

Pika

1	Introduzione	3
1.1	Informazioni sul progetto	3
1.2	Abstract	3
1.3	Scopo	3
Analisi		4
1.4	Analisi del dominio	4
1.5	Analisi e specifica dei requisiti	4
1.6	Use case	7
1.7	Pianificazione	8
1.7.1	Analisi dei mezzi	1
1.7.2	Software	1
1.7.3	Hardware	1
2	Progettazione	1
2.1	Struttura file JS	1
3	Implementazione	2
3.1.1	Index	2
3.2	Classi JS	5
3.2.1	Point	5
3.2.2	Rectangle	6
3.2.3	Circle	8
3.2.4	Pencil	9
3.3	Action	11
3.3.1	ActionPoints	11
3.3.2	ActionRects	14
3.3.3	actionCircle	16
3.3.4	actionLines	17
3.4	drawHelp.js	18
3.5	main	19
3.6	imageImport	20
3.7	ImageExport	21
4	Test	22
4.1	Protocollo di test	22
4.2	Risultati test	24
4.3	Mancanze/limitazioni conosciute	25
4.3.1	Strumento secchiello	25
5	Consuntivo	25
6	Conclusioni	25
6.1	Sviluppi futuri	26
6.2	Considerazioni personali	26
7	Glossario	26
8	Allegati	26

1 Introduzione

1.1 Informazioni sul progetto

Progetto sviluppato da Gioele Chiodoni sotto supervisione del docente Geo Petrini dal 01.09.2023 al 1.12.2023.

Allievo della Scuola Arti e Mestieri di Trevano nella classe I3BB anno 2023/2024.

1.2 Abstract

We find ourselves in an era where people are increasingly writing on computers. And because of this they are no longer able to write in beautiful handwriting.

This application aims to create fun exercises for children to improve handwriting.

To do this, the application requires simplicity of use and intuitiveness.

The exercise that allows you to create is an image of dots which, if united, form an image.

1.3 Scopo

Lo scopo didattico del progetto è quello di imparare a gestire nel modo poi autonomo possibile e ottimale un progetto IT in vista dell'esame pratico a fine quarta.

Mentre lo scopo operativo è quello di creare un'applicazione web o non per poter creare immagini a puntini che, se uniti formano un disegno. L'applicativo deve permettere di importare, disegnare ed esportare le immagini. L'applicazione dovrà essere facile e intuitiva in modo da essere accessibile a chiunque.

Pika ha lo scopo di essere un'applicazione semplice e intuitiva per poter creare esercizi con scopo didattico per i bambini per migliorare la calligrafica e la mobilità della mano.

Analisi

1.4 Analisi del dominio

Questo applicativo dovrà essere semplice e intuitivo da utilizzare per poter essere accessibile a chiunque abbia un computer. Gli utilizzatori principali per qui è pensato il progetto sono i genitori e i docenti per generare esercizi per i bambini. Questi non dovranno avere conoscenze o praticità con il computer per poter usare il programma.

Grazie a quest'applicazione sarà possibile creare degli esercizi pensati per i bambini di modo da migliorare la mobilità della mano.

1.5 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Manipolazione puntini
Priorità	1
Versione	1.0
Note	L'utente deve poter inserire i puntini a piacimento.
Sotto requisiti	
001	Deve anche poter cambiarne l'ordine.
002	Deve anche poter spostarli.
003	Deve anche poter eliminarli.

ID: REQ-02	
Nome	Strumenti di disegno
Priorità	1
Versione	1.0
Note	L'utente deve poter disegnare sull'immagine finale per poter aggiungere parti del disegno.
Sotto requisiti	
001	Deve poter usare gli strumenti: Penna, cerchio, rettangolo, secchiello.

ID: REQ-03	
Nome	Esportare l'immagine
Priorità	1
Versione	1.0
Note	Una volta finito di mettere i puntini l'utente deve poter esportare l'immagine con i punti in PNG.
Sotto requisiti	
001	Se fattibile anche in formato vettoriale.

ID: REQ-04	
Nome	Facilità
Priorità	1
Versione	1.0
Note	L'applicativo deve essere facile da usare

ID: REQ-05	
Nome	Soluzione
Priorità	1
Versione	1.0
Note	Deve essere possibile vedere la soluzione del disegno
Sotto requisiti	
001	Con e senza l'immagine finale.

ID: REQ-06	
Nome	Layer
Priorità	1
Versione	1.0
Note	Deve poter essere possibile scegliere cosa deve essere visto.
Sotto requisiti	
001	Si deve poter nascondere l'immagine
002	Si deve poter nascondere i puntini
003	Si deve poter nascondere la soluzione

ID: REQ-07	
Nome	Formato
Priorità	1
Versione	1.0
Note	Le immagini importate devono essere di tipo WEBP, JPEG, PNG

1.6 Use case

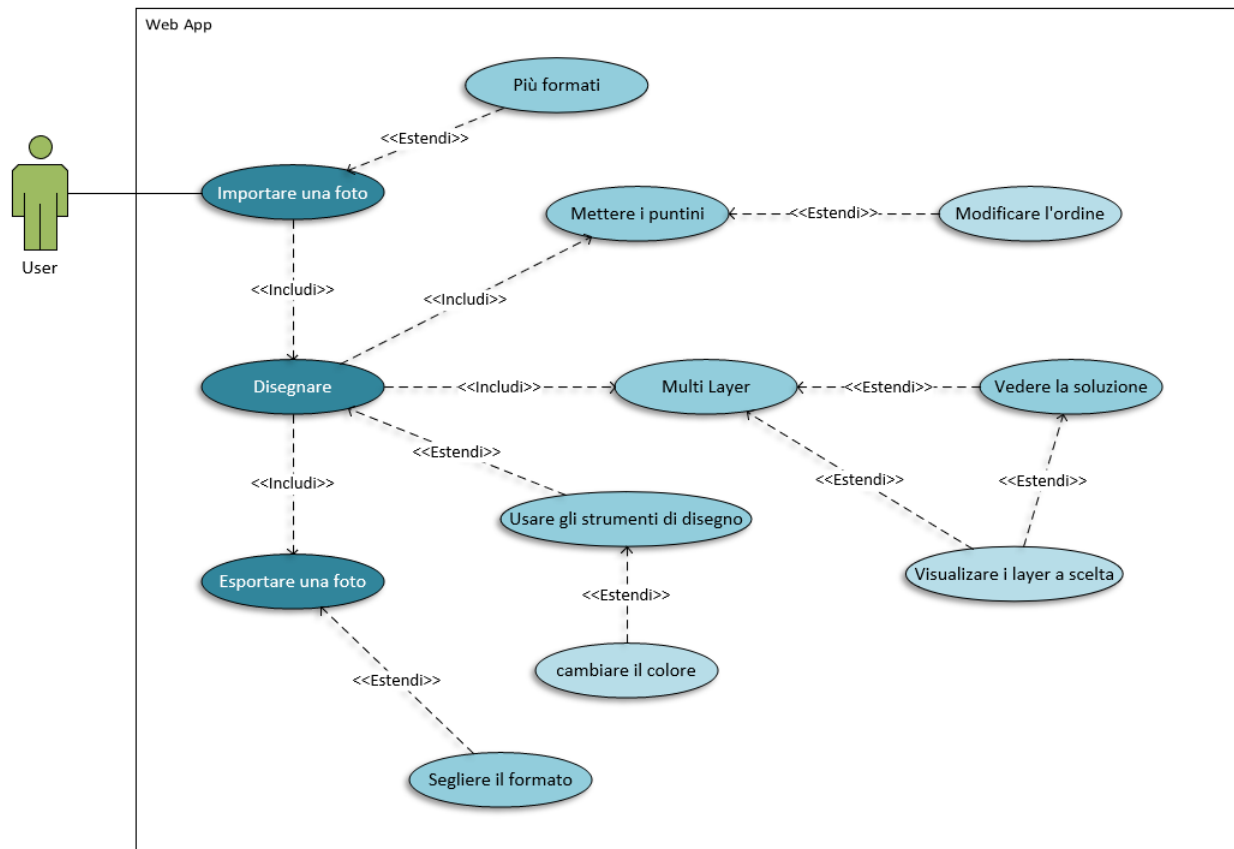


Figura 1 - Use Case

L'unico utente è l'admin che userà l'applicazione. Quest'ultimo per prima cosa potrà solo importare un'immagine. Una volta importata potrà fare diverse azioni: disegnare, manipolare i disegni, ecc. Una volta finito il disegno potrà esportare l'immagine nel formato desiderato.

1.7 Pianificazione

Il progetto è partito il 30.08.2023 ed è terminato il 1.12.2023 con la consegna del lavoro.

Per pianificare il progetto ho usato il modello water fall.

Durante la pianificazione ho tenuto un margine d'errore per aver margine in caso di imprevisti o nel caso di un giorno di malattia.

Le macrocategorie in cui ho incluso le attività sono:

- **Analisi:** All'interno di questa categoria ho messo tutto ciò che riguarda la pianificazione, la raccolta dei requisiti e le scelte tecniche iniziali. A questa categoria ho assegnato 1.5 giorni.
- **Progettazione:** All'interno di questa categoria ho inserito le azioni che riguardano la progettazione tecniche come: design delle classi e il design dell'interfaccia grafica. Per questa categoria ho pianificato 1 giorno.
- **Implementazione:** All'interno dell'implementazione ho inserito tutte le azioni che riguardano lo sviluppo effettivo dell'applicazione. Per lo sviluppo ho pianificato 7.5 giorni.
- **Test:** Nei test ho messo tutto ciò che riguarda testare il corretto funzionamento dell'applicazione e i suoi ritocchi necessari. Per questa categoria ho pianificato 1.5 giorni.

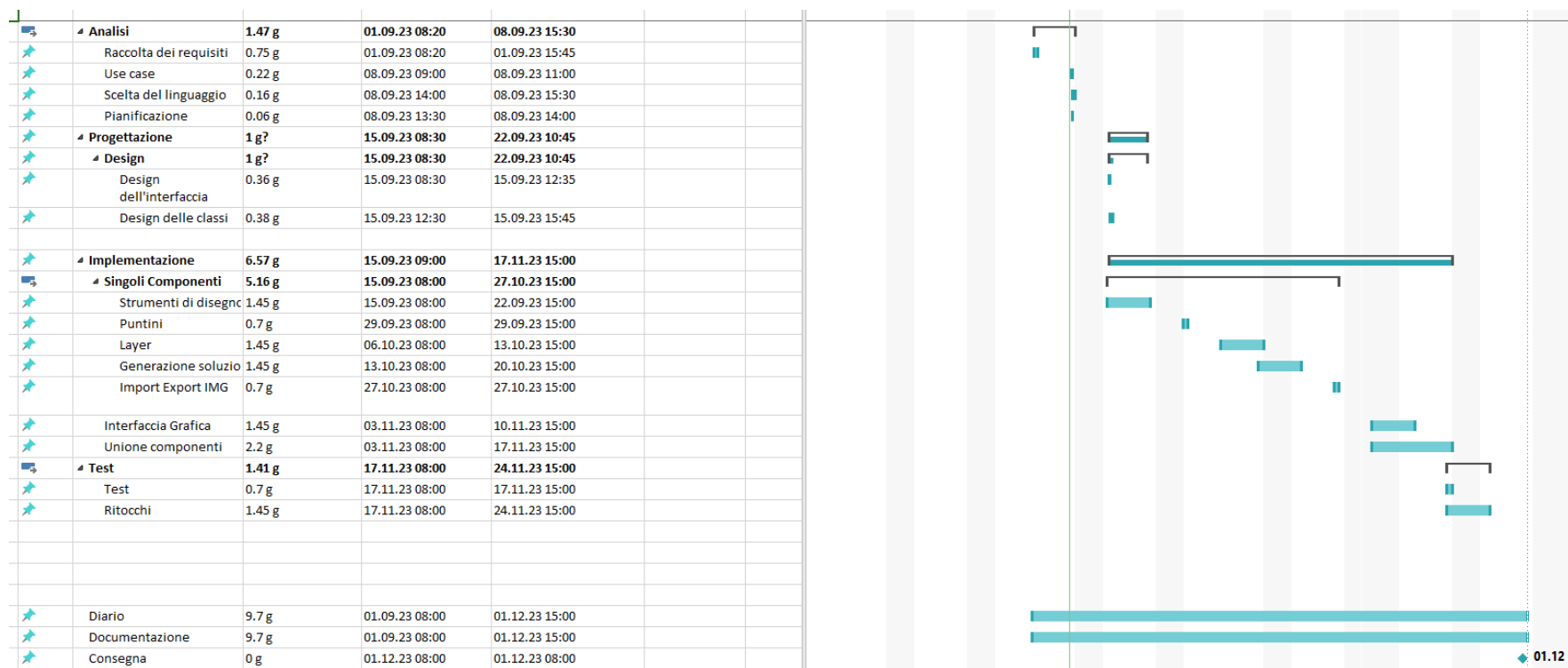


Figura 2 - Gant iniziale

1.7.1 Analisi dei mezzi

1.7.2 Software

Visual Studio Code, WebStorm, HTML 5, CSS 3, JavaScript, Bootstrap 5

1.7.3 Hardware

Computer fornito dalla scuola e un server FTP.

2 Progettazione

2.1 Struttura file JS

Il codice JavaScript lo ho strutturato nei seguenti file:

Nella cartella models ho inserito i file contenenti le classi e i file che permettono di fare azioni di esse.

Mentre fuori dalla cartella model ho messo i file imageImport e imageExport che permettono di importare l'immagine ed esportare i disegni fatti sul canvas.

Poi ho messo anche il main.js che dove controllo le interazioni con i checkbox, radio button, il colore, lo spessore, ... In questo file non ci sono metodi che disegnano o servono per disegnare.

Nel file drawHelp.js invece ho messo tutte quelle cose che servono di aiuto a più classi per disegnare. In questo file ci sono gli array che salvano i disegni.

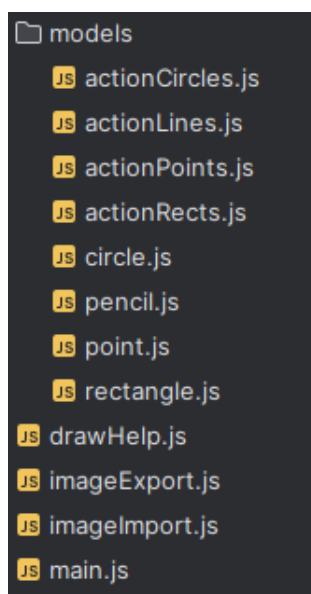


Figura 3 - Struttura file JS

3 Implementazione

3.1.1 Index

Lo stile della pagina HTML è stato interamente fatto con Bootstrap 5.
La pagina è tutta dentro un fluid container, che è diviso come mostrato nella seguente immagine.

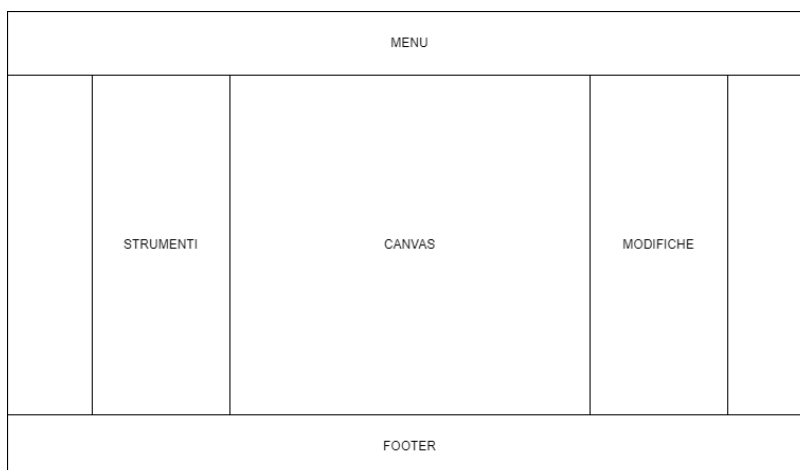


Figura 4 - Struttura HTML

3.1.1.1 Menu

[PIKA](#)

⬆️ LOAD

⬇️ SAVE

Il menu è composto dal nome dell'applicazione che è un semplice link a qui ho assegnato la seguente classe di bootstrap: *navbar-brand*.

Mentre il bottone LOAD è composto da un input di tipo file a qui ho assegnato un label:

```
<input type="file" accept=".jpg, .jpeg, .png, .webp" id='uploadImg'
  style="width:0px;visibility: hidden;">
<label id="label-for-file-selector" for="uploadImg" class="btn btn-outline-success me-4">
  <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
    class="bi bi-upload" viewBox="0 0 16 16">
  </svg>
  LOAD
</label>
```

Figura 5 - LOAD button

Il bottone SAVE invece è un semplice bottone:

```
<button type="button" class="btn btn-outline-success" onclick="openSave()">
  <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
    class="bi bi-download" viewBox="0 0 16 16">
  </svg>
  SAVE
</button>
```

Figura 6 - SAVE Button

NELLE SEGUENTI IMMAGINI NEI TAG SVG PER MOTIVI PRATICI È STATA RIMOSSA LA PATH.

Quando si preme il bottone di salvataggio verrà aperto un pannello che contiene le impostazioni di salvataggio. Questo verrà aperto sopra il canvas.

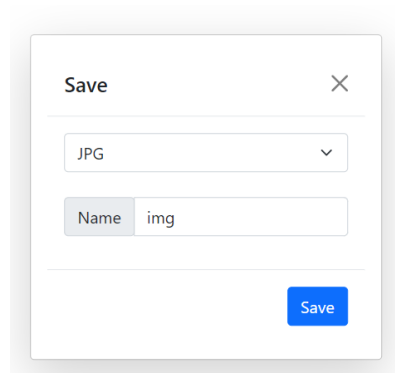


Figura 7 - Menu Save

3.1.1.2 Menu degli strumenti

Il menu che contiene gli strumenti contiene due card, una per gli strumenti e una per i layer.

La scelta degli strumenti avviene tramite degli input di tipo radio.

Mentre la scelta dei layer avviene tramite degli input checkbox.

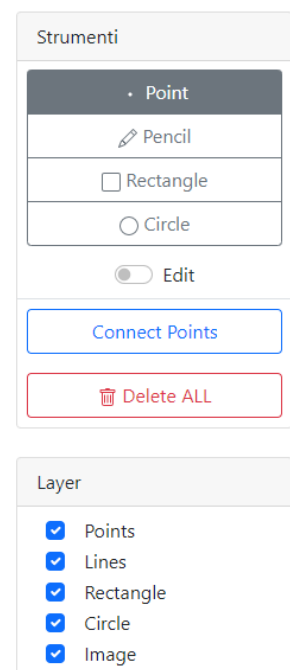


Figura 8 - Menu Strumenti

3.1.1.3 Canvas

Al caricamento della pagina, prima di aver importato un'immagine al posto del canvas verrà mostrato un messaggio d'informazione.

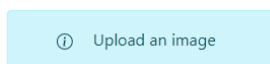


Figura 9 - Info upload

Una volta selezionata l'immagine questo verrà reso invisibile, e verrà reso visibile il canvas con l'immagine di sfondo.

Per far sì che i disegni non escano sgranati o deformati quando vengono disegnati sul canvas ho dovuto assegnare le seguenti proprietà CSS al canvas:

```
<canvas id="canvas" style=" left: 0px; right: 0px; max-height: 75vh;"  
class="border mw-100 z-0 invisible"></canvas>
```

Figura 10 - canvas

3.1.1.4 Menu di modifica

Il menu per modificare i disegni è anch'esso come il menu degli strumenti composto da un a card contenente una lista. Nella lista ho inserito il bottone delete, un input type color, un input type number e un input type range.

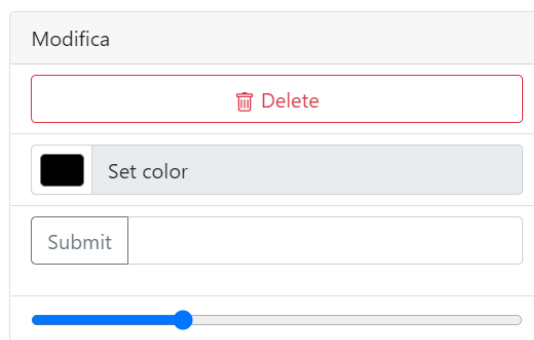


Figura 11 - Menu Modifica

3.2 Classi JS

I disegni sono rappresentati come classi.

Ogni disegno una volta fatto viene salvato nel suo array contenente tutti i disegni fatti dallo stesso strumento. Questi array si trovano nel file drawHelp.js.

```

26 // points array manipolato nel file point.js e actionPoints.js
27 var points : any[] = new Array(); // array con i puntini
28
29 // pencil array manipolato nel file pencil.js e actionLines.js
30 var lines : any[] = new Array();
31
32 // rectangle array manipolato nel file rectangle.js e actionRects.js
33 var rects : any[] = new Array();
34
35 // circle array manipolato nel file circle.js e actionCircle.js
36 var circle : any[] = new Array();

```

Figura 12 - Array disegni

3.2.1 Point

La classe Point rappresenta i puntini numerati.

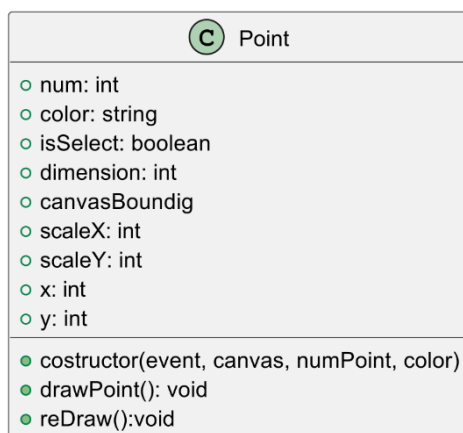


Figura 13 - Point

Per poter istanziare un Point il costruttore richiede un event per poterne ricavare le coordinate, il canvas per poter far il calcolo delle coordinate, il suo numero e il colore.

Una volta istanziato il costruttore richiamerà automaticamente la funzione drawPoint che permette di disegnare il punto.

Nel caso in cui il puntino è selezionato verrà disegnato con un'ombreggiatura rossa.

```
if (this.isSelect) {
    canvasDrawed.shadowBlur = 30;
    canvasDrawed.shadowColor = "red";
} else {
    canvasDrawed.shadowBlur = 0;
    canvasDrawed.shadowColor = null;
}
```

Figura 14 - selezione

La funzione reDraw si occupa semplicemente di richiamare la funzione drawPoint. Quest'ultima è stata fatta per far sì che ogni classe abbia una funzione reDraw (non necessaria).

3.2.1.1 Event

Per catturare l'evento che fa disegnare il puntino ho creato questa funzione anonima

```
66 // quando viene premuto il pulsante
67 canvas.addEventListener( type: "mousedown", listener: function ( e : MouseEvent ) : void {
68     // se e in modalita di disegno dei puntini
69     if (pointMode.checked && !editMode.checked) {
70         points.push(new Point(e, canvas, points.length, color));
71         document.getElementById( elementId: "newNumPoint").setAttribute( qualifiedName: "max", points.length);
72     }
73 });
74
```

Figura 15 – disegno puntino

Questa verrà chiamata al click del mouse.

Il Point verrà istanziato solo nel caso in cui l'utente avrà selezionato lo strumento Point e non sarà nella modalità di modifica. Se questa condizione si avvera allora aggiunto un nuovo punto nell'array 'points'.

3.2.2 Rectangle

La classe Rectangle rappresenta i rettangoli.


 Rectangle
<ul style="list-style-type: none"> ○ canvasBoundig ○ startX: int ○ startY: int ○ endX: int ○ endY: int ○ color: string ○ isSelect: boolean ○ dimension: int
<ul style="list-style-type: none"> ● costructor(event, color) ● move(event, canDraw): void ● end(): boolean ● reDraw(): void ● getX(): int ● getY(): int

Figura 16 - Rectangle

Per poter istanziare un rettangolo è necessario passargli un event e un color nel costruttore. L'event servirà per poter assegnare le coordinate startX e startY che definiscono il punto in cui parte il rettangolo. Mentre endX e endY servono per definire la larghezza e l'altezza del rettangolo.

Le funzioni getX e getY servono per poter calcolare le coordinate.

3.2.2.1 Event

Per poter iniziare a disegnare un rettangolo ho creato questa funzione anonima che se l'utente si trova in modalità rettangolo e con la modalità modifica disattivata aggiunge all'array rects una nuova istanza dei Rectangle.

```
110 // quando inizia a disegnare il rettangolo
111 canvas.addEventListener( type: "mousedown", listener: function (e : MouseEvent ) : void {
112     // se e in modalita disegno del rettangoli
113     if (rectangleMode.checked && !editMode.checked) {
114         rects.push(new Rectangle(e, canvas, color));
115     }
116 });
```

Figura 17 - pressione

Poi se l'utente tenendo premuto il mouse lo trascina verrà richiamata la seguente funzione anonima che prenderà l'ultima istanza nell'array e richiamerà la funzione move e continuerà a ridisegnare il rettangolo finché non molla il mouse.

```
118 // quando disegna il rettangolo
119 canvas.addEventListener( type: "mousemove", listener: function (e : MouseEvent ) : void {
120     // se puo disegnare
121     if (rectangleMode.checked && !editMode.checked && rects.length >= 1) {
122         rects[rects.length - 1].move(e, canDraw);
123         reDrawAll();
124     }
125 });
```

Figura 18 - spostare

A questo punto una volta aver rilasciato il mouse imposterà canDraw a false di modo che non possa più disegnare.

```
127 // quando finisce di disegnare il rettangolo
128 canvas.addEventListener( type: "mouseup", listener: function (e : MouseEvent ) : void {
129     if (rectangleMode.checked && !editMode.checked && rects.length >= 1) {
130         canDraw = rects[rects.length - 1].end();
131         reDrawAll();
132     }
133 });
```

Figura 19 - rilascio

3.2.3 Circle

La classe Circle serve per disegnare i cerchi.

Il funzionamento della classe Circle è uguale alla classe Rectangle.

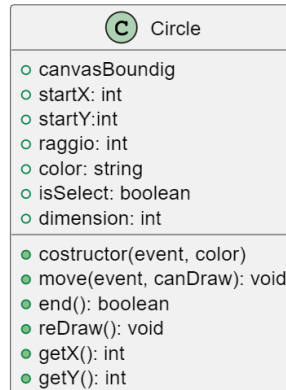


Figura 20 - Circle

L'unica differenza è che al posto di avere delle coordinate di inizio e di fine ha solo una coordinata che rappresenta il centro del cerchio e il suo raggio.

3.2.3.1 Event

Anche qui per disegnare un cerchio ho creato tre funzioni anonime.

La prima serve per quando inizia a disegnare il cerchio.

```

122 // quando viene premuto il mouse
123 canvas.addEventListener("mousedown", function (e) {
124     // se e in modalita per disegnare i cerchi
125     if (circleMode.checked && !edidtMode.checked) {
126         circle.push(new Circle(e, canvas, canvasDrawed, color));
127     }
128 });
  
```

Figura 21 - pressione

La seconda per quando muove il mouse per disegnarlo definendogli la lunghezza del raggio

```

131 // quando muove il mouse per disegnare il cerchio
132 canvas.addEventListener("mousemove", function (e) {
133     // se e in modalita per disegnare i cerchi
134     if (circleMode.checked && !edidtMode.checked && circle.length >= 1) {
135         circle[circle.length - 1].move(e, canDraw);
136         reDrawAll();
137     }
138 });
  
```

Figura 22 - spostare

Mentre la terza per quando finisce di disegnare il cerchio rilasciando il mouse.

```
140 // quando finisce di disegnare il cerchio
141 canvas.addEventListener("mouseup", function (e) {
142     if (circleMode.checked && !editMode.checked && circle.length >= 1) {
143         canDraw = circle[circle.length - 1].end();
144         redrawAll();
145     }
146 });
```

Figura 23 - rilascio

3.2.4 Pencil

La classe Pencil rappresenta le linee.

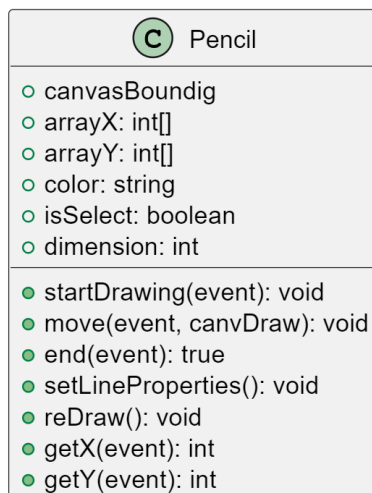


Figura 24 - Pencil

Una linea è composta da tante coordinate che rappresentano lo spostamento del mouse. Queste vengono salvate negli arrayX e arrayY e vengono aggiunte nella funzione move che viene chiamata nella seconda funzione anonima che viene richiamata allo spostamento del mouse. Ad ogni spostamento del mouse quest'ultima richiamerà move passandogli la posizione del mouse che poi verranno aggiunti agli array.

```
this.arrayX.push(this.getX(event));
this.arrayY.push(this.getY(event));
```

Figura 25 – array coordinate linea

La funzione redraw per ridisegnare la linea ripercorrerà gli array tracciando le linee da coordinata a coordinata.

3.2.4.1 Event

Per iniziare a disegnare la linea ho creato questa funzione anonima che verrà richiamata alla pressione del mouse.

```
126 // quando inizia a disegnare
127 canvas.addEventListener("mousedown", function (e) {
128     // se e in modalita per disegnare le linee
129     if (pencilMode.checked && !editMode.checked) {
130         lines.push(new Pencil(e, color));
131     }
132 });
```

Figura 26 - pressione

Quando inizia a disegnare nell'array lines verrà istanziata una nuova Pencil che rappresenterà la linea. Poi ho creato una seconda funzione anonima per quando trascina il mouse tracciando la linea.

```
134 // quando disegna la linea
135 canvas.addEventListener("mousemove", function (e) {
136     // se puo disegnare la linea
137     if (pencilMode.checked && !editMode.checked && lines.length >= 1) {
138         lines[lines.length - 1].move(e, canDraw);
139     }
140 });
```

Figura 27 - spostare

Questa richiamerà la funzione move.

E per finire di disegnare ho creato un'altra funzione anonima che viene chiamata al rilascio del mouse.

```
142 // quando finisce di disegnare la linea
143 canvas.addEventListener("mouseup", function (e) {
144     if (pencilMode.checked && !editMode.checked && lines.length >= 1) {
145         canDraw = lines[lines.length - 1].end(e);
146     }
147 });
```

Figura 28 - rilascio

3.3 Action

I file action sono i file che permettono di fare azioni sui disegni (oggetti).

3.3.1 ActionPoints

Questo file permette di fare azioni sui Point.

3.3.1.1 Selezione

Per Prima cosa per poter fare delle azioni su un puntino bisognerà selezionarne uno. Questo viene fatto tramite il doppio click con questa funzione anonima.

```
17 // per selezionare un puntino
18 canvas.addEventListener("dblclick", function (e) {
19
20     canvasBounding = canvas.getBoundingClientRect();
21     scaleX = canvas.width / canvasBounding.width;
22     scaleY = canvas.height / canvasBounding.height;
23
24     // se e in modalita modifica dei puntini
25     if (pointMode.checked && editMode.checked) {
26         let pointReturned = getPointClicked(e);
27         // se ritorna un puntino
28         if (pointReturned != null) {
29             // se era gia selezionato
30             if (pointReturned == pointSelected && isAPointSelected()) {
31                 deselectPoint();
32                 return;
33             }
34             selectThisPoint(pointReturned);
35         }
36     }
37 });
```

Figura 29 – selezione point

Per prima cosa aggiorno le variabili che permettono il calcolo delle coordinate.
Poi controllo se l'utente ha selezionato lo strumento disegno Point ed è nella modalità edit.
Poi se si trova nella modalità giusta controllo che il click sia avvenuto vicino ad un puntino tramite la funzione `getPointClicked(e)` che ritorna la posizione del puntino nell'array oppure null se non c'è nessun punto vicino.
Nel caso in cui il puntino ritornato sia già selezionato allora lo deselectiono, sennò lo seleziono.

3.3.1.2 Spostare

Per spostare un puntino ho dovuto creare queste tre funzioni anonime.
La prima controlla che ci sia un Point selezionato, nel caso in cui esiste allora potrà essere spostato.

```
39 // quando viene premuto il pulsante
40 canvas.addEventListener("mousedown", function (e) {
41     // se il puntino e selezionato allora si puo muovere
42     if (pointMode.checked && editMode.checked && isAPointSelected()) {
43         canMove = isSamePoint(e, pointSelected);
44     }
45 });
```

Figura 30 - pressione

Se il puntino può essere spostato allora al movimento del mouse le coordinate del puntino verranno aggiornate con la nuova posizione e poi ridisegno tutto il canvas.

```
48 // spostare il puntino
49 canvas.addEventListener("mousemove", function (e) {
50     // se si può muovere
51     if (pointMode.checked && editMode.checked && canMove && isAPointSelected()) {
52         points[pointSelected].x = Math.round((e.x - canvasBounding.left) * scaleX);
53         points[pointSelected].y = Math.round((e.y - canvasBounding.top) * scaleY);
54         redrawAll();
55     }
56 });
```

Figura 31- movimento

Al rilascio del mouse imposterò la variabile booleana canMove a false di modo da finire lo spostamento.

```
58 // fine movimento puntino
59 canvas.addEventListener("mouseup", function (e) {
60     if (pointMode.checked && editMode.checked) {
61         if (isAPointSelected()) {
62             canMove = false;
63         }
64     }
65 });
```

Figura 32 - rilascio

3.3.1.3 Rinominare

Per rinominare il puntino ho creato questa funzione che viene chiamata al click del bottone da parte all'input type number.

Questa funzione prenderà il nuovo numero del puntino, va nell'array points e prende il puntino selezionato, lo salva e poi lo elimina dopodiché reinserirà il puntino salvato lo reinserisce nella posizione desiderata -1.

In seguito cambierà tutti i numeri dei puntini assegnandogli la propria posizione nell'array points + 1.

```
let newPos = document.getElementById("newNumPoint").value;
let p = points[pointSelected];
deletePoint();
points.splice(newPos - 1, 0, p);
```

Figura 33 - sostituzione

```
for (let i = 0; i < points.length; i++) {
    points[i].num = i + 1;
}
```

Figura 34 - incremento

3.3.1.4 Colore

Per cambiare il colore prendo semplicemente il nuovo colore e lo cambio con quello vecchio del puntino selezionato. E poi ridisegno tutto.

```
172 function changeColorPoint() {
173     if (isAPointSelected() && pointMode.checked && editMode.checked) {
174         let color = document.getElementById("color").value;
175         points[pointSelected].color = color;
176         reDrawAll();
177     }
178 }
```

Figura 35 - Colore

3.3.1.5 Dimensione

Per cambiare la dimensione del puntino semplicemente prendo la nuova dimensione e la imposto al puntino selezionato e poi ridisegno tutto.

```
196 function changeDimensionPoint(){
197     if (isAPointSelected() && pointMode.checked && editMode.checked) {
198         points[pointSelected].dimension = dimensionRange.value;
199         reDrawAll();
200     }
201 }
202
```

Figura 36 - Dimensione

3.3.1.6 Delete

Per eliminare un puntino rimuovo il puntino selezionato dall'array points e poi rinomino tutti i puntini con la nuova posizione nell'array + 1.

Per far sì che non si possano assegnare numeri più alti ai puntini di quelli esistenti quando si rinomina devo infine cambiare il numero massimo possibile da assegnare.

```
137 function deletePoint() {
138     if (isAPointSelected() && pointMode.checked && editMode.checked) {
139         points.splice(points[pointSelected].num - 1, 1);
140         for (let i = 0; i < points.length; i++) {
141             points[i].num = i + 1;
142         }
143         document.getElementById("newNumPoint").setAttribute("max", points.length);
144         reDrawAll();
145     }
146 }
```

Figura 37 - Delete

3.3.2 ActionRects

Il codice in questo file permette di fare le azioni sui rettangoli.

3.3.2.1 Selezionare

Per selezionare i rettangoli ho creato una funzione anonima che viene richiamata con il doppio click.

```

18 canvas.addEventListener("dblclick", function (e) {
19 |
20     canvasBounding = canvas.getBoundingClientRect();
21     scaleX = canvas.width / canvasBounding.width;
22     scaleY = canvas.height / canvasBounding.height;
23
24     // se e in modalita di modifica dei rettangoli
25     if (rectangleMode.checked && editMode.checked) {
26
27         let x = Math.round((e.x - canvasBounding.left) * scaleX);
28         let y = Math.round((e.y - canvasBounding.top) * scaleY);
29
30         let rectReturned = isInARect(x, y);
31
32         // se e si puo selezionare un rettangolo
33         if (rectReturned >= 0) {
34             // se era gia selezionato
35             if (rectReturned == rectSelected && isARectSelect()) {
36                 deselectRect();
37                 return;
38             }
39             selectRect(rectReturned);
40         }
41     }
42 });

```

Figura 38 - Selezione

Prima di poter selezionare un rettangolo controllo che l'utente abbia selezionato lo strumento rettangolo e sia in edit mode.

Poi per selezionare un rettangolo prendo le coordinate del click e poi passo tutti i rettangoli e guardo se queste appartengono ad un rettangolo nell'array rects.

Se le coordinate appartengono ad un rettangolo allora ritorno la posizione del rettangolo nell'array sennò ritorno -1.

Poi nel caso in cui il seguente rettangolo è già selezionato lo deseleziono, sennò lo seleziono.

3.3.2.2 Spostare

Il principio per spostare un rettangolo è lo stesso di quello dei puntini. La differenza sta che l'utente per spostare il rettangolo non premerà nella coordinata (0,0) ma magari lo sposta dal centro del rettangolo quindi per prendere le nuove coordinate del punto (0,0) aggiungo al punto startX e startY la differenza di spostamento tra la posizione precedente e quella nuova.

```

75 // se si vuole spostare
76 canvas.addEventListener("mousemove", function (e) {
77     // se puo essere spostato
78     if (rectangleMode.checked && editMode.checked && canMove && isARectSelect()) {
79         // spostato il rettangolo
80         let difX = Math.round((e.x - canvasBounding.left) * scaleX) - oldXMouseRect;
81         let difY = Math.round((e.y - canvasBounding.top) * scaleY) - oldYMouseRect;
82         rects[rectSelected].startX += difX;
83         rects[rectSelected].startY += difY;
84         redrawAll();
85     }
86     oldXMouseRect = Math.round((e.x - canvasBounding.left) * scaleX);
87     oldYMouseRect = Math.round((e.y - canvasBounding.top) * scaleY);
88 });

```

Figura 39 - Spostare

3.3.2.3 Altre azioni

Il funzionamento delle altre azioni è presso che uguale alle azioni fatte sui puntini con la differenza che vengono fatte sull'array rects e non bisogna rinominare i rettangoli quando ne viene eliminato uno essendo che non hanno un numero.

3.3.3 actionCircle

In questo file c'è il codice che permette di fare delle azioni sui cerchi.

3.3.3.1 Selezionare

Per selezionare un cerchio ho creato una funzione anonima che per prima cosa guarda che l'utente si trova nella modalità corretta.

Poi prende le coordinate del doppio click e guarda se nell'array esiste un cerchio che contiene la seguente coordinata.

Per far questo quando guardo se la coordinata potrebbe appartenere al cerchio guardo se la distanza della coordinata dal centro è inferiore al raggio.

Poi nel caso esiste un cerchio ritorno la posizione di esso nell'array.

```

17 | // per selezionare un rettangolo
18 | canvas.addEventListener("dblclick", function (e) {
19 |
20 |     canvasBounding = canvas.getBoundingClientRect();
21 |     scaleX = canvas.width / canvasBounding.width;
22 |     scaleY = canvas.height / canvasBounding.height;
23 |
24 |     // se e in modalita di modifica dei rettangoli
25 |     if (rectangleMode.checked && editMode.checked) {
26 |
27 |         let x = Math.round((e.x - canvasBounding.left) * scaleX);
28 |         let y = Math.round((e.y - canvasBounding.top) * scaleY);
29 |
30 |         let rectReturned = isInARect(x, y);
31 |
32 |         // se e si puo selezionare un rettangolo
33 |         if (rectReturned >= 0) {
34 |             // se era gia selezionato
35 |             if (rectReturned == rectSelected && isARectSelect()) {
36 |                 deselectRect();
37 |                 return;
38 |             }
39 |             selectRect(rectReturned);
40 |         }
41 |     }
42 | });

```

Figura 40 - Selezione

3.3.3.2 Altre azioni

Il funzionamento delle altre azioni è uguale a quello dei rettangoli e dei puntini.

3.3.4 actionLines

Questo file contiene il codice per fare le azioni sulle linee.

3.3.4.1 Selezionare

Per selezionare una linea ho creato una funzione anonima che viene richiamata al doppio click. Poi controllo che l'utente si trovi nella modalità corretta (strumento Pencil e edit mode attivata).

Essendo la linea salvata come un insieme di puntini per selezionarle passo tutta la line e per quasi ogni puntino controllo se il click è avvenuto su oppure da parte ad esso.

Quasi perché essendo la linea molto densa di puntini per ottimizzare il tutto passo un puntino su due per far sì che ci siano meno puntini da confrontare. Questo avviene nella funzione isOnLine.

```
75 function isOnALine(x, y) {
76   for (var i = 0; i < lines.length; i++) {
77     for (var j = 0; j < lines[i].arrayX.length - 2; j+= 0) {
78       if (getDistancePointLine(lines[i].arrayX[j], lines[i].arrayY[j], x, y) < lines[i].dimension) {
79         return i;
80       }
81       j+= 2;
82     }
83   }
84   return -1;
85 }
```

Figura 42 - selezioni linee

```
16 // quando fine fatto doppio click per selezionare
17 canvas.addEventListener("dblclick", function (e) {
18
19   canvasBounding = canvas.getBoundingClientRect();
20   scaleX = canvas.width / canvasBounding.width;
21   scaleY = canvas.height / canvasBounding.height;
22
23   // se e in modalita di modiffica delle linee
24   if (pencilMode.checked && editMode.checked) {
25
26     let x = Math.round((e.x - canvasBounding.left) * scaleX);
27     let y = Math.round((e.y - canvasBounding.top) * scaleY);
28
29     // il click e avvenuto su una linea
30     let line = isOnALine(x, y);
31     if (line >= 0) {
32
33       // se era gia selezionata deselectionala
34       if (line == lineSelected) {
35         deselectLine();
36         return;
37       }
38
39       // seleziona la linea
40       selectLine(line);
41     }
42   }
43 });
```

Figura 41 - linee

3.3.4.2 Altre azioni

Essendo la linea uno strumento pesante per quanto riguarda le prestazioni non do la possibilità all'utente di poter spostare le linee una volta disegnate. Ma solo di poterle cambiare di spessore, colore e cancellarle.

Il funzionamento di quest'ultime azioni è presso che identico a quanto fatto con gli altri strumenti.

3.4 drawHelp.js

In questo file ho messo tutto quel codice che non si riferisce ad un singolo oggetto.

In cima al file ho dichiarato gli array in cui sono salvati i disegni.

```
26 // points array manipolato nel file point.js e actionPoints.js
27 var points = new Array(); // array con i puntini
28
29 // pencil array manipolato nel file pencil.js e actionLines.js
30 var lines = new Array();
31
32 // rectangle array manipolato nel file rectangle.js e actionRects.js
33 var rects = new Array();
34
35 // circle array manipolato nel file circle.js e actionCircle.js
36 var circle = new Array();
```

Figura 43 - array disegni

Poi ho inserito i seguenti metodi: `reDrawAll()`, `reDrawAllWhidoutClear()`. Questi metodi permettono di ridisegnare tutti gli oggetti salvati negli array.

Ho inserito anche la funzione `deselectAll()` che consente di deselectionare tutti gli oggetti. Questo è utile per esempio prima di esportare l'immagine.

Poi ho inserito anche le funzioni `connectDots()` e `reConnectDots()`. Ho deciso di inserirli in questo file e non in `actionPoints` perché questi non servono per fare azioni sui puntini ma disegnano una linea che li collega.

`connectDots()` viene richiamato solo dall'utente quando preme sul bottone apposito.

Connect Points

Mente `reConnectDots()` viene richiamato nel codice quando faccio un `reDrawAll()`.

3.5 main

Nel main.js ho inserito istanzio le variabili che mi permettono di controllare che opzioni l'utente ha selezionato.

```

9 | // strumenti
10 | let pointMode = document.getElementById('Point');
11 | let pencilMode = document.getElementById('Pencil');
12 | let rectangleMode = document.getElementById('Rectangle');
13 | let circleMode = document.getElementById('Circle');
14 |
15 | let editMode = document.getElementById('edit');
16 |
17 | // layer
18 | let pointLayer = document.getElementById('PointsLayer');
19 | let pencilLayer = document.getElementById('LinesLayer');
20 | let rectLayer = document.getElementById('RectLayer');
21 | let circleLayer = document.getElementById('CircleLayer');
22 | let imgLayer = document.getElementById('ImgLayer');
```

Figura 44 - radio & checkbox

Poi gestisco anche la selezione dei checkbox tramite per esempio la funzione linesLayerCheck() che controlla se il layer delle linee è visibile.

```

<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="LinesLayer" checked
    onclick="linesLayerCheck()">
  <label class="form-check-label ms-2" for="LinesLayer">
    Lines
  </label>
</div>
```

Figura 45 - layer

```

95 | // layer delle linee
96 | function linesLayerCheck() {
97 |   if (!pencilLayer.checked) {
98 |     pencilMode.disabled = true;
99 |   } else {
100 |     pencilMode.disabled = false;
101 |   }
102 |
103 |   if (pencilMode.checked) {
104 |     pointMode.checked = true;
105 |   }
106 |   reDrawAll();
107 | }
```

Figura 46 - layer

Il codice è presso che identico per ogni layer di ogni strumento.

3.6 imageImport

In questo file c'è il codice che permette di importare l'immagine.

Tramite la funzione loadImage() quando verrà importata l'immagine verrà chiamata e farà il render dell'immagine.

Prima di caricarla nel canvas faccio un controllo che l'immagine sia abbastanza grande (per far sì che non si possa importare immagine troppo piccole), e poi la carico.

Dopo averla caricata setto le variabili per il calcolo delle coordinate e dichiaro gli array che conterranno i disegni.

```
reader.onloadend = function (e) {

    var image = new Image();
    image.src = e.target.result;
    image.onload = function () {

        // se le dimensioni del immagine sono accettabili
        if(this.width > 250 && this.height > 150){

            canvas.style.backgroundImage = 'url("' + this.src + '")';
            ImgSrc = canvas.style.backgroundImage;
            canvas.style.backgroundSize = 'contain';

            canvas.width = this.width;
            canvas.height = this.height;

            // setto le variabili per il calcolo delle coordinate del file drawHelp.js
            canvasBounding = canvas.getBoundingClientRect();
            scaleX = canvas.width / canvasBounding.width;
            scaleY = canvas.height / canvasBounding.height;

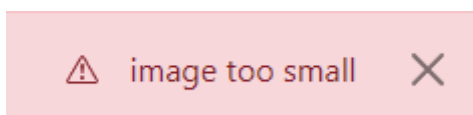
            // dichiaro gli array del file drawHelp.js
            points = new Array();
            lines = new Array();
            rects = new Array();
            circle = new Array();

            canvas.classList.remove("invisible");
            document.getElementById("allertUpload").classList.remove("visible");
            document.getElementById("allertUpload").classList.add("invisible");
            canvas.classList.add("visible");

        }else{
            // se l'immagine non va bene
            openLoadError();
        }
    }
}
```

Figura 47 - import

Se l'immagine sia troppo piccola verrà mostrato a schermo il seguente messaggio.



3.7 ImageExport

Nel seguente file ho messo il codice per poter esportare l'immagine.

Prima di esportare l'immagine verrà mostrato a schermo il seguente menu per settare le impostazioni di salvataggio.

Questo viene gestito dalla funzione openSave() che apre il pannello.

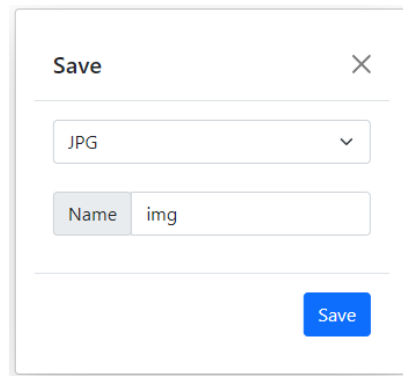


Figura 48 - menu save

Quando l'utente premerà Save verrà chiamato il metodo imageExport() che preparerà il download con il nome e il formato scelto.

Poi per far sì che lo sfondo sia bianco coloro il canvas di bianco e poi ridisegno tutto senza prima pulire il canvas.

E infine la scarico. E per tornare allo stato iniziale ridisegno tutto pulendo prima il canvas per rimuovere lo sfondo bianco.

```

17 function imageExport(){
18     let link = document.createElement('a');
19     let name = document.getElementById("nameFile").value;
20     let format = document.getElementById("formats").value;
21
22     link.download = name + '.' + format;
23
24     canvasDrawed.fillStyle = "#FFFFFF";
25     canvasDrawed.fillRect(0, 0, canvas.width, canvas.height);
26     deselectAll();
27     redrawAllWhidoutClear();
28     link.href = document.getElementById('canvas').toDataURL()
29     link.click();
30     redrawAll();
31     closeSave();
32     saveSuccess();
33 }

```

Figura 49 - export

4 Test

4.1 Protocollo di test

Test Case:	TC-001	Nome:	Manipolazione puntini
Riferimento:	REQ-01		
Descrizione:	Disegnare, spostare, cambiare colore, ingrandire, cambiare numero, eliminare dei puntini		
Prerequisiti:	Aver importato in immagine		
Procedura:	<ol style="list-style-type: none"> 1. Creazione di un puntino 2. Selezionare il puntino 3. Ingrandirlo 4. Cambiargli il colore 5. Cambiargli il numero 6. Spostarlo 7. Eliminarlo 		
Risultati attesi:	Il puntino viene spostato, selezionato, ingrandito, viene cambiato il numero e infine eliminato.		

Test Case:	TC-002	Nome:	Strumenti di disegno
Riferimento:	REQ-02		
Descrizione:	Disegnare con ogni strumento		
Prerequisiti:	Aver importato in immagine		
Procedura:	<ol style="list-style-type: none"> 1. Disegnare con la linea Eseguire tutte le operazioni possibili 2. Disegnare con il cerchio Eseguire tutte le operazioni possibili 3. Disegnare con il rettangolo Eseguire tutte le operazioni possibili 4. Eliminare tutti i disegni 		
Risultati attesi:	Tutti gli strumenti vengono disegnati e modificato per quanto concesso e poi vengono eliminati.		

Test Case:	TC-003	Nome:	Esportare l'immagine
Riferimento:	REQ-03		
Descrizione:	Esportazione dell'immagine		
Prerequisiti:	Aver importato in immagine		
Procedura:	<ol style="list-style-type: none"> 1. Importare l'immagine 2. Fare un disegno 3. Esportarla con tutti i formati disponibili 		
Risultati attesi:	L'immagine viene esportata nel formato desiderato con sfondo bianco e sopra i disegni.		

Test Case:	TC-004	Nome:	Soluzione
Riferimento:	REQ-04		
Descrizione:	Deve poter essere visibile la soluzione		
Prerequisiti:	Aver importato in immagine		
Procedura:	<ol style="list-style-type: none"> 1. Creare un disegno 2. Collegare i puntini tramite l'apposito bottone 3. Rimuovere il collegamento 		
Risultati attesi:	I puntini disegnati vengono collegati da una linea retta in ordine numerico e poi ripremendo il bottone la soluzione viene cancellata.		

Test Case:	TC-005	Nome:	Layer
Riferimento:	REQ-05		
Descrizione:	Deve poter essere visibile la soluzione		
Prerequisiti:	Aver importato in immagine		
Procedura:	<ol style="list-style-type: none"> 1. Creare un disegno con tutti gli strumenti 2. Fare tutte le combinazioni possibili con i layer 		
Risultati attesi:	Durante le combinazioni vengono mostrati solo i layer selezionati.		

Test Case:	TC-006	Nome:	Delete All
Riferimento:			
Descrizione:	Eliminare tutti i disegni in un solo colpo.		
Prerequisiti:	Aver importato in immagine		
Procedura:	<ol style="list-style-type: none"> 1. Creare un disegno con tutti gli strumenti 2. Eliminare tutto tramite Delete All 		
Risultati attesi:	Richiesta di conferma, eliminazione di tutti i disegni.		

Test Case:	TC-007	Nome:	Importare con più formati
Riferimento:	REQ-07		
Descrizione:	Poter importare immagini con più formati e dimensioni.		
Prerequisiti:			
Procedura:	1. Importare immagini con i seguenti formati: jpg, jpeg, png, webp		
Risultati attesi:	L'immagine dovrà essere importata con successo.		

Test Case:	TC-008	Nome:	Immagine importata troppo piccola.
Riferimento:			
Descrizione:	Se l'immagine importata è troppo piccola verrà mostrato un messaggio di errore.		
Prerequisiti:			
Procedura:	1. Importare un'immagine con dimensioni inferiori a 250x150		
Risultati attesi:	Verrà mostrato un messaggio di errore.		

4.2 Risultati test

Test Case	Nome	Risultato	Problema
TC-001	Manipolazione puntini	1	
TC-002	Strumenti di disegno	0.8	Strumento secchiello non fatto
TC-003	Esportare l'immagine	1	
TC-004	Soluzione	1	
TC-005	Layer	1	
TC-006	Delete All	1	
TC-007	Importare con più formati	1	
TC-008	Immagine importata troppo piccola.	1	

4.3 Mancanze/limitazioni conosciute

4.3.1 Strumento secchiello

Durante lo sviluppo dello strumento secchiello con l'algoritmo da me sviluppato siccome usavo la ricorsione il programma usciva dallo stack. Poi per mancanza di tempo sono dovuto andare avanti.

```

Uncaught RangeError: Maximum call stack size exceeded
    at Bucket.fill (bucket.js:16:9)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)
    at Bucket.fill (bucket.js:31:30)

```

Figura 50 - error stack

5 Consuntivo

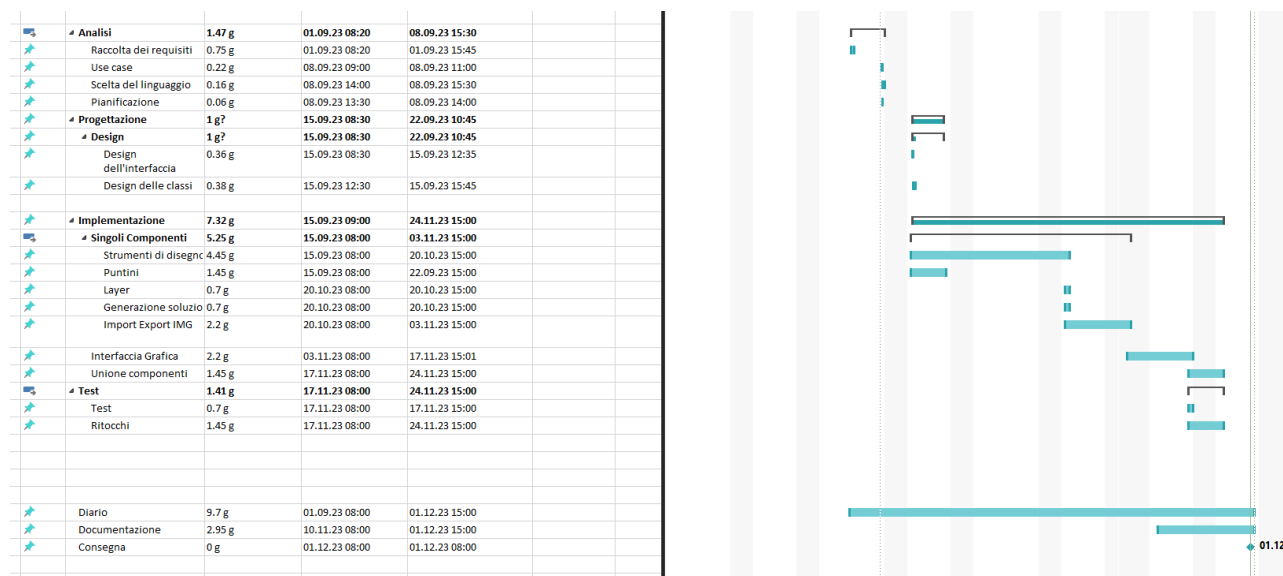


Figura 51 - Gant finale

Nella realizzazione degli strumenti ho impiegato molto molto più tempo del previsto. E a causa di un venerdì malato sono rimasto indietro con l'implementazione del codice con l'interfaccia grafica.

6 Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un'aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

6.1 Sviluppi futuri

In futuro si potrebbe ricreare lo strumento secchiello. E ottimizzare maggiormente le linee per permettere di spostarle.

Rendere completamente responsive l'interfaccia grafica.

6.2 Considerazioni personali

Ho imparato a gestire un progetto di dimensioni maggiori rispetto agli altri esercizi fatti in precedenza.

Le basi di git e GitHub.

Ad utilizzare meglio il canvas e Bootstrap.

Ho imparato l'importanza delle documentazioni e di non farla solo alla fine ma farla man mano col lo sviluppo del progetto.

7 Glossario

Termine	Descrizione
Array	Elemento che contiene un insieme di variabili.
Bootstrap	Libreria per il CSS.
Canvas	Elemento JavaScript che permette di disegnare facilmente.
CSS	Cascading Style Sheets: linguaggio che permette di definire il layout e la grafica di una pagina web.
Funzione / metodo	Blocco di codice che può avere parametri in entrata e può ritornare qual cosa.

8 Allegati

Elenco degli allegati, esempio:

- Gant
- Use Case

9 Immagini

Figura 1 - Use Case.....	7
Figura 2 - Gant iniziale.....	1
Figura 3 - Struttura file JS.....	1
Figura 4 - Struttura HTML.....	2
Figura 5 - LOAD button.....	2
Figura 6 - SAVE Button	3
Figura 7 - Menu Save	3
Figura 8 - Menu Strumenti	3
Figura 9 - Info upload.....	4
Figura 10 - canvas.....	4
Figura 11 - Menu Modifica	4
Figura 12 - Array disegni	5
Figura 13 - Point	5
Figura 14 - selezione	6
Figura 15 – disegno puntino	6
Figura 16 - Rectangle	6
Figura 17 - pressione.....	7
Figura 18 - spostare.....	7
Figura 19 - rilascio	7
Figura 20 - Circle	8
Figura 21 - pressione.....	8
Figura 22 - spostare.....	8
Figura 23 - rilascio	9
Figura 24 - Pencil.....	9
Figura 25 – array coordinate linea	9
Figura 26 - pressione.....	10
Figura 27 - spostare.....	10
Figura 28 - rilascio	10
Figura 29 – selezione point	11
Figura 30 - pressione.....	11
Figura 31- movimento.....	12
Figura 32 - rilascio	12
Figura 33 - sostituzione	12
Figura 34 - incremento.....	12
Figura 35 - Colore	13
Figura 36 - Dimensione	13
Figura 37 - Delete	13
Figura 38 - Selezione.....	14
Figura 39 - Spostare	15
Figura 40 - Selezione.....	16
Figura 41 - linee.....	17
Figura 42 - selezioni linee.....	17
Figura 43 - array disegni.....	18
Figura 44 - radio & checkbox.....	19
Figura 45 - layer.....	19
Figura 46 - layer.....	19
Figura 47 - import	20
Figura 48 - menu save.....	21
Figura 49 - export	21
Figura 50 - error stack	25
Figura 51 - Gant finale	25