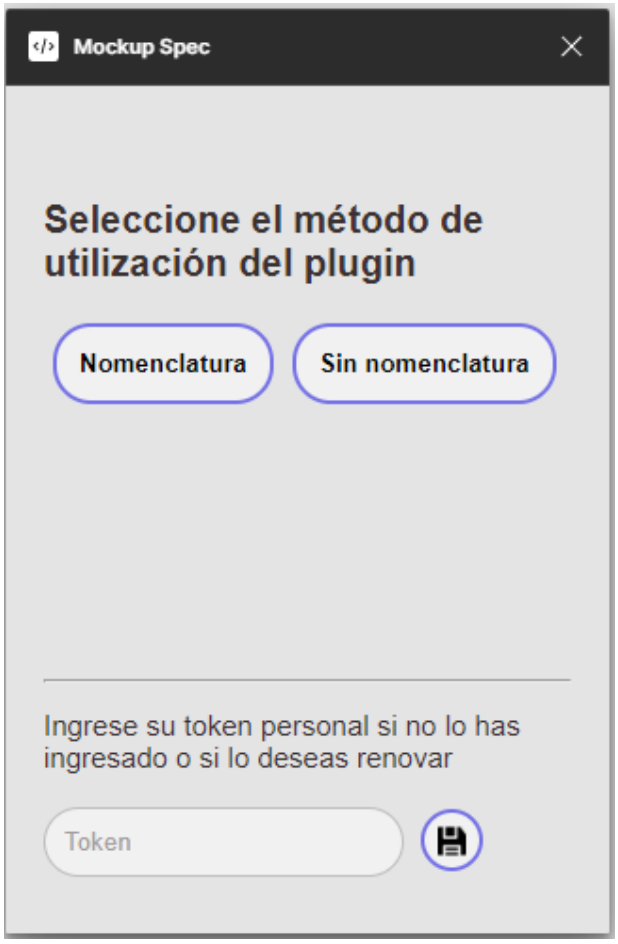


Fig.20 Home plugin

Se solicita al usuario que seleccione el método de utilización de la herramienta. Si se selecciona “nomenclatura”, se listaran aquellos componentes cuyo nombre coincida con algún nombre de la acción listada en el archivo Json de precondiciones y postcondiciones. En caso de seleccionar “Sin nomenclatura”, se listarán todos los tipos de acciones diferentes, es decir, se podrá trabajar libremente seleccionando la acción que necesitemos sin tener en cuenta el nombre del componente. La diferencia consiste, en que al leer el nombre del componente, por ejemplo, input, listará ese componente y consecuentemente, las variantes a ese tipo de acción.

Debido a los nombres de los componentes (fig.19) de nuestro ejemplo (fig.18), seleccionaremos el método nomenclatura.**Fig.21**

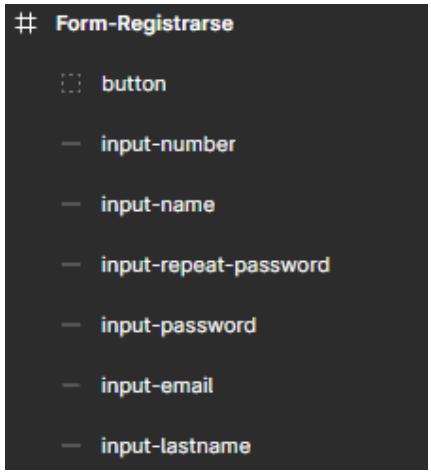


Fig.21 Nombre de los componentes

Si ejecutamos la herramienta sobre un formulario donde dispone de varios componentes se verá de la siguiente manera. **Fig.22**:

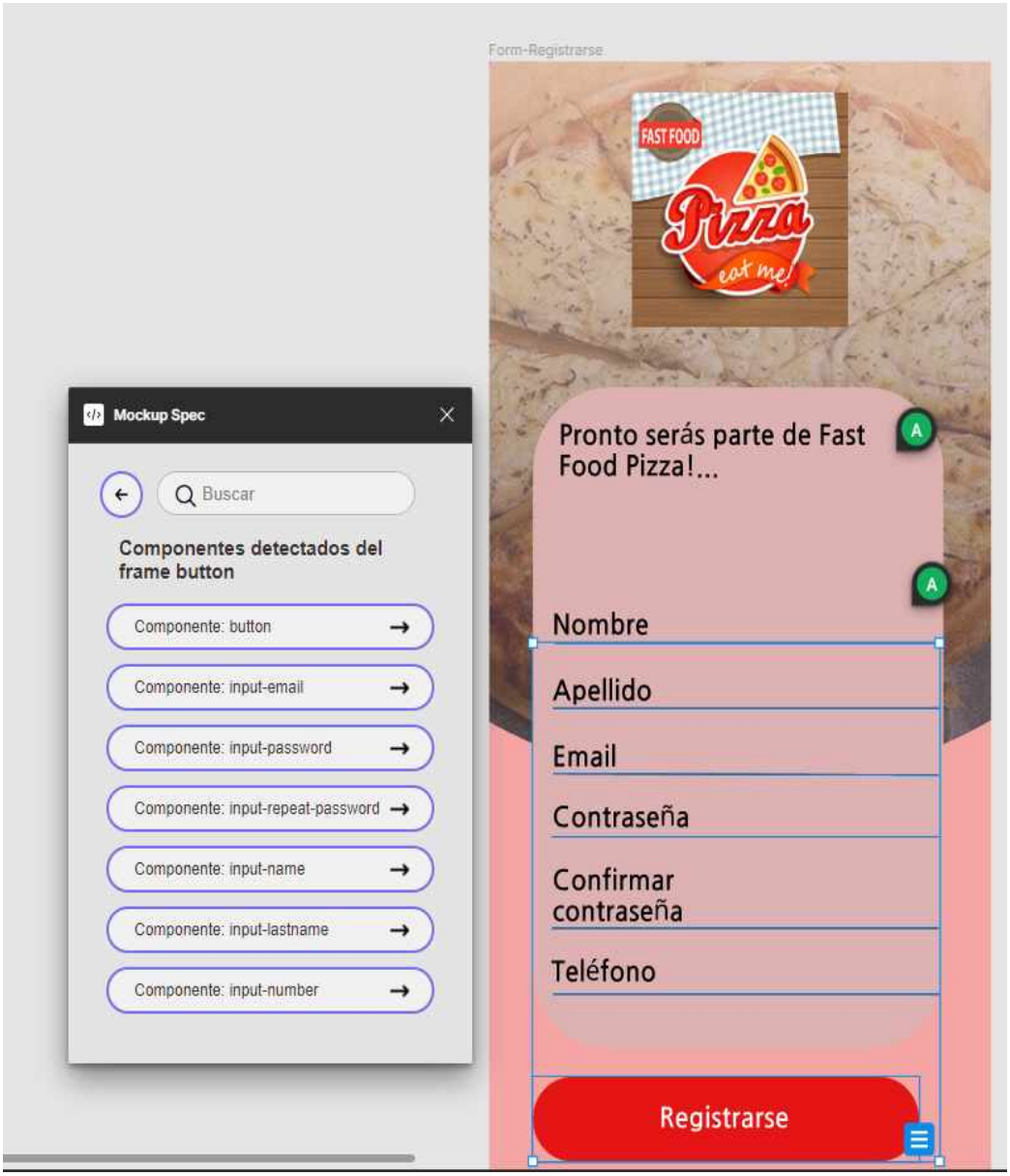


Fig.22 Ejecución del plugin utilizando nuestra nomenclatura

En esta imagen podemos ver la lista de componentes que posee el formulario que tomamos como ejemplo, donde se refleja los nombres de los componentes que componen el formulario.

En cambio, si seleccionamos el método “Sin Nomenclatura” se verá de esta forma. **Fig.23**:

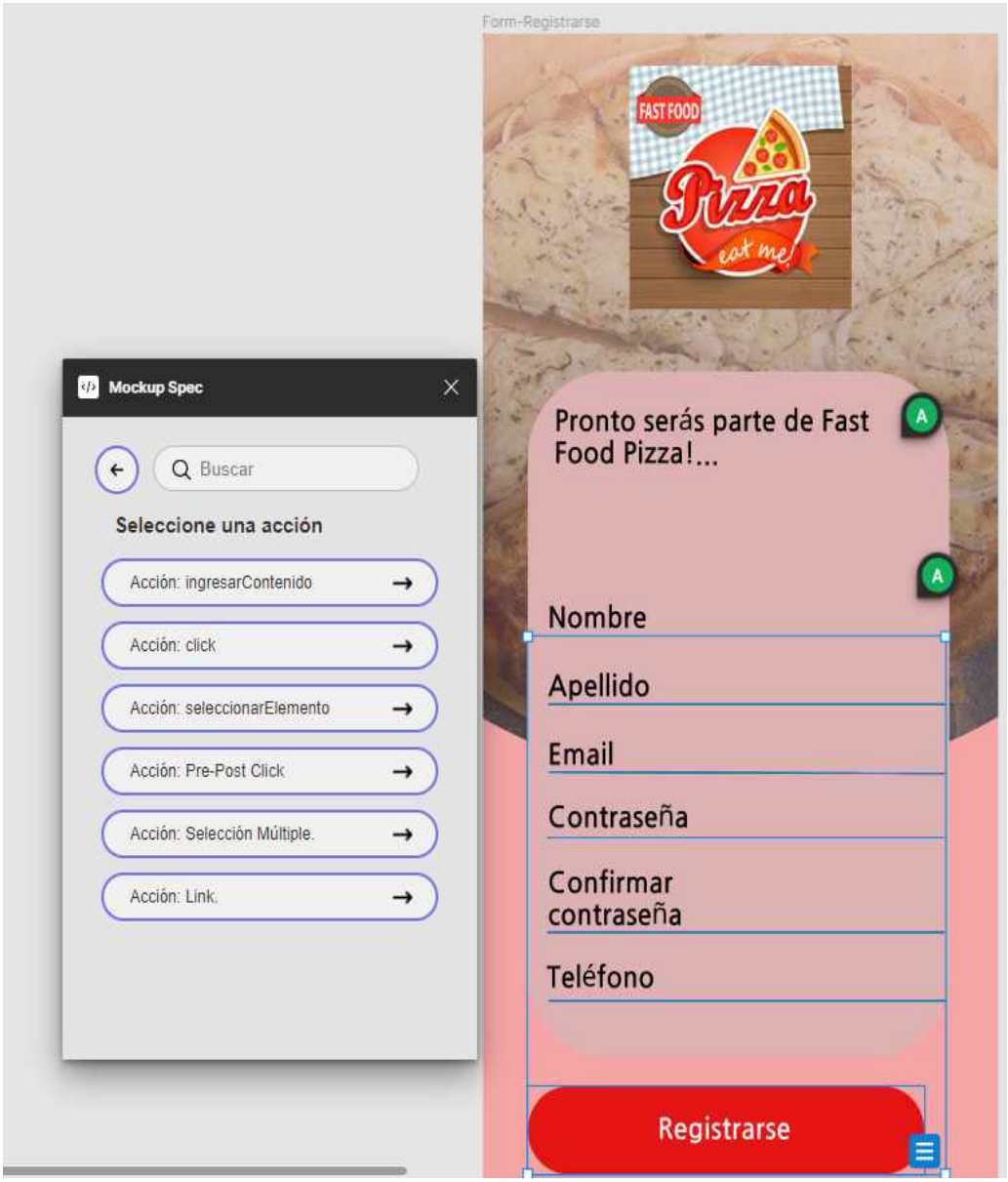


Fig.23 Utilización del plugin sin utilizar nuestra nomenclatura

5.3.1.1 *Camino 1, método respetando la nomenclatura*:

Luego de seleccionar nuestro componente, en nuestro ejemplo tomaremos el botón, se listan los tips relacionados a ese tipo de componente, donde intervienen varias funcionalidades. **Fig.24**

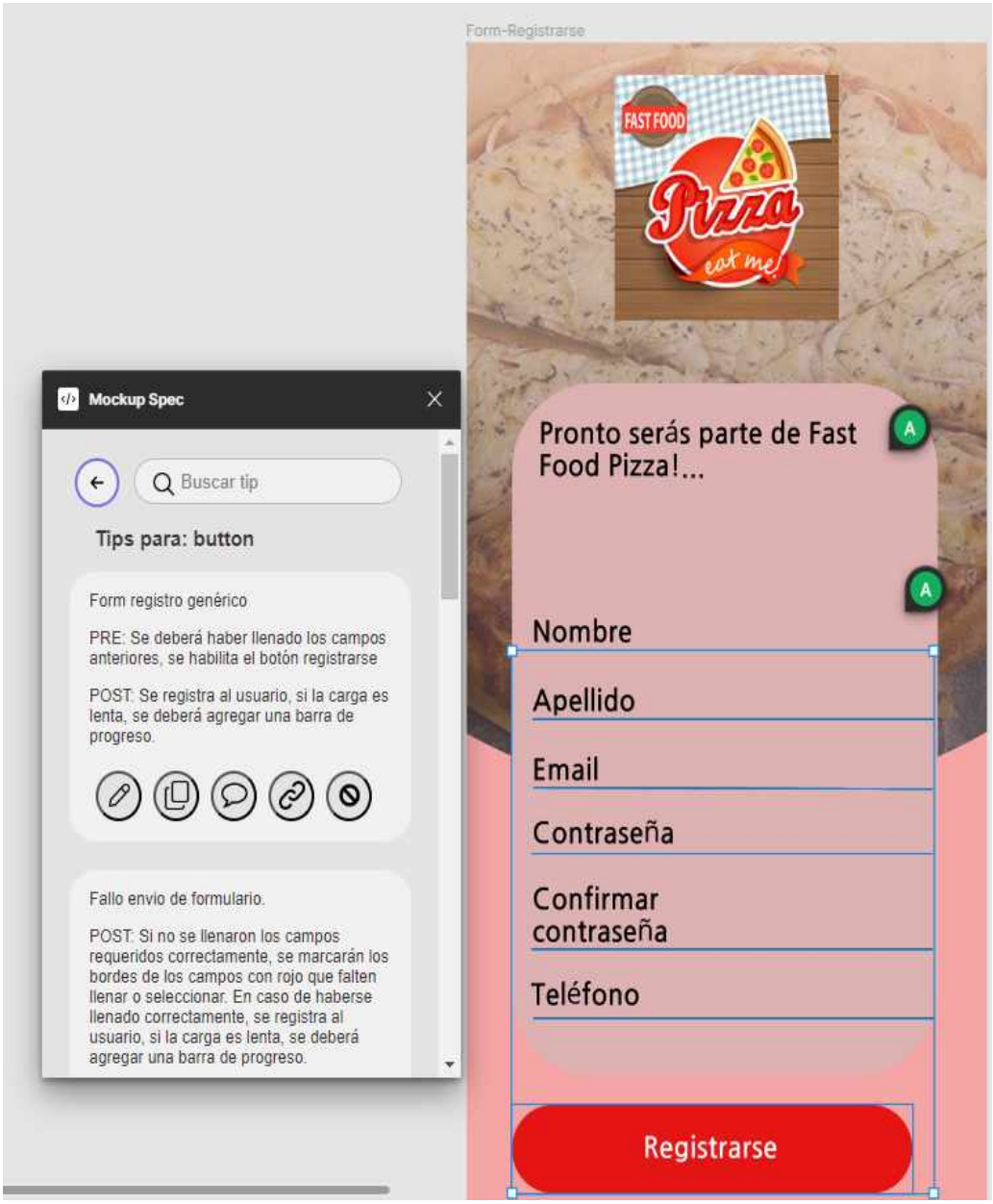


Fig.24 Tips boton

5.3.1.1.1 Editar precondición y postcondición: El primer botón consiste en poder editar el texto sugerido, pudiendo así cambiar el contenido a gusto del diseñador. El icono en color rojo refleja que estamos editando ese tip, para dejar de editar simplemente debemos presionar nuevamente este botón. **Fig.25**

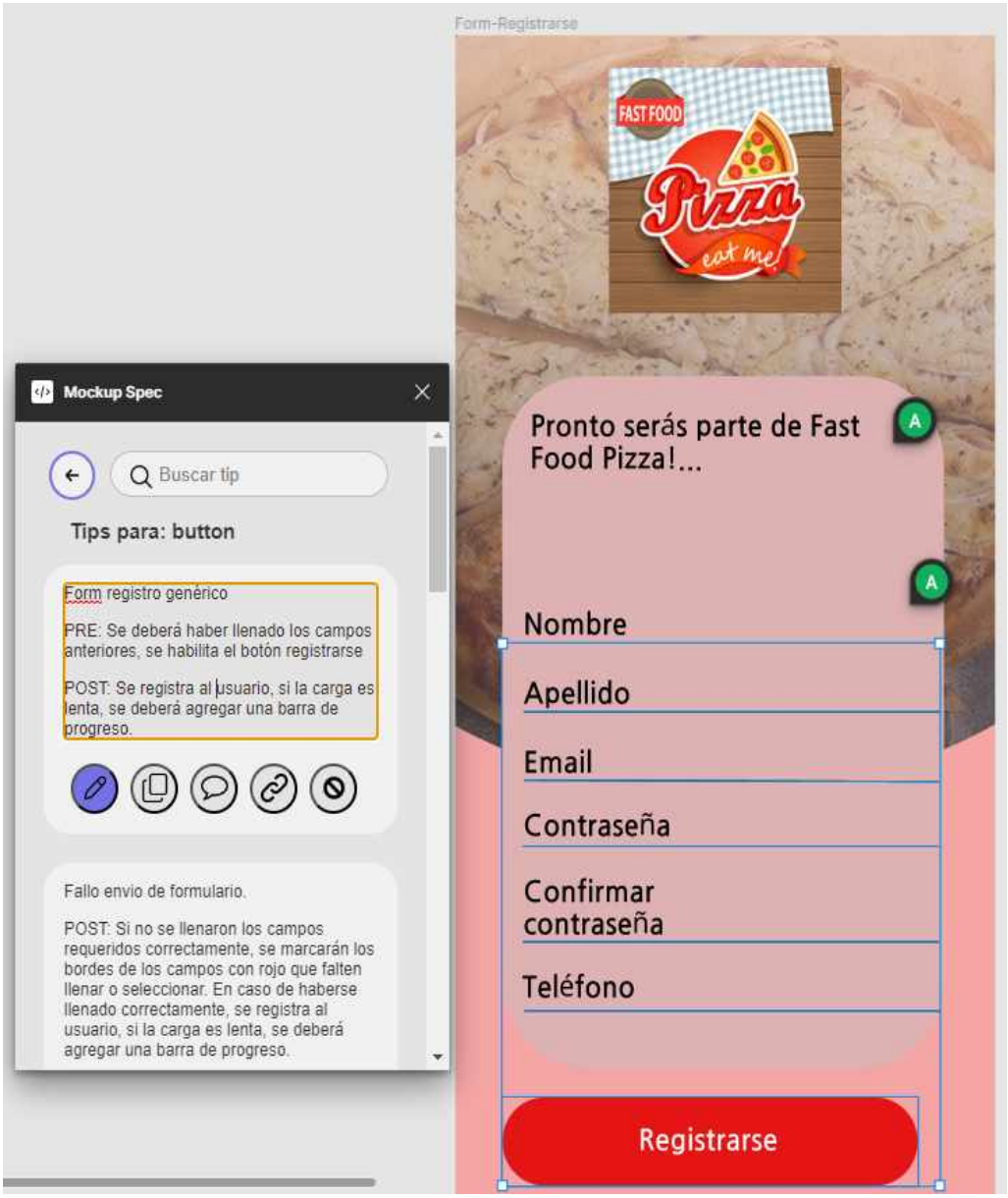


Fig.25 Editar tip

5.3.1.1.2 Copiar al portapapeles

El segundo botón consiste en copiar el contenido del tip al portapapeles, permitiendo así poder crear un comentario manualmente y pegar la información en el mismo. Es una variante al crear comentario. **Fig.26**

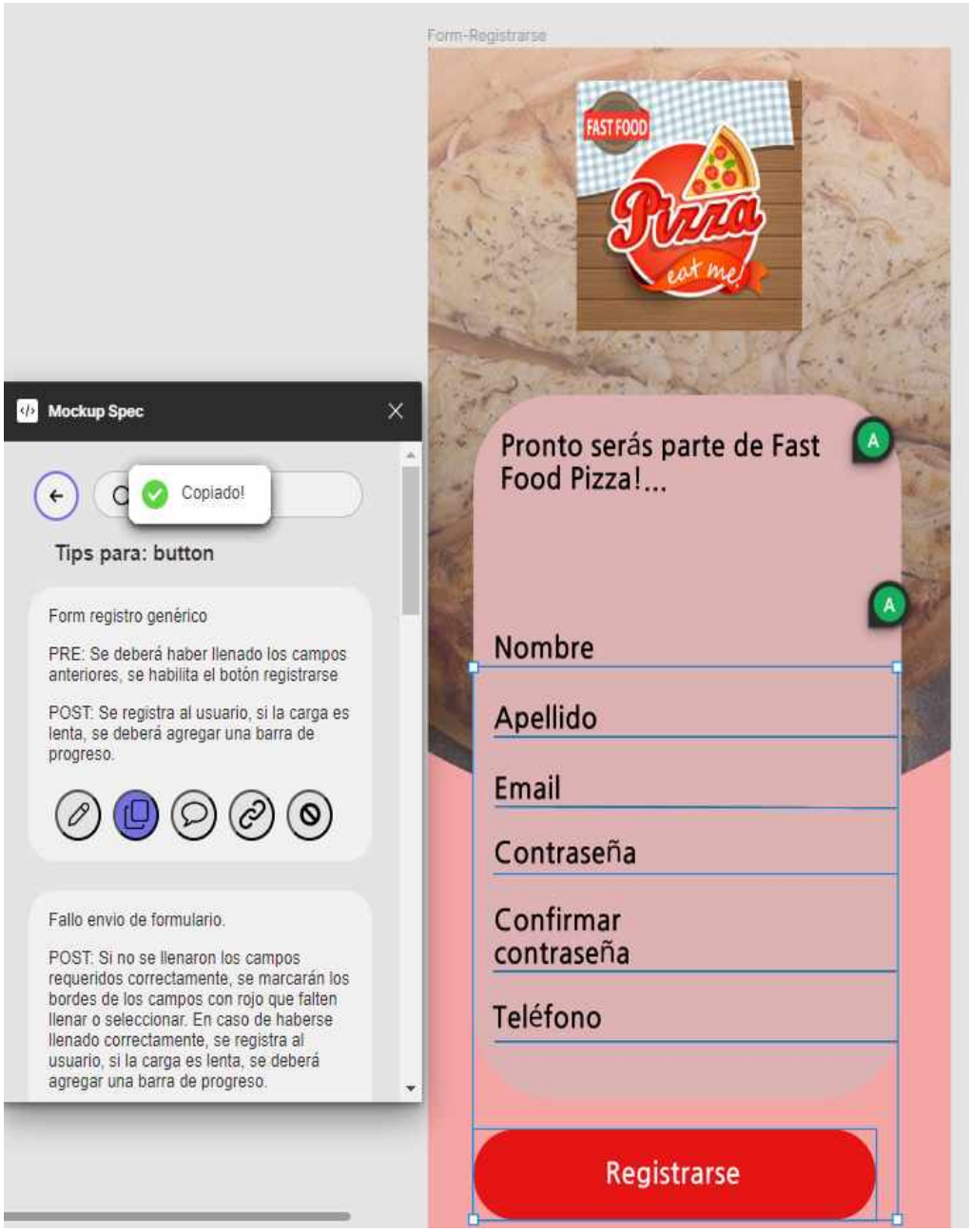


Fig.26 Copiar al portapapeles

El contenido se pega de la siguiente manera:

Form registro genérico

PRE: Se deberá haber llenado los campos anteriores, se habilita el botón registrarse

POST: Se registra al usuario, si la carga es lenta, se deberá agregar una barra de progreso.

5.3.1.1.3 Crear comentario

El tercer botón consiste en publicar un comentario, el contenido de dicho comentario será el tip. Para poder utilizar esta funcionalidad se deberá ingresar un token.**Fig.27**

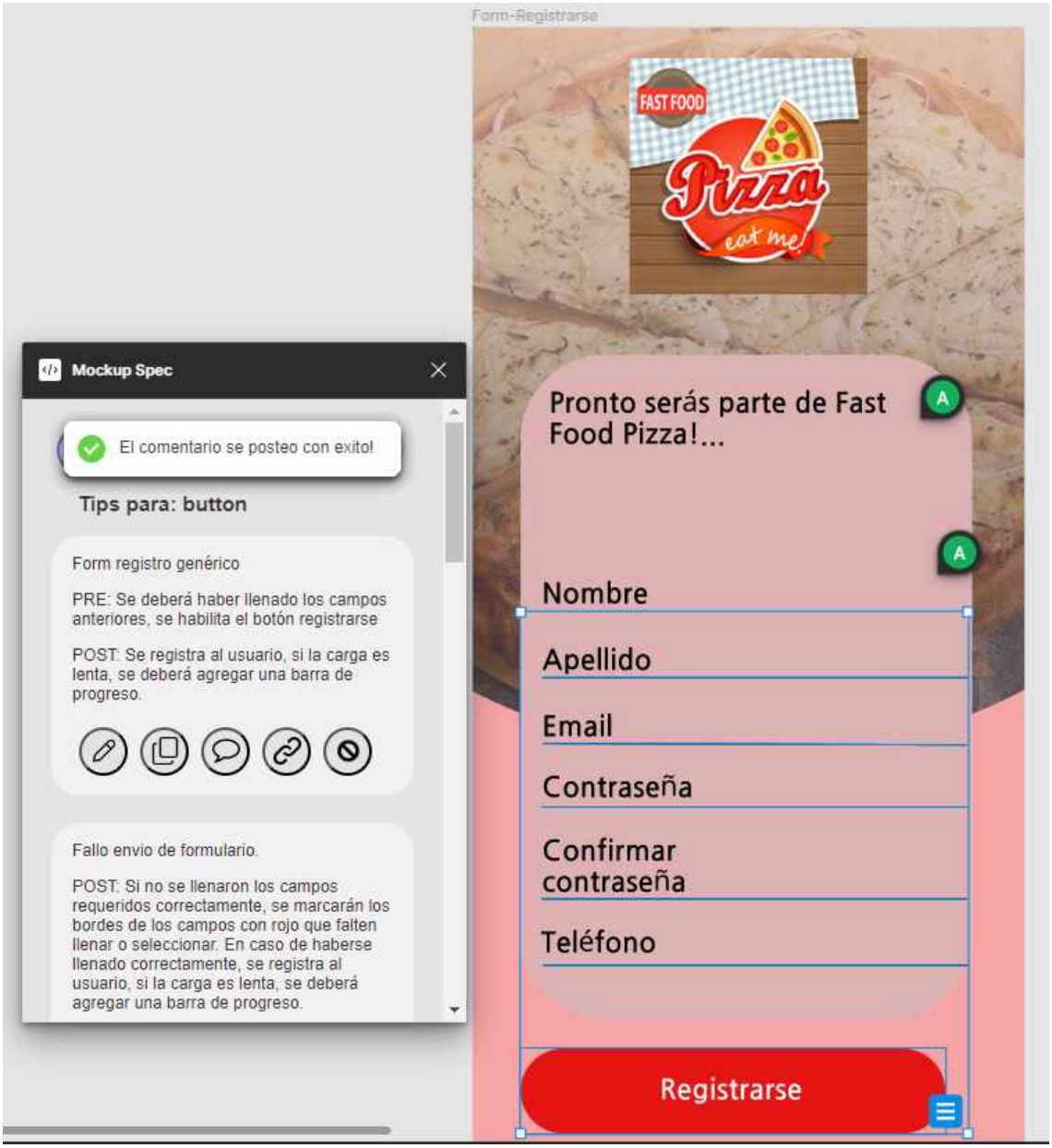


Fig.27 Crear comentario

El comentario se publicará y se mostrará en la vista de comentarios, sólo queda arrastrarlo hasta el lugar correspondiente.

5.3.1.4 Relacionar componentes

Cuarto botón consiste en relacionar los componentes. Al presionar el botón, se listará el resto de los componentes disponibles para relacionar, a través de un checkbox, seleccionaremos aquellos que queremos relacionar con ese componente y el estado, sí debe estar lleno, vacío, habilitado o deshabilitado.

Siguiendo el caso anterior sería. **Fig.28**

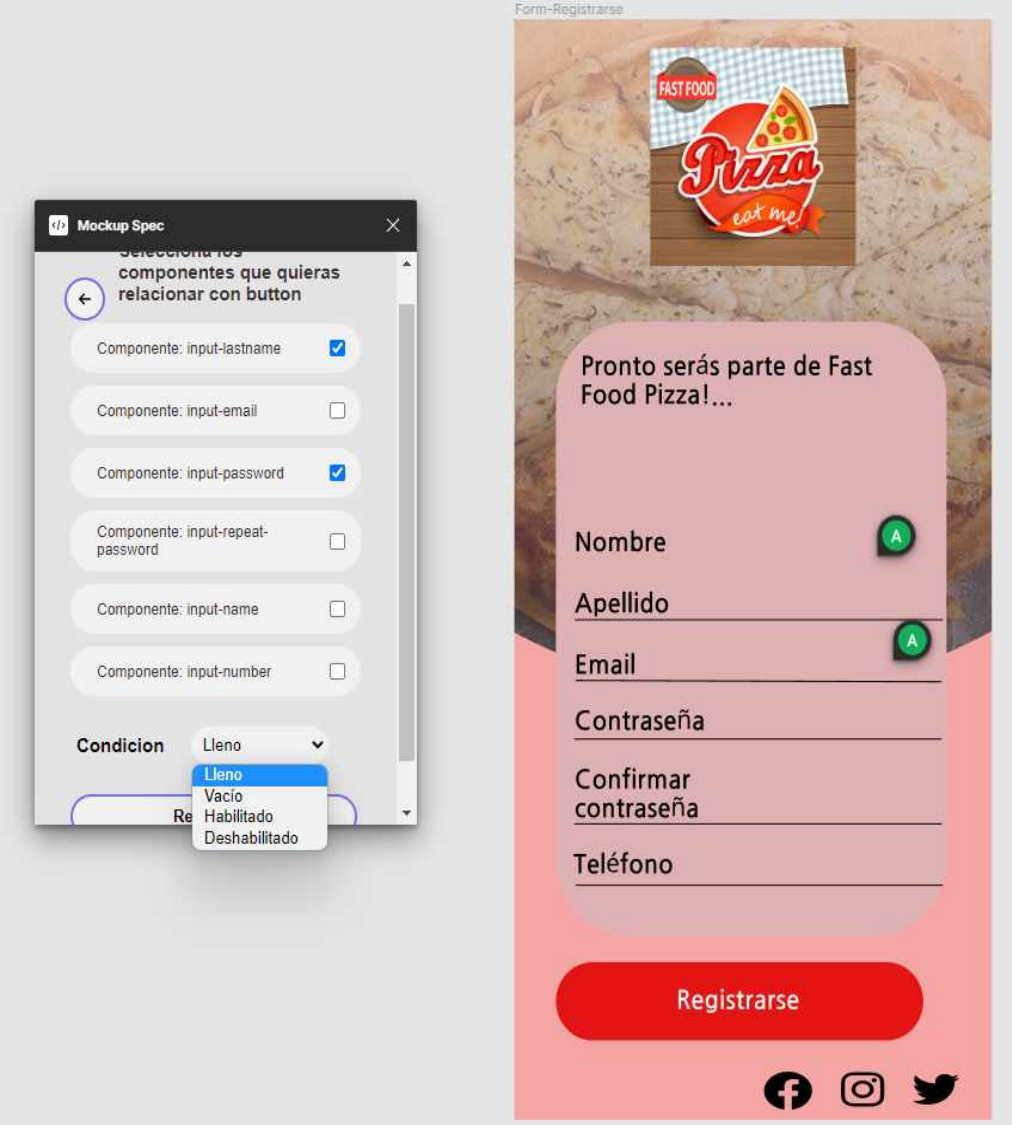


Fig.28 Relacionar componentes

Una vez que seleccionamos los componentes con el que queremos relacionar el componente actual (en este caso el botón), presionamos "*Relacionar*" y el tip quedará de la siguiente manera. **Fig.29**:

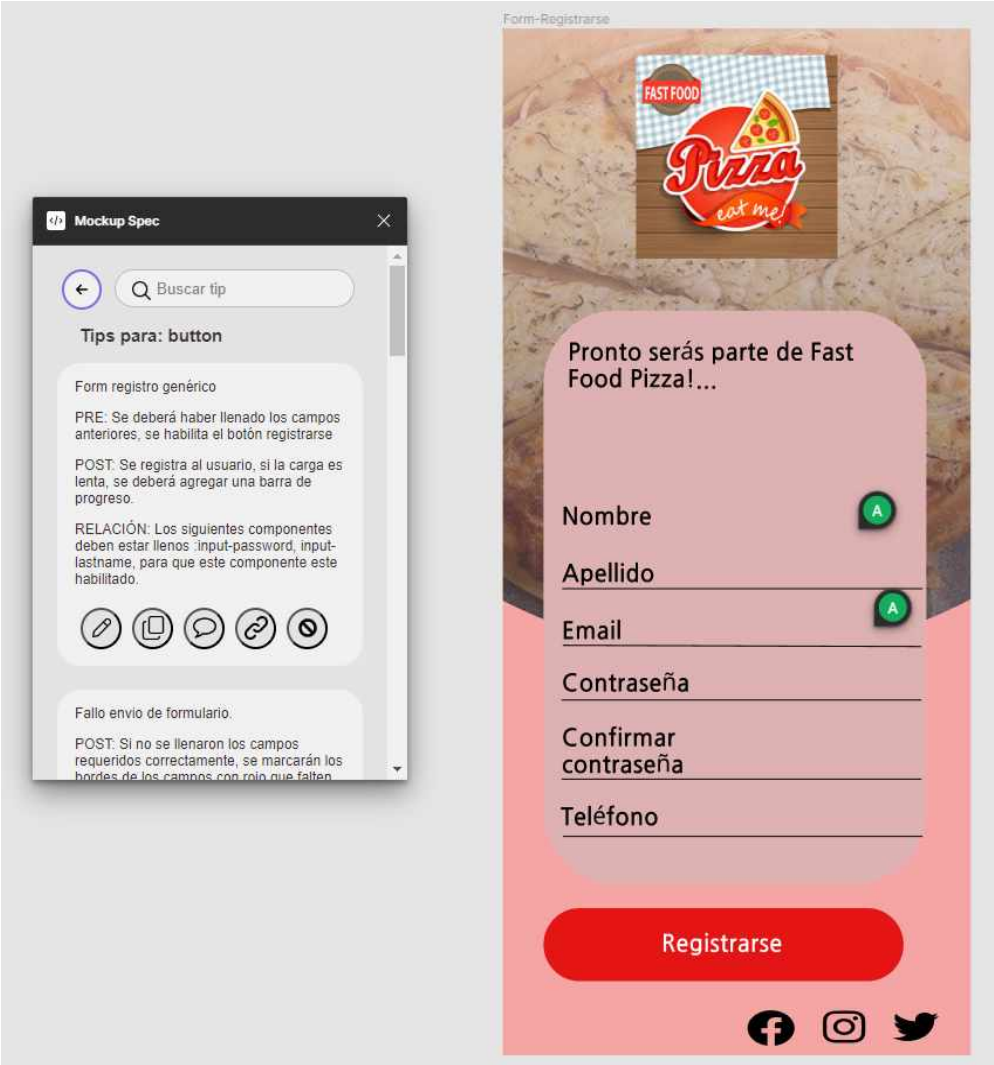


Fig.29. Resultado de relacionar componentes

5.3.1.5 Agregar restricción

5to y último botón, esta funcionalidad consiste en agregar algún tipo de restricción al campo. Como por ejemplo, la longitud del mismo, formato, etc. Para ello, presionaremos el 5to botón y se nos mostrará una lista de restricciones cargadas en el sistema. Asignaremos la que más nos convenga y presionaremos generar restricción. **Fig.30**

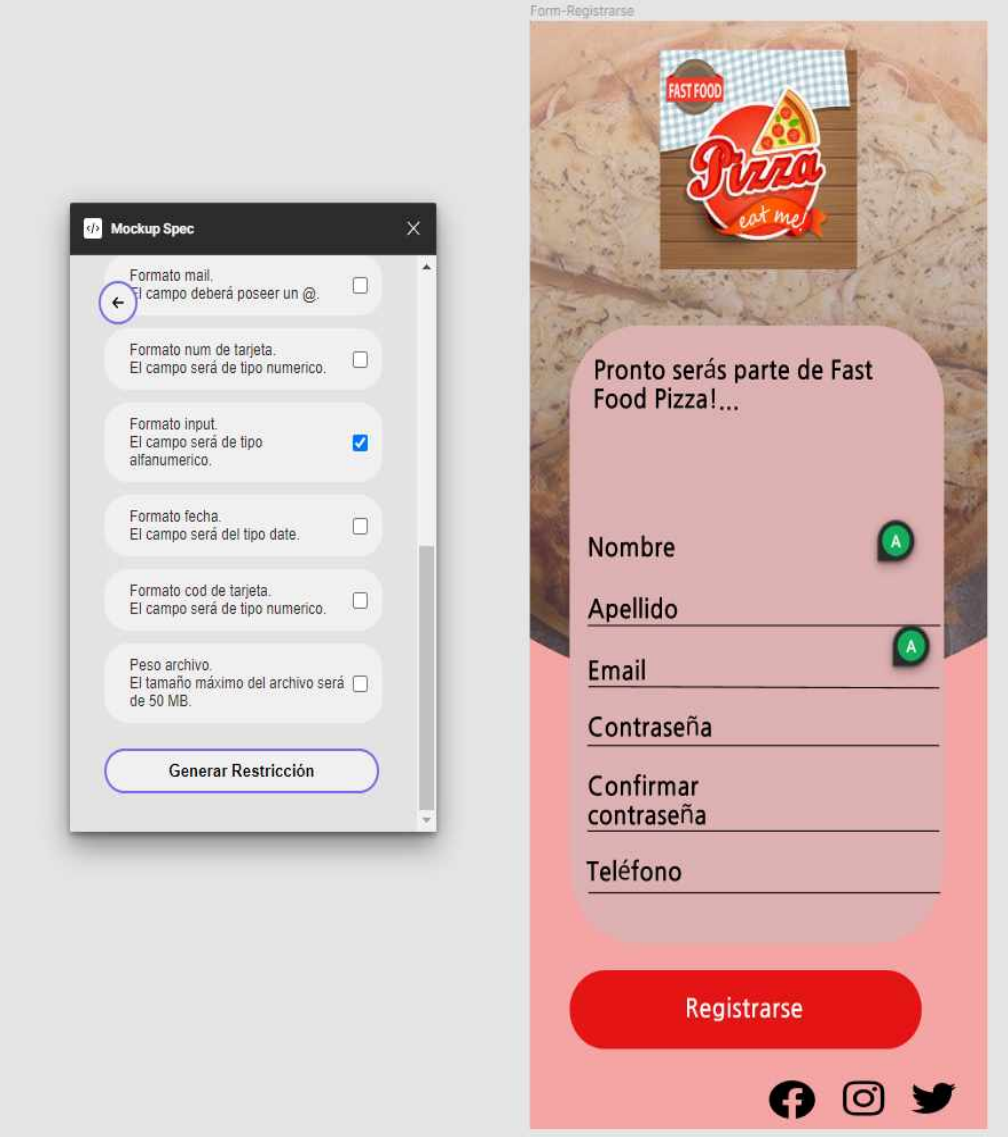


Fig.30 Agregar una restricción

Luego, se mostrará el tip de la siguiente manera. **Fig.31**

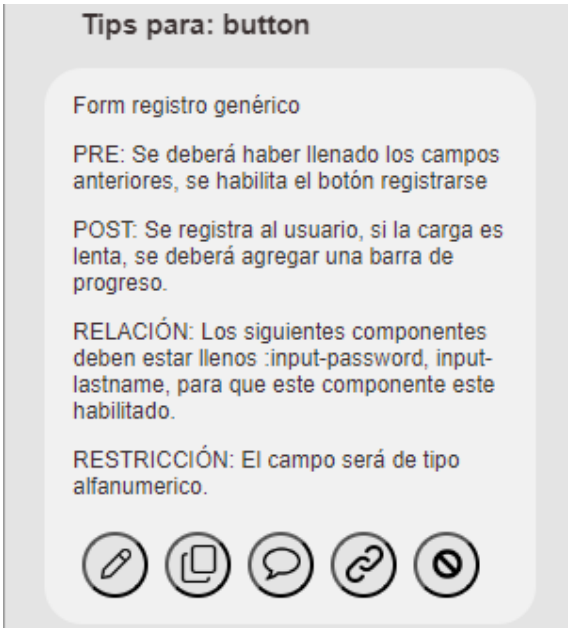


Fig.31 Resultado de agregar una restricción

5.3.1.2 Camino 2, método sin respetar la nomenclatura:

Se podrá utilizar todas las funcionalidades anteriormente mencionadas, excepto la relación entre componentes y crear comentario. La relación entre componentes no se podrá utilizar debido a que no se seleccionan componentes o los nombres de los componentes seleccionados no corresponden al nombre de tipo de acción. La funcionalidad crear comentario no se podrá utilizar si no se seleccionan componentes o elementos, debido a que no se podrá recuperar las coordenadas de dichos elementos para poder crear el comentario. Al intentar utilizar alguna de esas funcionalidades, se desplegará una alerta informando el evento. **Fig.32**

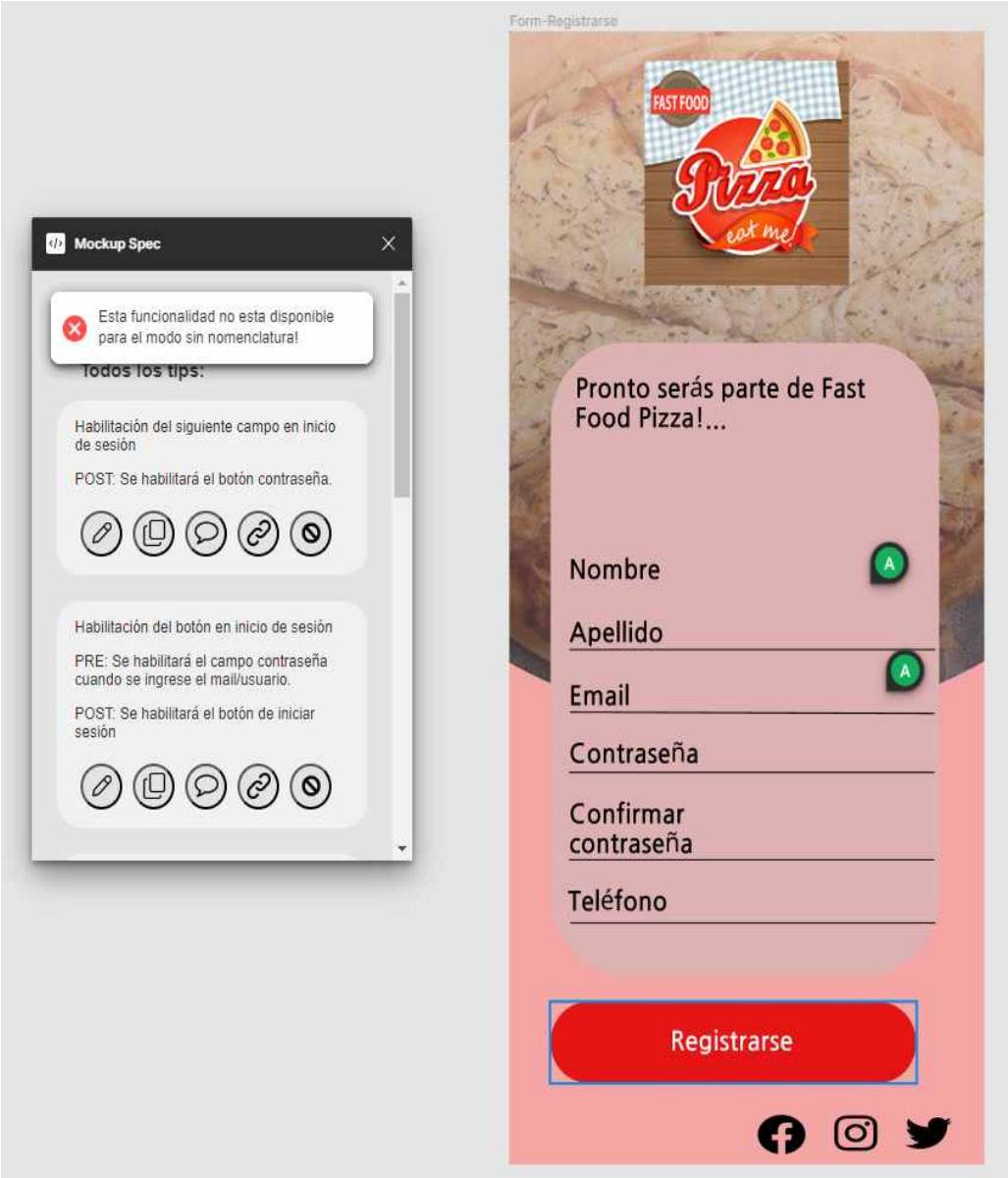


Fig.32 Limitaciones sin nomenclatura

En caso de querer utilizar el modo “Nomenclatura” cuando seleccionamos componentes que no respeten la nomenclatura, se desplegará una alerta informando que no se encontraron tips que tengan relación con los nombres de los componentes seleccionados. En este ejemplo, seleccionamos un componente que posee el nombre “fondo”. **Fig.33**

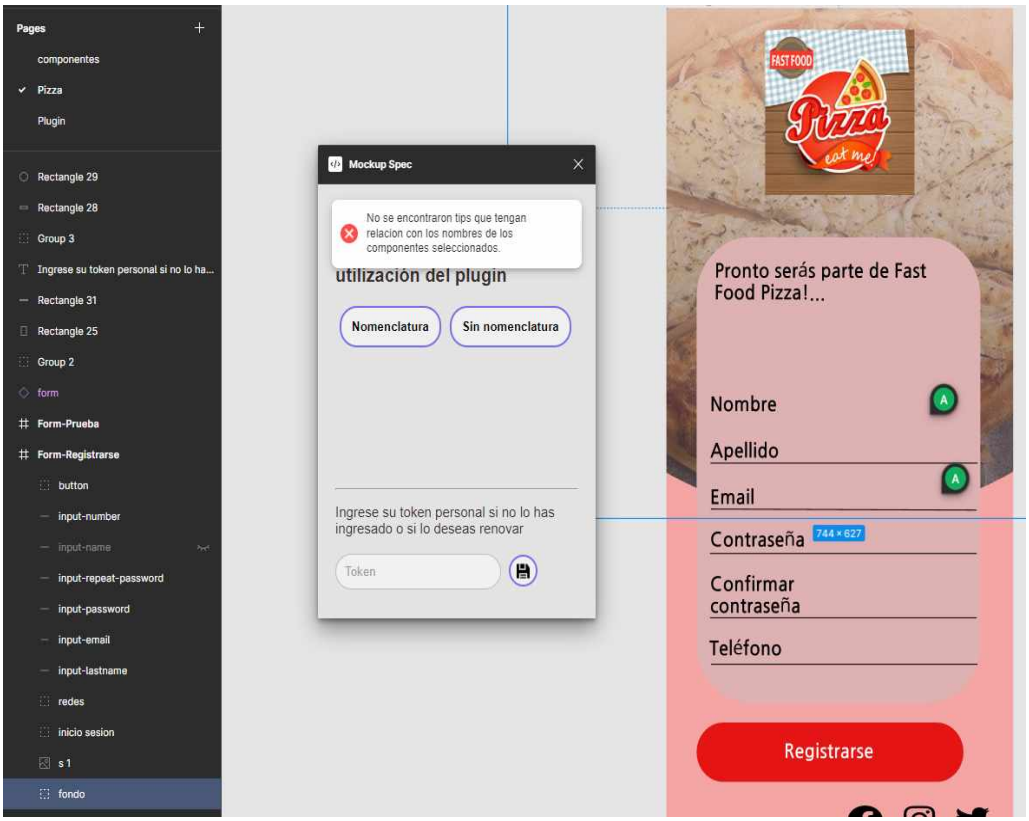


Fig.33 No se respeta la nomenclatura

En el caso que no seleccionemos componentes, se mostrará una alerta informando que se debe seleccionar un componente para poder ejecutar el modo "Nomenclatura". **Fig.34**

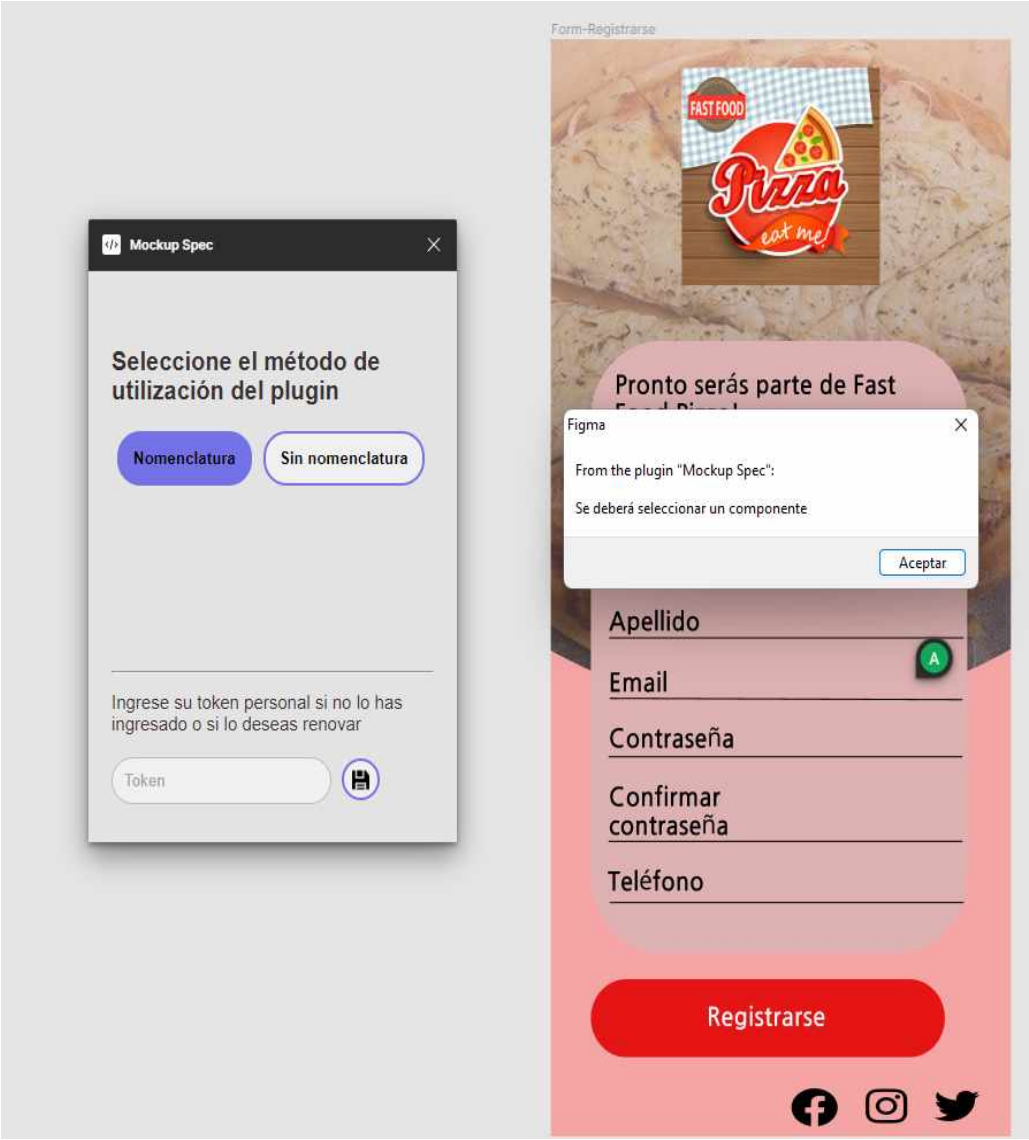


Fig.34. No se seleccionan componentes y se ejecuta el modo nomenclatura

5.3.1.3 Barra de búsqueda:

Permite buscar entre las diversas variantes, por nombre de la acción o nombre del componente. **Fig.35**

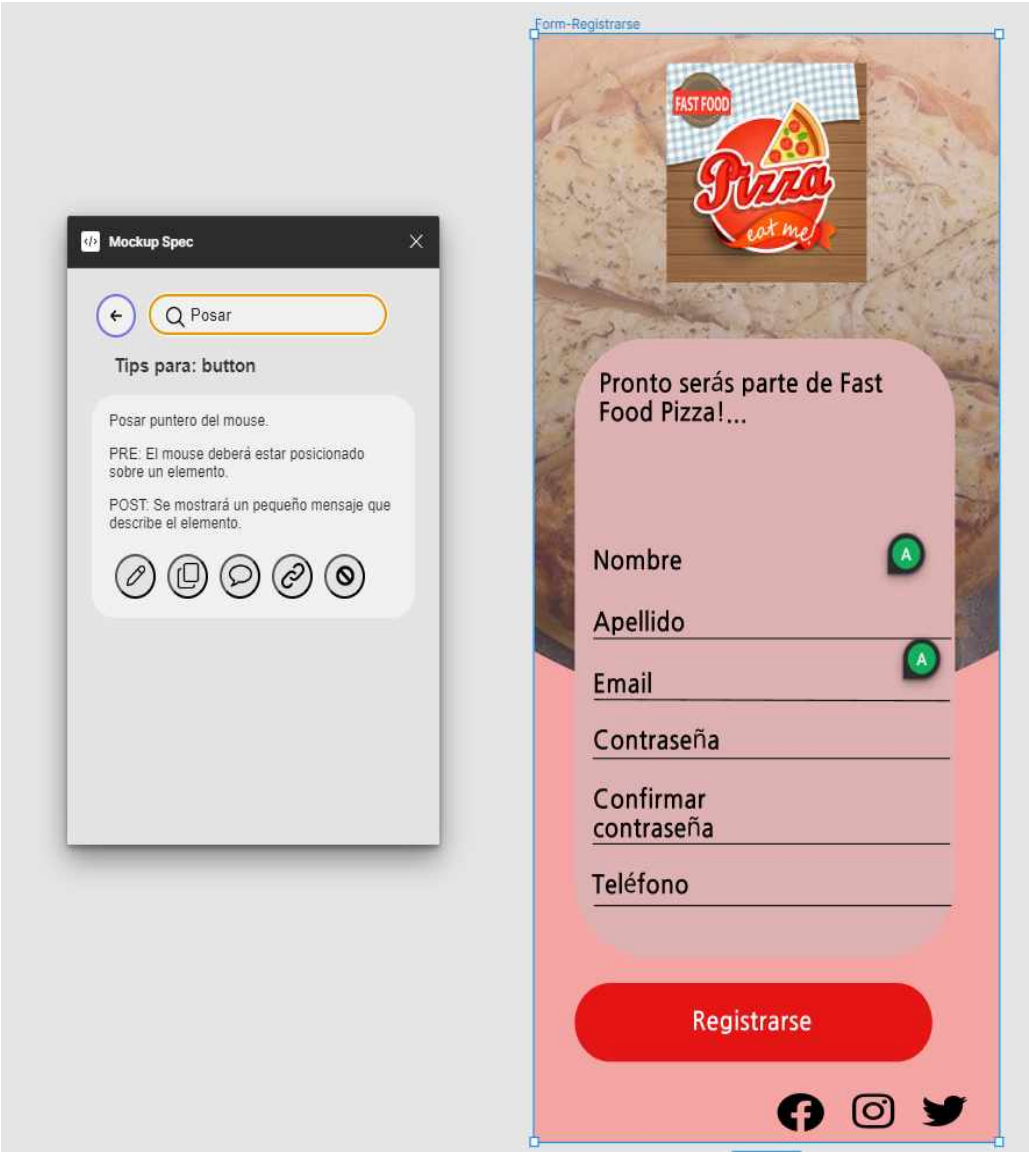


Fig.35. Barra de búsqueda

CAPÍTULO 6
Validación

En esta sección se describe el resultado de una evaluación del plugin, donde un diseñador probará la misma y deberá cumplir cierta cantidad de tareas, las cuales las planteamos a continuación. A su vez, le daremos una pequeña introducción del funcionamiento de la herramienta pero sin mostrarle cómo utilizar cada funcionalidad; esto se pensó así para ver el grado de eficacia de nuestra interfaz en cuanto a su facilidad de aprendizaje. Describiremos la tarea lo más concreto posible y registraremos los lugares donde más se equivoca o no sepa más avanzar.

Introducción: El propósito de nuestro plugin es disminuir el tiempo en rediseño y proveer una mejor comunicación especificando el diseño entre el diseñador y el área de desarrollo, con el fin de evitar ambigüedades. Por lo que utilizando nuestro plugin se obtendrá un diseño más específico y completo.

En cuanto a las funcionalidades del mismo, se deberá tener en cuenta una nomenclatura especial para utilizar al máximo sus funcionalidades. Con nomenclatura nos referimos a especificar el nombre del componente/elemento correctamente, deben contener en el nombre alguna de estas palabras: input, button, boton, link, checkbox, select o radio button. Una vez ejecutado el plugin, se listan los componentes y en base al componente seleccionado para trabajar, se listarán los tips asociados a ese tipo de componente. En cada tip se dispondrá de 5 botones con 5 funcionalidades diferentes, las cuales son: Editar tip, copiar tip al portapapeles, publicar comentario, relacionar componentes y crear una restricción.

En caso de no seguir la nomenclatura, se listará los tipos de tips (Acciones) y seleccionando uno de ellos se listaran los tips relacionados a ese tipo. La diferencia radica en la cantidad de funcionalidades que se podrá utilizar, quedando sólo editar tip, copiar tip al portapapeles y crear una restricción, por lo que el diseñador tendrá libertad de seleccionar el tipo de tips.

A continuación describiremos las dos tareas que se diseñaron para la prueba de usuario.

Tarea 1: Se pide crear un formulario para el área de desarrollo, donde se tendrá los siguientes campos:

-Nombre.
-Apellido.
-Fecha de nacimiento.
-Documento
-Email.
-Teléfono
-Archivo donde no debe superar los 200MB.
-Botón de enviar.

Se deberá tener en cuenta:
-El formato y la longitud de los mismos.
-Se podrá modificar el contenido del tip a gusto.
-Para que se habilite el botón deberán estar lleno todos los campos anteriores.

El diseño quedará a manos del usuario, eligiendo los colores y el formato que más le guste.

Tarea 2: Se pide modificar un formulario para el área de desarrollo, donde se deberá especificar el diseño, ya que hay ciertas ambigüedades que los desarrolladores no saben con exactitud de qué manera lo quiere el equipo de diseño.

{Se utilizará el formulario de ejemplo “Form-Registrarse”}
Consta de:

-Nombre.
-Apellido.
-Email.
-Contraseña
-Confirmar contraseña.
-Teléfono

Se solicita que el teléfono no sea obligatorio para que el botón enviar se habilite. A su vez, se pide que cada input se vaya habilitando a medida que se haya completado el anterior (exceptuando el teléfono). Se pide agregar un nuevo campo “Género“ de formato checkbox. Se pide restringir los campos nombre y apellido a sólo 20 caracteres de longitud.

6.1 Conclusiones de la validación

A continuación, describiremos un breve reporte sobre las tareas ejecutadas por parte del diseñador utilizando nuestra herramienta. Para obtener el detalle completo se encuentran disponibles los videos en el siguiente link: https://drive.google.com/drive/folders/1Je6p7osqaK3rKanmv1xR3SBZZvXWx9rY?usp=drive_link

En principio se notó una leve dificultad a la hora de ejecutar el plugin seleccionando los componentes, ya que no seleccionó ningún componente y lo ejecutó, por lo que no listaba los tips para los componentes que diseño. De todas formas supo ejecutar nuevamente el plugin seleccionando bien los componentes y así se listaron correctamente los tips.

Luego, detectamos que el usuario encontró un uso distinto de la herramienta el cual no estaba previsto en el diseño original que consistía en el uso de los tips, es decir, seleccionó un componente (en el ejemplo de tipo button) y en base a eso fue creando comentarios, y en el caso necesario, los editó. Por lo que las dos funcionalidades que más utilizó fueron “editar tip” y “crear comentario”.

Notamos que al agregar un nuevo componente, no le asignó correctamente el nombre según la nomenclatura especificada. Cabe destacar que este diseñador está acostumbrado a no poner nombre a los componentes. Sin embargo, nos aclaró que es una buena práctica tener una nomenclatura apropiada. Por último, observamos que al utilizar la restricción la hizo de forma manual y no utilizando el

botón “Agregar una restricción al componente”. Al cabo de un breve periodo de tiempo, localizó el botón y comenzó a utilizar la funcionalidad de restricción, editando lo necesario.

Una de las limitaciones que nos encontramos a la hora de grabar la validación, fue que en el entorno de Figma no tomaba el input del teclado mientras se estaba grabando la pantalla.

Al finalizar la prueba, el usuario contestó un cuestionario, el mismo se denomina “SUS” y estas siglas hacen referencia a System Usability Scale. Se trata de un método rápido para evaluar la usabilidad de un sistema [SUS]. Este método permite realizar una evaluación de:

- Eficacia: ¿los usuarios pueden alcanzar con éxito sus objetivos?
- Eficiencia: ¿cuánto esfuerzo es necesario para que pueda alcanzar esos objetivos?
- Satisfacción: ¿el uso del sistema fue satisfactorio?

A continuación detallamos el cuestionario junto con las respuestas del usuario:

1. Creo que me gustaría utilizar este sistema con frecuencia
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

2. Encontré el sistema innecesariamente complejo
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

3. Pensé que el sistema era fácil de usar
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

4. Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

5. Encontré que las diversas funciones de este sistema estaban bien integradas
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

6. Pensé que había demasiada inconsistencia en este sistema
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

7. Me imagino que la mayoría de la gente aprendería a utilizar este sistema muy rápidamente
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo

E. Totalmente de acuerdo

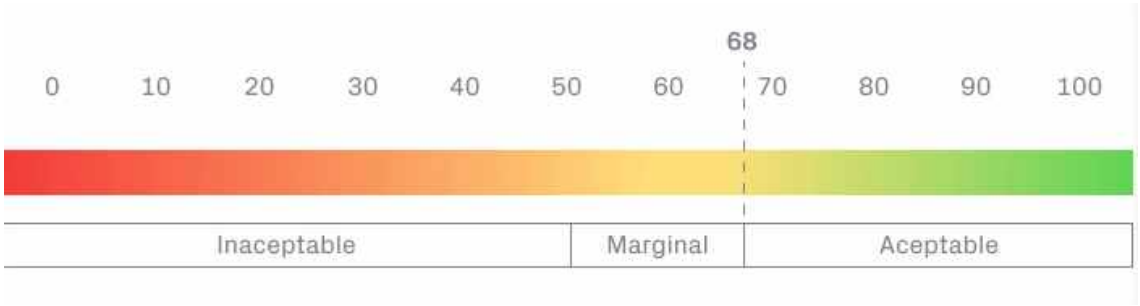
8. Encontré el sistema muy complicado de usar
   A. Totalmente en desacuerdo
   B. En desacuerdo
   C. Neutro
   D. De acuerdo
   E. Totalmente de acuerdo

9. Me sentí muy seguro usando el sistema
   A. Totalmente en desacuerdo
   B. En desacuerdo
   C. Neutro
   D. De acuerdo
   E. Totalmente de acuerdo

10. Necesitaba aprender muchas cosas antes de empezar con este sistema
    A. Totalmente en desacuerdo
    B. En desacuerdo
    C. Neutro
    D. De acuerdo
    E. Totalmente de acuerdo

Para calcular el resultado del SUS se realiza de la siguiente manera:
Sumamos las respuestas de los enunciados impares y después restamos 5.
Sumamos las respuestas de los enunciados pares y luego restamos ese total a 25.
Sumamos ambos resultados y lo multiplicamos por 2,5.
Realizamos el cálculo de nuestra herramienta:

Impares: (4 + 4 + 4 + 4 + 3) - 5 = 14
Pares: 25 - (2 + 1 + 2 + 2 + 3) = 15
Total: (14 + 15) * 2,5 =**72,5**



A pesar que el SUS administrado sólo con un usuario no puede dar un dato concluyente estos resultados preliminares son alentadores sobre la usabilidad de la herramienta.

CAPÍTULO 7

Conclusiones

Comenzamos este trabajo investigando y analizando sobre los smells de UX y las formas de solucionarlo. En base a ello, nos fuimos dando cuenta que había un problema que se repite reiteradas veces: la falta de comunicación entre el área de diseño y desarrollo. A su vez, notamos que en la práctica la atención que se le presta al diseño de la interfaz de usuario y la UX es menor a las nuevas funcionalidades o arreglo de bugs.

A causa de eso, entendimos que se necesitaba una herramienta con la cual lograr disminuir los smells de UX y la mala comunicación entre ambas áreas, y en consecuencia se pueda optimizar tanto el proceso de diseño como el de desarrollo de las interfaces. Cabe destacar que este es un proyecto de software libre y abierto, por lo que podrá ser extendido o modificado por cualquier persona que lo desee para mejorarlo o agregar más tips o restricciones

Desde el punto de vista tecnológico, ambos tuvimos que aprender sobre tecnologías no estudiadas previamente en la carrera, como Figma, TypeScript y React.
El resultado final es una herramienta para el entorno de Figma que hace uso de las últimas tecnologías para lograr una experiencia de diseño sencilla y detallada para los usuarios con el objetivo principal de incorporar especificación en el diseño interactivo a través del uso de las mejores prácticas y pudiendo así evitar ambigüedades entre el área de diseño y el de desarrollo.

7.1 Contribuciones

En nuestra herramienta, los objetivos principales siempre fueron que el diseñador tenga la posibilidad de especificar el diseño interactivo de la mejor manera, utilizando los tips, restricciones, etc para lograr una simplificación del trabajo en el área de desarrollo y el área de diseño; como consecuencia se reducirán los desacuerdos entre el diseñador y el desarrollador, optimizando el tiempo de un rediseño de las interfaces por una mala comunicación o mala especificación. Con estos objetivos descritos podemos concluir que nuestra herramienta cumplió con el

objetivo de poder realizar una especificación más completa y detallada de las decisiones del diseño, es decir, el diseño estaría más completo utilizando las funcionalidades que brindamos en el plugin.

7.2 Limitaciones

“Mockup Spec” es una herramienta que sólo trabaja dentro del entorno de Figma, por lo que no es usable fuera del mismo.

Figma no provee tipos de objetos, por lo que no se puede saber si un componente es de tipo input, button, etc. Para ello, se debe especificar en el nombre del componente el tipo que es. De esta manera, el plugin detecta el tipo y lista la acción acorde al mismo. Debido a esto, especificamos la correcta nomenclatura que se debe respetar para aprovechar al máximo la herramienta.

A su vez, esta herramienta sólo está desarrollada con una lista de tips que ayudan a evitar los smells listados en la sección 2.4.2. Para nuevos smells puede que se necesite la definición de nuevos tips.

7.3 Trabajos futuros

En el transcurso de la realización de este trabajo nos encontramos con diferentes puntos de extensión que valdría la pena encarar como posibles extensiones en un futuro. A continuación describimos cada uno de ellos:

- Chequear automáticamente si hay UX smells.
- Importar/Exportar el diseño en otro entorno del trabajo. Figma es una herramienta que nos limita bastante para poder identificar distintos componentes, sólo lo podemos hacer a través del nombre. De esta manera, al llevarlo a otro entorno de trabajo, se podría pensar en una lógica que haga un proceso más complejo.
- Exportar los tips a un documento de especificación de UX. Básicamente sería un documento o informe donde se detalle los componentes que posean un tip, dicho tip puede estar compuesto por su descripción, su pre y post condición, sus enlaces (si está

relacionado o no) y si tiene alguna restricción. El fin de este documento sería tener un acceso más limpio al diseño especificado.

- Asociar la cuenta de Figma dentro de nuestra herramienta. El beneficio de ello sería para poder almacenar los tips y/o restricciones favoritas del diseñador, con el fin de serle más sencillo el alcance de los tips más utilizados. A su vez, se podría investigar si se puede almacenar los tips creados por el diseñador, para no tener que crearlos nuevamente.
- Comunidad de tips. Podría haber una sección donde cada diseñador publica sus tips o restricciones.

REFERENCIAS BIBLIOGRÁFICAS

-[APIfigma] https://www.figma.com/developers/api

-[Ashmore 14] Ashmore, S., & Runyan, K. (2014). *Introduction to agile methods*. Addison-Wesley Professional.

[Baltes 21] S. Baltes, V. Dashuber,*Ux debt: Developers borrow while users pay*, arXiv preprint arXiv:2104.06908 (2021).

-[Brhel15] -Brhel, Meth, Maedche, Werder,*Exploring principles of user-centered agile software development: A literature review*, Information and Software Technology (2015).

-[Cunningham 92] W. Cunningham,*The wycash portfolio management system*, ACM SIGPLAN OOPS Messenger 4 (1992).

- [DaFonseca 19] L. da Fonseca Lage, M. Kalinowski, D. Trevisan, R. Spinola, *Usability technical debt in software projects: A multi-case study*, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, IEEE, 2019.

-[DaSilva 11] T. S. Da Silva, A. Martin, F. Maurer, M. Silveira,*User-centered design and agile methods: A systematic review*, in: Agile 2011, 2011.

-[DocumentacionFigma] https://www.figma.com/plugin-docs/

- [Figma] Figma: la herramienta de diseño de interfaz colaborativa. www.figma.com.

-[globalObjects] https://www.figma.com/plugin-docs/api/global-objects

-[GDSS19] Garcia A., Da Silva T. S., Silveira M. S.*Artifact-Facilitated Communication in Agile User-Centered Design.*2019

-[GRD11] Garrido A., Rossi G., Distante D.,*Refactoring for Usability in Web Applications.*2011

-[GS21] J. Gothelf, J. Seiden,*Lean UX: Applying Lean Principles to Improve User Experience*, O'Reilly Media, Inc., 2021.

-[GGRR17] J. Grigera, A. Garrido, JM. Rivero, G. Rossi. Automatic detection of usability smells in web applications.2017

-[JHM14] Jurca G., Hellmann T. D, Maurer F.,*Integrating Agile and User-Centered Design.*2014

-[LGG] Lubomirsky Ana Liz, Gardey Juan Cruz, Garrido Alejandra.*Análisis de deuda técnica de UX en repositorios de GitHub.*

- [Man97] Mandel, Theo (1997).*The Elements of User Interface Design, Wiley*

-[Nielsen94] J. Nielsen,*10 Usability Heuristics for User Interface Design*, [online] Available: http://www.nngroup.com/articles/ten-usability-heuristics (1994)

-[SRP19] Helen Sharp, Yvonne Rogers, Jennifer Preece. (2019).*Interaction Design: beyond human-computer interaction,*Fifth Edition (pp 472, 473)

-[SSPSJ19] Suleri S., Sermuga Pandian V. P., Shishkovets S., Jarke M.,*EVE: A Sketch-based Software Prototyping Workbench.*2019

-[Somerville 11] -Baxter, G., and Sommerville, I. (2011) Socio-Technical Systems: From Design Methods to Systems Engineering, Interacting with Computers, 23, (1) 4–17.

-[Sommerville16] Ian Sommerville (2016).*Software Engineering,*tenth edition (p 74).

-[SUS] https://www.uifrommars.com/como-medir-usabilidad-que-es-sus/

-[template] https://github.com/nirsky/figma-plugin-react-template

- [Pressman10] Roger S. Pressman (2010).*Ingeniería del software.Un enfoque practico*. Séptima edición. (p 339, 266) (p 69-70)

-[RESTAPI] https://www.figma.com/developers/api

-[Rodriguez23] Rodriguez, A., Gardey, J.C., Grigera, J., Rossi, G. & Garrido, A. UX debt in an agile development process: evidence and characterization. Software Quality Journal (2023). Springer.