Editor de AquilesWire

En el competitivo mundo de las barandillas es importante mantener la innovación. Nuestro cliente está fuertemente concienciado con ello y nos ha pedido que desarrollemos un estándar internacional para la definición de formas de barandillas. Además, con vistas a convertirse en un referente internacional en el sector, nos ha pedido un editor para el nuevo estándar, de manera que cualquier de sus potenciales clientes e incluso su competencia lo adopten como solución de facto. Para potenciar la difusión, su intención es distribuir el producto como open source y promocionarlo hasta hacerlo viral.

Actualmente ya hemos definido el estándar (que describe la forma de la barandilla con un código alfanumérico) y queremos empezar el desarrollo del editor, que debe ser ejemplar y moderno ya que también llevará nuestra marca y su código quedará expuesto al mundo. Aunque lo principal es que sea robusto y fiable, por supuesto.

Como nuevo miembro del equipo y como primera tarea, se te pedirá que desarrolles un prototipo del editor, siguiendo el estándar ya definido y aplicando algunas validaciones básicas.

Tecnología y diseño

El prototipo será totalmente frontend y podrás desarrollarlo con el framework de tu elección (Angular, React o Vue). Aunque la arquitectura será sencilla, deberá emplear varios componentes así como otros recursos típicos del framework. Podrás emplear el boilerplate con el que te sientas más cómodo, incluyendo tus herramientas habituales de trabajo como Webpack /Parcel, etc.

Para esta fase de prototipado el diseño no es importante por lo que no se tendrá en cuenta, la UI ha de ser mínimamente clara para un usuario técnico, pero no consideraremos la estética.

Funcionalidades

Como en nuestro equipo estamos habituados a tratar con el cliente directamente, a continuación tienes los requisitos funcionales en forma de las user stories que hemos acordado con él.

#1 El usuario puede ver en pantalla todas las formas cargadas en la aplicación.

Para cada forma tendrá su visualización y un campo de texto con su código.

#2 El usuario puede editar cualquiera de las formas que tiene en la lista.

El campo del código de la forma es editable de manera que su visualización se actualiza mientras el usuario escribe.

#3 El usuario puede ver, para cada forma, qué validaciones incumple o si las cumple todas.

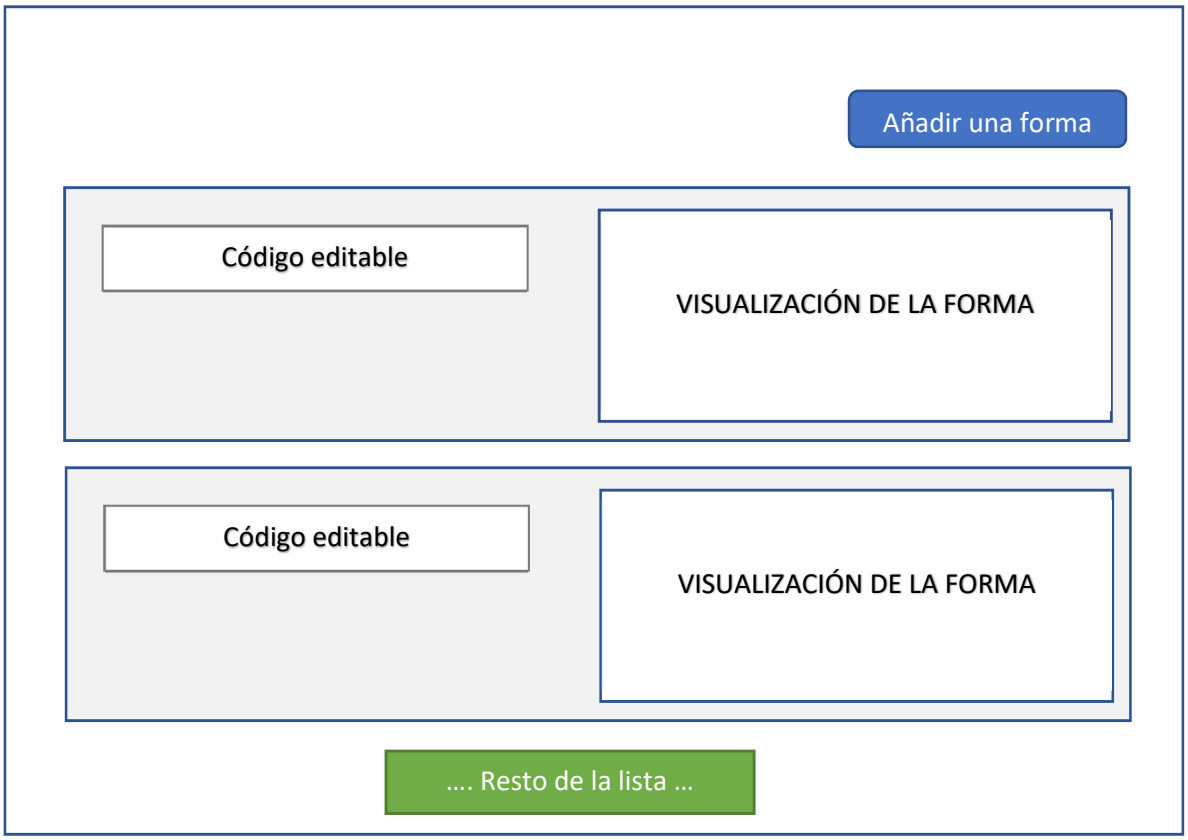
Las validaciones se detallan más adelante.

#4 El usuario puede añadir una forma nueva a la lista que tiene en pantalla.

Inicialmente la forma tendrá un código vacío que el usuario podrá cumplimentar.

Diseño esquemático

Aunque no tenemos un diseño gráfico trabajado como tal, te ofrecemos un esquema de cómo podría ser el prototipo.



Especificaciones

Definición del estándar

Nuestro nuevo estándar para definición de barras de acero es bastante sencillo, básicamente describe la barra como una sucesión de segmentos rectos indicando el ángulo que hay entre cada segmento y el siguiente.

El código tiene la siguiente estructura:

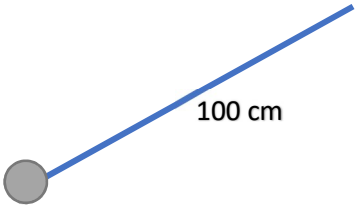
- La cadena **AQW#** como inicio.
- Una sucesión de segmentos descritos con su ángulo de giro inicial y su longitud:
  - El símbolo **@** precede al ángulo de la barra en grados sexagesimales y sin decimales.
  - El símbolo **L** precede a la longitud del segmente en centímetros, también sin decimales.
- El símbolo **#** como final del código.

El ángulo del primer segmento define la orientación del primer segmento, mientras que los demás ángulos indican el giro del siguiente segmento con respecto al anterior.

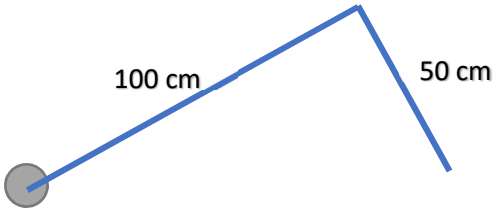
A continuación, se muestran algunos ejemplos con su visualización. El punto gris sólo indica el inicio de la barra, no deberá mostrarse en la visualización.

- AQW#@0L100@90L50#

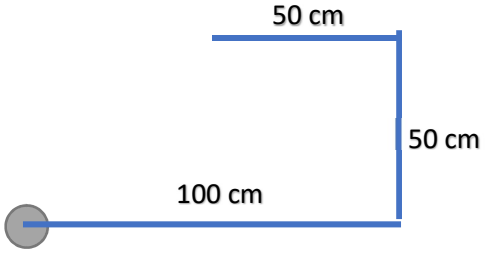
- AQW#@45L100#



- AQW#@45L100@-90L50#



- AQW#@0L100@90L50@90L50#



Validaciones

Las validaciones serán una de las funcionalidades clave del producto final pero en esta primera fase sólo contemplaremos algunas básicas.

Las validaciones previstas son:

- #1 El código de la forma es correcto según el estándar definido.
- #2 La suma total de longitudes de los segmentos es menor a 12 metros.
- #3 Ningún ángulo es mayor de 165º ni menor de -165º.

Datos de partida

Se proporciona un archivo *formas.json* con algunos ejemplos que cargar en la aplicación.

Librerías

Además del framework escogido puedes emplear las librerías que prefieras, con algunas salvedades que se comentan a continuación.

Para nosotros es clave entender el estándar por lo **que tanto la validación del formato del código** (validación #1) **como la definición de qué dibujar debe estar escrito por ti mismo**. Puedes emplear la librería que prefieras para el dibujado en sí (Paper.js o Rough.js por ejemplo) pero es importante que tu código gestione los giros y las longitudes de cada segmento.

Notas adicionales

La visualización puede estar a cualquier escala y con cualquier ancho de línea mientras la forma se vea suficientemente clara y las distancias sean proporcionales.

Puedes realizar cualquier asunción que necesites para complementar este documento, siempre y cuando nos las envíes junto con tu solución.