**Documentazione**

|  |  |  |
| --- | --- | --- |
| **USER STORY:** | **Interfaccia base** | **Planned Story Points: 2**  **Effective Story Points: 2** |

**Come utente, voglio visualizzare una finestra inizialmente vuota così da poter iniziare un nuovo disegno.**

Immagine che contiene testo, schermata, schermo, software

Il contenuto generato dall'IA potrebbe non essere corretto.

**DESCRIZIONE FUNZIONALITA’**

All’avvio dell’applicazione, viene mostrata una lavagna bianca su cui è possibile inserire figure geometriche.

L’interfaccia mostra due barre laterali:

* A *sinistra*: la barra permette di selezionare una tra le figure : “Ellisse”, “Segmento”, “Rettangolo”;
* A *destra*: la barra permette di scegliere i colori di riempimento (selettore superiore) e colore bordo (selettore inferiore);

La toolbar superiore presenta i menu a tendina:

* “File”: mostra gli item “Salva con nome” e “Apri”;
* “Edit”: mostra l’item “Delete”;
* “Help” : mostra l’item “About”;

**TASK TECNICI**

L’interfaccia base è scritta in .xml, implementata utilizzando l’applicazione SceneBuilder. Le figure sulla sinistra sono dei ToggleButton; sulla destra ci sono dei ColorPicker.

|  |  |  |
| --- | --- | --- |
| **USER STORY:** | **Seleziona figura geometrica** | **Planned Story Points: 2**  **Effective Story Points: 2** |

**Come utente, voglio selezionare dalla toolbar una figura geometrica tra:**

* **segmento**
* **rettangolo**
* **ellisse**

**così da inserirla nel disegno.**

Immagine che contiene testo, schermo, schermata, computer

Il contenuto generato dall'IA potrebbe non essere corretto.

**DESCRIZIONE FUNZIONALITA’**

La selezione della figura geometrica avviene tramite il click su una delle figure presenti.

La selezione è mutamente esclusiva, quindi alla selezione di una figura si deseleziona la figura precedentemente selezionata. Se si preme due volte sulla stessa figura, la figura si deseleziona.

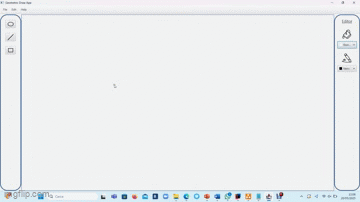
**TASK TECNICI**

L’implementazione della selezione mutamente esclusiva è stata gestita direttamente da *SceneBuiler*, inserendo i **ToggleButton** in un nodo **ToggleGroup** che gestisce questo tipo di selezione.

|  |  |  |
| --- | --- | --- |
| **USER STORY:** | **Inserimento e Posizionamento figura** | **Planned Story Points: 5**  **Effective Story Points: 5** |

**Come utente, voglio poter inserire la figura selezionata nel punto indicato dal mouse,**

**così da poter disegnare liberamente.**

****

**DESCRIZIONE FUNZIONALITA’**

Dopo aver selezionato la figura, è possibile inserirla liberamente sulla lavagna cliccando inizialmente sul punto in cui si vuole ancorare la figura, e trascinando il cursore fino al punto in cui si vuole estendere la figura.

Al momento della creazione, le figure hanno il bordo nero e il riempimento bianco.

**TASK TECNICI**

Per l’inserimento delle figure sulla lavagna sono stati utilizzati i **pattern Factory Method, Command, State.** Per gestire lo stato dell’applicazione in funzione del bottone premuto è stato utilizzato il **pattern State**.

Quando l'utente seleziona un tipo di figura (es. Rettangolo) tramite un ToggleButton, il Controller imposta lo **stato corrente** in StatoManager:

***statoManager.setStato(new DisegnaRettangoloStato(…))***

Dopo aver selezionato la figura, l’utente clicca (*1*), trascina (*2*) e rilascia (*3*) il cursore sulla lavagna. A queste operazioni sono associati i corrispondenti eventi.

1. Il click sulla lavagna viene gestito da:

***lavagna.setOnMousePressed(e -> statoManager.getStato().onMousePressed(e))***

La *OnMousePressed* dello stato corrente effettua due operazioni:

* Salva la posizione iniziale del mouse;
* Mostra una figura temporanea sulla lavagna (un Rectangle traslucido) per il feedback visivo;

1. Il trascinamento della figura temporanea viene gestito da:

***lavagna.setOnMouseDragged(e -> statoManager.getStato().onMouseDragged(e))***

La *OnMouseDragged* dello stato corrente aggiorna dinamicamente le dimensioni della figura temporanea in base alla posizione corrente del mouse.

1. Il rilascio del pulsante del mouse viene gestito da:

***lavagna.setOnMouseRelease(e -> statoManager.getStato().onMouseReleased(e))***

La *onMouseReleased* dello stato corrente:

* Rimuove la figura temporanea dalla lavagna.
* Crea ed **esegue un comando** per inserire la figura reale nel modello (**LavagnaModel**) tramite l’Invoker.

La creazione e l’esecuzione del comando per inserire la figura reale sono implementate dalle seguenti righe:

***Command cmd = new AggiungiFiguraCommand(model, new RettangoloFactory(), ...);***

***Invoker.getInstance().executeCommand(cmd);***

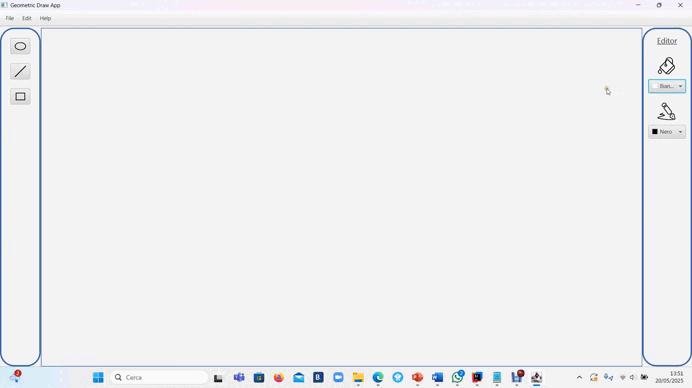
in cui si chiede ad una **Factory** (es. RettangoloFactory) di creare un oggetto Figura (es. un Rettangolo) con le coordinate e i colori. La execute di *AggiungiFiguraCommand*, aggiunge la figura al **modello della lavagna** (*LavagnaModel*) e la proietta sulla lavagna reale.

|  |  |  |
| --- | --- | --- |
| **USER STORY** | **Colore riempimento figura** | **Planned Story Points: 3**  **Effective Story Points: 3** |

**Come utente,**

**voglio poter scegliere uno tra i colori disponibili (almeno 8) per colorare l’interno della figura in caso essa sia chiusa,**

**così da poterla personalizzare.**

****

**DESCRIZIONE FUNZIONALITA’**

Prima di inserire la figura, è possibile scegliere il colore di riempimento.

All’inserimento, la figura avrà il colore di riempimento presente nel selettore di colore superiore.

**TASK TECNICI**

Quando l’utente seleziona una figura (es. Rettangolo), nel Controller viene creato un oggetto Stato specifico, al quale viene passato il valore del ColorPicker corrente:

***statoManager.setStato(new DisegnaRettangoloStato(…, fillColor, …))***

Alla fine del disegno (onMouseReleased), viene creato ed eseguito *AggiungiFiguraCommand* in cui il colore reale viene passato al comando:

***Command cmd = new AggiungiFiguraCommand(model, new RettangoloFactory(), fillColor, ...);***

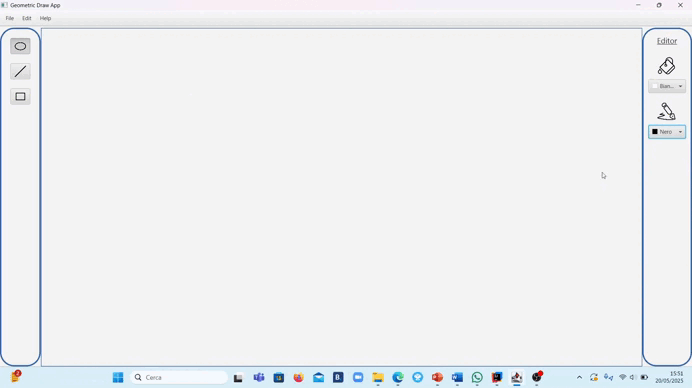
***Invoker.getInstance().executeCommand(cmd);***

|  |  |  |
| --- | --- | --- |
| **USER STORY** | **Colore Bordo figura** | **Planned Story Points: 3**  **Effective Story Points: 3** |

**Come utente,**

**voglio poter colorare il bordo della figura (almeno 8 colori),**

**affinché la possa personalizzare.**

****

**DESCRIZIONE FUNZIONALITA’**

Prima di inserire la figura, è possibile scegliere il colore del bordo.

All’inserimento, la figura avrà il colore del bordo presente nel selettore di colore inferiore.

**TASK TECNICI**

Quando l’utente seleziona una figura (es. Rettangolo), nel Controller viene creato un oggetto Stato specifico, al quale viene passato il valore del ColorPicker corrente:

***statoManager.setStato(new DisegnaRettangoloStato(…, strokeColor, …))***

Alla fine del disegno (onMouseReleased), viene creato ed eseguito *AggiungiFiguraCommand* in cui il colore reale viene passato al comando:

***Command cmd = new AggiungiFiguraCommand(model, new RettangoloFactory(), strokeColor, ...);***

***Invoker.getInstance().executeCommand(cmd);***

|  |  |  |
| --- | --- | --- |
| **USER STORY** | **Salvare Disegno** | **Planned Story Points: 5**  **Effective Story Points: 5** |

**Come utente,**

**voglio poter salvare il disegno in un file,**

**così da poterlo conservare o caricare in un secondo momento.**

****

**DESCRIZIONE FUNZIONALITA’**

L’Utente può salvare il disegno fino a quel momento creato.

**TASK TECNICI**

Il click dell’Utente sul menu item **Salva con nome**, genera l’evento

**salvaConNome.setOnAction(e ->{  
 Command cmd = new SalvaFiguraCommand(salvaConNome, lavagnaModel);  
 Invoker.*getInstance*().executeCommand(cmd);  
 System.*out*.println("FIGURA SALVATA");  
});**

Viene creato un oggetto **Command**  di tipo **SalvaFiguraCommand** successivamento passato all’Invoker che ne chiama la *execute()*.

La *execute* di **SalvaFiguraCommand** crea una copia delle figure inserite fino a quel momento e ne salva le proprietà in un file .txt. Le proprietà salvate sono:

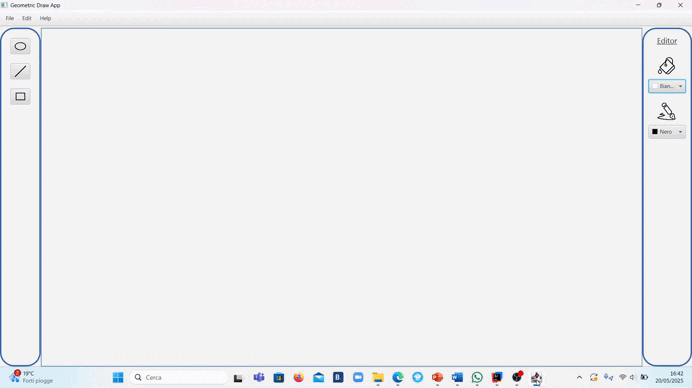
* Tipo di figura (es. Ellisse);
* Coordinate (x1,y1, x2,y2);
* Colore riempimento (fillColor);
* Colore bordo.

|  |  |  |
| --- | --- | --- |
| **USER STORY** | **Caricare disegno** | **Planned Story Points: 5**  **Effective Story Points: 5** |

**Come utente,**

**voglio poter caricare il disegno in un file,**

**così da poterlo conservare o caricare in un secondo momento.**

****

**DESCRIZIONE**

L’Utente può salvare il disegno fino a quel momento creato.

**TASK TECNICI**

Il click dell’Utente sul menu item **Apri**, genera l’evento:

**caricaFile.setOnAction(e->{  
 Command cmd = new CaricaCommand(lavagnaModel, caricaFile);  
 Invoker.*getInstance*().executeCommand(cmd);  
 System.*out*.println("FILE CARICATO");  
});**

Viene creato un oggetto **Command**  di tipo **CaricaCommand** successivamente passato all’Invoker che ne chiama la *execute()*.

La *execute* di **CaricaCommand** legge dal file .txt che l’Utente sceglie le proprietà inserite facendo **SalvaFigura**.

In caso di errata formattazione del testo, il comando stampa a video un messaggio di Errore.