

**Università degli Studi di Bergamo**

**Dipartimento di Ingegneria Gestionale, dell’Informazione e della trasmissione**

**VolText**

**Manuale utente**

**Colombi Simone**

**Gambarara Alberto**

**Scarpellini Stefano**

**Indice**

1. **Introduzione**
2. **Grammatica**
   1. **Descrizione generale**
   2. **Elementi di una pagina**

**2.2.1. Immagini**

**2.2.2. Testi**

**2.2.3. Liste**

**2.2.4. Div**

* 1. **Errori della grammatica**

1. **Tool utilizzati**
2. **Interfaccia utente**
3. **Possibili sviluppi futuri**
4. **Introduzione**

VolText è un progetto sviluppato nell’ambito del corso di Linguaggi Formali e Compilatori, tenuto dal professor Giuseppe Psaila nell’Anno Accademico 2020/2021.

L’obiettivo è la progettazione e implementazione di una libreria in Java per la generazione di un documento PDF e di una interfaccia grafica che permetta l’utilizzo semplificato di essa. La parte più inerente al corso è stata la realizzazione di una grammatica che descrive la struttura di un documento PDF usando i costrutti tipici della sintassi HTML, quali div, immagini, testi e liste, ma sfruttando una sintassi simile a quella di JSON.

La libreria è incentrata in particolare sullo sviluppo di documenti che abbiano una grafica semplice come volantini, piccoli annunci, avvisi, biglietti da visita, etc.

Il tool non vuole e non funge da sostituto di LaTeX ma prende gli elementi essenziali, creando una versione diversa sia in termini di linguaggio sia in termini di componenti, eliminando formule, grafici, tabelle, etc. non utili allo sviluppo di documenti grafici.

1. **Grammatica**
   1. **Descrizione generale**

La grammatica è stata progettata e implementata pensando a chi lavora con linguaggi web e che è familiare ad HTML.

Sfruttando una tipica sintassi del linguaggio JSON per la costruzione di tag personalizzati si è strutturata la grammatica come un documento PDF composto da vari elementi che possono essere disegnati all’interno in posizioni e dimensioni a piacimento.

Si riporta la grammatica e, per ogni regola, se ne descrive la funzione.

**grammar** VolText;

@header {

**package** antlr;

}

// match keyword hello followed by an identifier

pdf : *A* pdfattr\* (stylesheet)? (page)+ *C*;

pdfattr: 'title:' *STRING* *ENDNLINE*

| 'author:' *STRING* *ENDNLINE*

| 'path:' *STRING* *ENDNLINE*;

stylesheet: 'stylesheet' *O* element\* *C*;

element: '@' *STRING* *O* attrStyle\* *C*;

attrStyle: 'cross-point:' *NVAL* *ENDNLINE*

| 'shape:"' ('RECTANGLE' | 'CIRCLE' | 'TRIANGLE') *ENDLINE*

| ('fit-x' | 'fit-y') ':' *TFVAL* *ENDNLINE*

| ('pos-x' | 'pos-y') ':' *NOTVAL*? *NVAL* (*UNIT*)? *ENDNLINE*

| 'angle-rotation' ':' *NOTVAL*? *NVAL* *ENDNLINE*

| ('height' | 'width') ':' *NVAL* (*UNIT*)? *ENDNLINE*

| ('p\_height' | 'p\_width') ':' *NVAL* *ENDNLINE*

| 'ordered:' *TFVAL* *ENDNLINE*

| 'bullet:' *STRING* *ENDNLINE*

| ('font-family:' | 'font-family-ttf:' | 'font-family-otf:') *STRING* *ENDNLINE*

| 'font-size:' *NVAL* *ENDNLINE*

| ('bold:' | 'italics:' |'underline:') *TFVAL* *ENDNLINE*

| 'colorT-bullet:' *STRING* *ENDNLINE*

| 'color-bullet:' *COLORVAL* *ENDNLINE*

| 'colorT:' *STRING* *ENDNLINE*

| 'color:' *COLORVAL* *ENDNLINE*

| 'position:' *POSVAL* *ENDNLINE*

| 'alignment:' *ALIGNVAL* *ENDLINE*

| 'orientation:' *ORIENTATION* *ENDNLINE*

| 'oob:' *TFVAL* *ENDNLINE*

| 'format:' *FORMATVAL* *ENDNLINE*;

page: 'page' *O* pageattr\* pae\* *C*;

pae: elemd | div;

div: 'div' *O* (color | idval | imganumber | elemd | positionv | fitAttr | figure | tvalue)\* *C*;

tvalue: 'cross-point:' *NVAL* *ENDNLINE*;

figure: 'shape:"' ('RECTANGLE' | 'CIRCLE' | 'TRIANGLE') *ENDLINE*;

elemd: text

| list

| img;

img: 'img' *O* imgattr\* imgElem imgattr\* *C*;

imgattr: (idval | imganumber | positionv | fitAttr);

imgElem: 'URL:' *STRING* *ENDNLINE*;

list: 'list' *O* (listattr | listElem)\* *C*;

listElem: 'item:' *STRING* *ENDNLINE*;

text: 'text' *O* txtattr\* txtElem\* txtattr\* txtElem\* txtattr\* *C*;

txtattr: (color | idval | imganumber | positionv | alignment | txtval | fitAttr);

txtElem: 'string:' *STRING* *ENDNLINE*;

//ATTRIBUTES

fitAttr: ('fit-x'

| 'fit-y') ':' *TFVAL* *ENDNLINE*;

imganumber: ('pos-x'

| 'pos-y' ) ':' *NOTVAL*? *NVAL* (*UNIT*)? *ENDNLINE*

| 'angle-rotation' ':' *NOTVAL*? *NVAL* *ENDNLINE*

| ('height'

| 'width') ':' *NVAL* (*UNIT*)? *ENDNLINE*;

*NOTVAL*: '-';

*UNIT*: 'mm' | '%' | 'pt';

idval: 'id:' *STRING* *ENDNLINE*;

listattr: 'ordered:' *TFVAL* *ENDNLINE*

| 'bullet:' *STRING* *ENDNLINE*

| idval

| imganumber

| txtval

| positionv

| color

| colorBullet

| fitAttr;

txtval: ('font-family:'

| 'font-family-ttf:'

| 'font-family-otf:') *STRING* *ENDNLINE*

| 'font-size:' *NVAL* *ENDNLINE*

| ('bold:'

| 'italics:'

| 'underline:') *TFVAL* *ENDNLINE*;

colorBullet:'colorT-bullet:' *STRING* *ENDNLINE*

| 'color-bullet:' *COLORVAL* *ENDNLINE*;

color: 'colorT:' *STRING* *ENDNLINE*

| 'color:' *COLORVAL* *ENDNLINE*;

positionv: 'position:' *POSVAL* *ENDNLINE*;

alignment: 'alignment:' *ALIGNVAL* *ENDLINE*;

pageattr: 'orientation:' *ORIENTATION* *ENDNLINE*

| 'oob:' *TFVAL* *ENDNLINE*

| 'p\_width' ':' *NVAL* *ENDNLINE*

| 'p\_height' ':' *NVAL* *ENDNLINE*

| 'format:' *FORMATVAL* *ENDNLINE*

| idval;

*FORMATVAL*: 'A0'

| 'A1'

| 'A2'

| 'A3'

| 'A4'

| 'A5'

| 'A6';

*ORIENTATION*:('hor' | 'ver');

//TERMINALI

*TXTATF*: 'bold'

| 'italics'

| 'underline';

*COLORVAL*: '#' ([0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F]);

*ALIGNVAL*: 'left'

| 'center'

| 'right'

| 'justify';

*TFVAL*: 'true'

| 'false';

*POSVAL*: 'lt'

| 'ct'

| 'rt'

| 'lc'

| 'cc'

| 'rc'

| 'lb'

| 'cb'

| 'rb' ;

*NVAL*: [0-9]+('.'[0-9]+)?;

*ENDLINE*: '";';

*ENDNLINE*: ';';

*O* : ':{';

*A* : '{';

*C* : '}';

*STRING*: '"' (~[\t\r\n])+ '"';

*WS* : [ \t\r\n]+ **->** skip ; // skip spaces, tabs, newlines

La prima regola, *pdf*, permette l’apertura del documento, l’inserimento di attributi quali titolo, autore e percorso di salvataggio del pdf, relativi al documento, la possibilità di inserire una parte di stile del documento (non implementata ma disponibile per futuri sviluppi), una o più pagine e la chiusura del documento.

Una struttura sommaria del tag *stylesheet* è stata realizzata come

stylesheet:{

@IdElemento1:{

<< attributi relativi a quell’elemento, quali *imganumber*, *positionv* e *txtval*>>

}

}

La pagina può contenere vari attributi quali orientamento, oob (out-of-bound), larghezza, altezza e formato, come specificato nella seguente tabella:

|  |  |
| --- | --- |
| **attributo** | **valori possibili** |
| orientation | hor, ver |
| oob | true,false |
| width | numero decimale |
| height | numero decimale |
| format | A0, A1, A2, A3, A4, A5, A6 |

Ogni pagina contiene al suo interno o dei div, quindi dei blocchi che possono contenere a loro volta altri elementi, oppure elementi stessi quali immagini, liste e testi. Questi elementi e i loro attributi verranno trattati nel capitolo 3.2.

Una pagina, come anche segnalato nella specifica ISO 32000-1, sezione 11.4.7, deve avere colore di fondo bianco. Quindi se si vuole colorare completamente una pagina è possibile inserire un div con dimensioni fittate rispetto alla pagina e inserirci altri elementi.

* 1. **Elementi di una pagina**

Trattiamo ora i vari elementi che possono comporre una pagina. Tratteremo per prime le immagini, i testi e le liste che possono essere inserite riferendosi alle dimensioni della pagina. Per ultimi tratteremo i div, elementi più complessi e che possono contenere al loro interno gli altri tipi di elemento ma con riferimenti di misura legati al div nel quale sono incapsulati. Ogni attributo di ogni elemento deve sempre essere chiuso da un ; mentre la chiusura degli elementi è realizzata tramite una graffa chiusa.

**2.2.1. Immagini**

Una immagine deve essere inserita attraverso il tag *“img:{ <<attributi>> }”*.

Essa può contenere attributi messi in posizione anche successiva all’unico attributo obbligatorio, ovvero l’URL. L’*url* è una stringa che specifica il percorso relativo o assoluto di quella immagine all’interno del computer. Se relativo, il riferimento partirà dalla cartella dove si trova il documento testuale e, come vedremo in seguito, se non viene aperto un documento ma solo incollato il testo all’interno della GUI avremo degli errori riguardo la mancanza dell’oggetto all'indirizzo specificato.

Si elencano ora i vari attributi disponibili per le immagini:

* *id*, ovvero l’identificativo dell’elemento all’interno del documento. Questo è stato pensato per l’uso soprattutto nel foglio di stile e per la segnalazione degli errori relativi al posizionamento e dimensionamento degli oggetti.
* *pos-x* e *pos-y*, indicano la posizione dell’elemento sull’asse orizzontale e verticale. Il valore può essere positivo o negativo e contenere anche l’unità di misura (millimetri (mm), percentuale (%) o punti (pt), di default mm).
* *angle-rotation*, indica la rotazione dell’elemento rispetto al suo centro in gradi. Se il valore specificato è positivo verrà effettuata una rotazione in senso antiorario, al contrario con un valore negativo verrà svolta una rotazione in senso orario.
* *height* e *width*, indicano l’altezza e la larghezza dell’immagine nel caso serva ridimensionare l’elemento all’interno della pagina o del div. Anch’esse possono essere espresse in millimetri, percentuale o punti.
* *position*, contiene una coppia di lettere che permettono di inserire l’elemento nel suo contenitore (pagina o div) in una posizione prefissata (immaginando una suddivisione in 9 quadrati del contenitore). Essi possono quindi essere *l,c* e *r* per segnalare se l’elemento deve stare rispettivamente a sinistra, centrato o a destra nel contenitore, e *t,c* o *b* per segnalare se l’elemento deve essere in alto (top, *t*), centrale (*c*) o in basso (bottom, *b*).
* *fit-x* e *fit-y*, contengono un valore booleano (true, false) per segnalare se l’elemento deve essere ridimensionato sull’intera dimensione del contenitore (sia esso la pagina o il div) rispetto all’asse x o y.

**2.2.2. Testi**

Un testo può essere inserito tramite il tag “*text:{ <<attributi>> }”*.

Esso può contenere attributi misti a stringhe di testo senza vincoli di precedenza.

Le stringhe/capitoli sono contenuti in attributi *“string”* e vengono scritti nel PDF nell’ordine che si è usato nella scrittura dell’elemento text.

Gli attributi dell’elemento *text* sono i seguenti:

* *id*, come già descritto per le immagini, identifica l’oggetto.
* *colorT* e *color*, attributi che segnalano il colore del testo (univoco per tutti le stringhe). *color* specifica un colore in formato *#RGBA* esadecimale, dove R è il colore rosso, G il colore verde, B il colore blu e A l’alpha, ovvero il valore di trasparenza dell’elemento. Nei testi e nelle liste l’alpha non è considerato, al contrario di ciò che accade per i div. *colorT* specifica un colore sotto forma di testo (black, blue, red, green, …). L’ultimo attributo che viene dichiarato verrà usato nella redazione del documento.
* *pos-x, pos-y, angle-rotation, height, width, position, fit-x* e *fit-y*: si veda la trattazione nel capitolo 2.2.1 sulle immagini.
* *alignment*, utilizzato per l’allineamento del testo (left per sinistra, center per centrato, right per destra e justify per giustificato)
* *font-family, font-family-ttf* e *font-family-otf*, specificano il font da utilizzare nella redazione del documento. Se si utilizza font-family si possono utilizzare i font standard quali helvetica, courier e times. Se si utilizza font-family-ttf, allora si può specificare il path dove è disponibile il file con estensione ttf (True Type Font) che descrive il font da usare. Se la cartella dove risiede il font contiene anche il relativo font in grassetto, corsivo e/o grassetto corsivo allora il path da specificare sarà quello del font regolare (terminante con Regular.ttf, e in cui gli altri file termineranno con Bold.ttf, Italic.ttf e BoldItalic.ttf, rispettivamente per grassetto, corsivo e grassetto corsivo): in questo caso verranno elaborati correttamente anche eventuali testi o parti di testo in grassetto (scritte come *\bold testo in bold \bold*), corsivo (scritte come *\italic testo in corsivo \italic*) o grassetto corsivo (scritte come *\bold\italic testo in bold italic \italic \bold*). Le stesse regole valgono per l’attributo font-family-otf, a cui si applicano font con estensione otf (Open Type Font)
* *font-size*, contiene la dimensione del carattere da usare nei testi.
* *bold, italic* e *underline*, sono attributi booleani che permettono di specificare se l’intero testo debba essere scritto in corsivo, in italico o sottolineato (quest’ultima specifica è stata implementata ma non conclusa per difficoltà di integrazione della libreria PDFBox-layout con PDFBox).

**2.2.3. Liste**

Una lista può essere inserita attraverso il tag *list:{ <<attributi>> }*

Essa può contenere attributi misti a stringhe di testo (voci della lista, *item*) senza vincoli di precedenza.

Gli attributi di una lista possono essere:

* *id*, come i precedenti elementi
* *ordered*, che serve per segnalare se è una lista ordinata di elementi (lista numerica) oppure se è una lista non ordinata (lista a punti)
* *bullet*, specifica il tipo di “punto” che fa da inizio dell’elemento della lista. I possibili punti sono *odd* (pallino pieno, ), *even* (doppia freccia, ), oppure *new(Bullet)* (in cui il *Bullet* è una stringa decisa dall’utente e usabile come punto).
* tutti gli attributi relativi ai testi, tranne l’allineamento, sono implementati anche per le liste
* *colorT-bullet* e *color-bullet* , attributi che segnalano il colore del bullet (univoco per tutti i bullet). *color-bullet* specifica un colore in formato *#RGBA* esadecimale, dove R è il colore rosso, G il colore verde, B il colore blu e A l’alpha, ovvero il valore di trasparenza dell’elemento. Il valore di alpha non è considerato. *colorT-bullet* specifica un colore sotto forma di testo (black, blue, red, green, …). L’ultimo attributo che viene dichiarato verrà usato nella redazione del documento.

**2.2.4. Div**

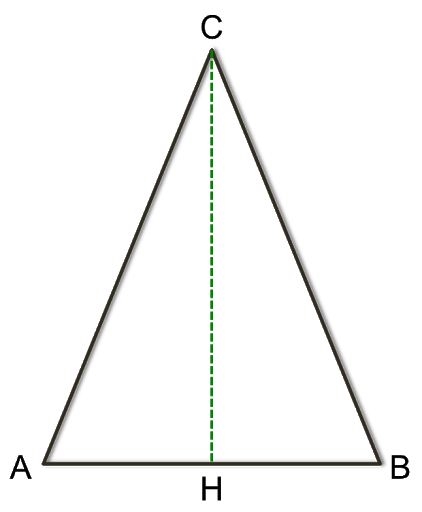
Un div può essere inserito attraverso il tag *div:{ <<attributi e/o elementi>>}*.

Il div, come descritto anche in html, è un contenitore che permette di raggruppare elementi al suo interno. In questa versione del progetto un div può contenere solo immagini, testi e liste ma non altri contenitori. Il contenitore può avere personalizzato attraverso vari attributi e gli elementi che sono al suo interno prendono come riferimento dimensionale la dimensione del contenitore.

Esso può contenere attributi misti a elementi (*img*, *text* e *list*) senza vincoli di precedenza.

Si analizzano ora i vari attributi di un contenitore:

* *id,* come ogni elemento, specifica un identificativo che permette di riconoscere l’elemento.
* *color* e *colorT*, come già presentato precedentemente per quanto riguarda i testi e le liste, specifica il colore di riempimento del contenitore. Non è stata realizzata una suddivisione in colore di riempimento e colore del bordo, che potrà essere inserita in futuri sviluppi. *color* specifica il colore in formato *#RGBA* e in questo caso l’alpha (A) specifica la trasparenza del contenitore, per fornire la possibilità di vedere altri elementi sottostanti ad esso. *colorT* invece permette l’uso di colori attraverso l’uso di nomi di colori.
* *pos-x*, *pos-y*, *angle-rotation*, *height*, *width*, *fit-x* e *fit-y*, come già detto nelle precedenti sezioni, specificano la posizione, l’angolo di rotazione, le dimensioni e l’adattamento rispetto all’asse orizzontale e verticale.
* *position*, specifica la posizione del contenitore rispetto alla pagina nel quale viene inserito senza bisogno di specificare le coordinate spaziali
* *shape*, specifica la forma del contenitore, tra *RECTANGLE*, *CIRCLE* e *TRIANGLE*. Nel caso del rettangolo verranno usati i valori di posizione e dimensione per il disegno di esso. Nel caso del cerchio verranno utilizzati i valori di posizione e dimensione per disegnare un cerchio o una ellisse. Nel caso del triangolo oltre a posizione e dimensione sarà necessario l’inserimento di un attributo *cross-point* espresso come percentuale che specifica il punto H specificato in figura:



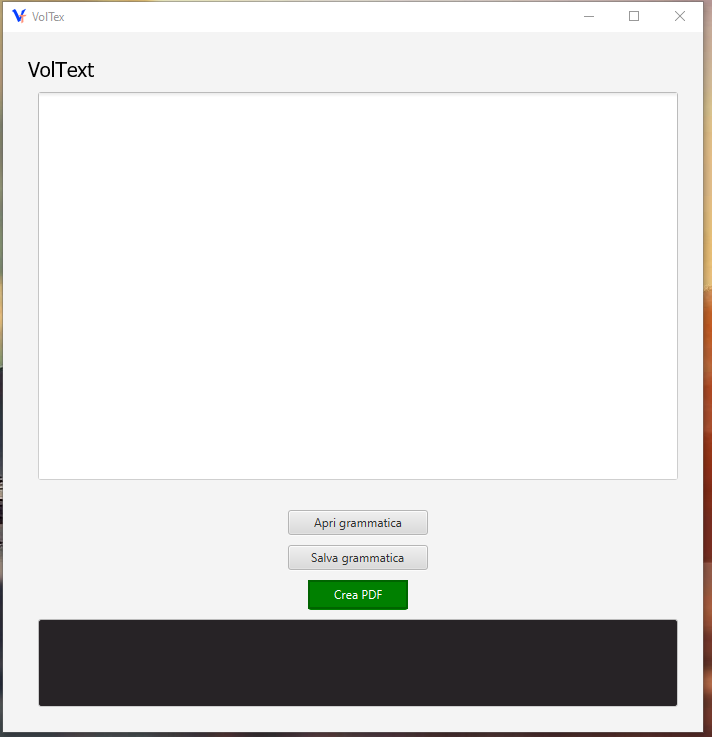
1. **Tool utilizzati**

Per lo sviluppo sono stati utilizzati diversi strumenti:

* GitHub: GitHub è un servizio di Version Control utilizzato nel nostro progetto per tenere traccia delle modifiche e dei mutamenti implementati nel nostro software. Ciò ha aiutato anche la condivisione quasi real-time di codice per un lavoro di gruppo migliore.
* GitHub Desktop: client che permette un utilizzo semplificato e rapido di GitHub, utile anche per tenere traccia dei cambiamenti e delle versioni del progetto.
* ANTLR 4: ANTLR (ANother Tool for Language Recognition) è un potente generatore di parser per leggere, elaborare, eseguire o tradurre testo strutturato o file binari. Da una grammatica, ANTLR genera un parser che può costruire e analizzare alberi di analisi. La versione 4 è stata utilizzata per stare al passo con le versioni e per una comodità d’uso dell'IDE Eclipse, vista la presenza di un plugin apposito.
* Eclipse: IDE per lo sviluppo della libreria e della GUI, che supporta anche vari plugin per ANTLR e JavaFX.
* PDFBox: libreria java per la creazione e modifica di documenti PDF
* PDFBox-layout: libreria java aggiuntiva a PDF-box che permette una gestione (non del tutto compatibile) semplificata di testi e liste.
* JavaFX: libreria che permette l’implementazione di una interfaccia grafica per un programma java. In particolare, per Eclipse, è stato utilizzato il plugin e(fx)clipse che permette la creazione di progetti JavaFX e l’utilizzo delle librerie annesse.
* Launch4j: software per l’incapsulamento di eseguibili di tipo exe altamente personalizzabili a partire da eseguibili di tipo jar.

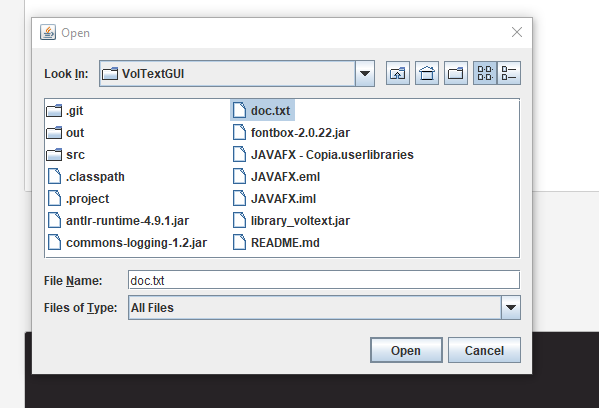
1. **Interfaccia utente**

L’applicazione è utilizzabile tramite un’interfaccia grafica sviluppata con JavaFX. All’esecuzione si presenta con una GUI composta da un’area testuale, in cui è possibile scrivere e modificare la grammatica di un documento PDF, due pulsanti simili tra loro, rispettivamente per l’apertura e il salvataggio di un file di testo, un pulsante verde per la creazione del documento, e un’area di testo non modificabile che serve per comunicare all’utente eventuali messaggi ed errori.

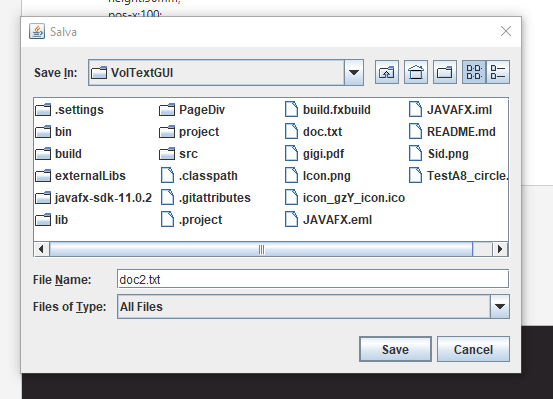


* 1. **Caso d’uso generale**

L’utente può scrivere la grammatica da zero, ma il caso d’uso più comune è sicuramente quello in cui la grammatica, o una sua parte, è in un file di testo già esistente. In questo caso con il pulsante “Apri grammatica” è possibile selezionare il file da caricare tramite un’interfaccia di navigazione del file system, ricevendo un messaggio di conferma.



Dopo aver aperto il file è possibile fare alla grammatica tutte le modifiche necessarie, e premendo il pulsante “Salva grammatica” si può salvarla in un file tramite un’interfaccia simile a quella usata durante il caricamento.



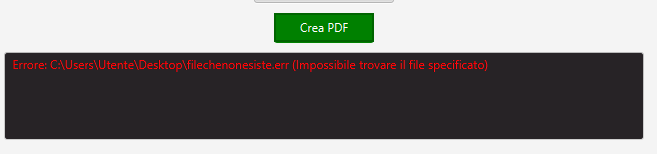
Quando l’utente è pronto per avviare la generazione del documento può farlo premendo il pulsante “Crea PDF”, al termine della procedura, nel caso vada a buon fine, viene mostrato un messaggio di conferma, oltre ad eventuali errori o avvisi da segnalare all’utente.



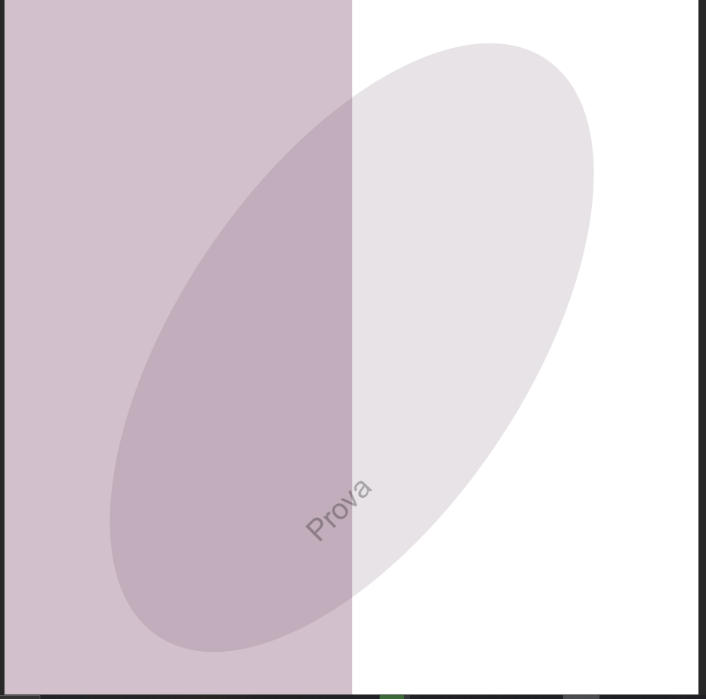
* 1. **Reliability**

L’applicazione è stata sviluppata per tollerare un utilizzo scorretto da parte dell’utente senza compromettere l’esecuzione del software, ma mostrando invece dei chiari messaggi di errore che possono aiutare l’utente a capirne la causa.

Ad esempio, nel caso durante l’apertura di un documento l’utente inserisca il nome di un file non esistente viene restituito il messaggio “Impossibile trovare il file specificato”, o ancora se il path di destinazione del file pdf esiste già ed è aperto in un altro programma viene mostrato il messaggio “Impossibile accedere al file. Il file è utilizzato da un altro processo”.

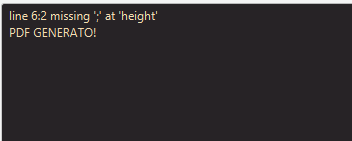


Un possibile esempio di grammatica è la seguente, la quale descrive un semplice documento PDF di nome Test contenente una sola pagina di dimensioni 100x100, due div colorati e un testo ruotato:



È interessante notare che l’applicazione genera un documento PDF anche in presenza di errori nella grammatica, restituendo comunque un messaggio di segnalazione per l’utente.

Per esempio, eliminando il punto e virgola dopo ‘width:100’ nella grammatica precedente, il software produce un documento corretto, segnalando però l’errore tramite l’interfaccia grafica.



1. **Possibili sviluppi futuri**

Durante lo sviluppo dell’applicativo sono emerse interessanti possibili aggiunte per l’ampliamento del progetto. Si riportano alcuni spunti:

* L’attributo *underline*, seppur implementato, non funziona in modo corretto a causa di metodi efficaci nella libreria PDFBox e di incompatibilità tra PDFBox e PDFBox-layout.

La generazione di una funzione che permetta di sottolineare il testo con una modalità più semplice in termini di programmazione e di calcoli.

* L’implementazione di costrutti quali tabelle, link, formule matematiche e grafici.
* Una possibile miglioria a livello grammaticale come semplificazione di costrutti, id senza doppi apici, gestione dei colori anche tramite gradienti o altri formati.
* Ottimizzazione del codice, soprattutto per la classe VolTextListener.java.
* Gestione di contenitori all’interno di altri contenitori, gestione di oggetti interni ai contenitori di forma non rettangolare, gestione interna alla grammatica di particolari costrutti per ridurre il documento redatto dall’utente.
* Implementazione di tool ausiliari per la visualizzazione in anteprima del documento.