UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E MATEMATICA APPLICATA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

RELAZIONE DI

SOFTWARE ARCHITECTURE DESIGN

Docente:
Prof. Ritrovato Pierluigi

Gruppo:
11 – GeoDraw

Membri:
Ferrara Francesco – 0623200064
Luongo Leonardo – 0622702606
Monda Francesco – 0622702620
Parente Alessia – 0623200061

Anno Accademico 2024/2025

INDICE

FASE 0: PRE-GAME

1. Definizione della "Definition of Done" (DoD)

- **Implementazione sviluppata:**

  Il codice relativo alla User Story è stato completato e correttamente integrato nel flusso di lavoro principale.

- **Analisi del codice:**

  Il codice è stato sottoposto a “Code Review” e approvato dal team.

- **Test completati:**

  Sono stati scritti e superati Test Driven per il codice sviluppato.

  Sono stati eseguiti test di integrazione per verificare il corretto funzionamento con altri moduli.

  Se necessario, sono stati eseguiti test manuali o test funzionali.

- **Assenza di bug critici:**

  Nessun errore bloccante o crash.

2. Product Backlog

Durante l'analisi dei requisiti funzionali per l'applicazione di disegno geometrico, è stato adottato l'approccio **Agile User-Centered Design**, con l'obiettivo di rappresentare chiaramente, dal punto di vista dell'utente, tutte le funzionalità richieste dal sistema. A tal fine, ogni funzionalità è stata convertita in una **User Story** secondo il formato:

Come utente voglio *[obiettivo]* **per** *[beneficio]*

A ciascuna user story sono stati associati dei **criteri di accettazione** formulati secondo il pattern:

Dato che *[contesto iniziale]*, **Quando** *[evento/interazione]*, **Allora** *[risultato atteso]*

Questo approccio consente una chiara tracciabilità tra i requisiti e la loro implementazione concreta, garantendo che ogni funzionalità sia verificabile tramite condizioni ben definite.

1. **Apertura finestra**

   Come utente voglio visualizzare una finestra inizialmente vuota per iniziare un nuovo disegno da zero.

   Criteri di accettazione
   AC1: Dato che avvio l'applicazione,
   Quando si apre la finestra principale,
   Allora il canvas deve essere privo di figure.

   Priorità: Alta (5)
   Story points: 1
   Ruolo: User

2. **Aggiunta forma**

   Come utente voglio aggiungere una forma geometrica definendone i punti o l'estensione direttamente sul canvas tramite un'operazione di click-e-trascinamento (drag) per costruirla in modo interattivo.

   Criteri di accettazione

   AC1: Dato che ho selezionato uno strumento di disegno (es. 'Rettangolo'),
   Quando premo il pulsante del mouse sul canvas (mousePressed), tengo premuto e trascino il cursore (mouseDragged), e poi rilascio il pulsante del mouse (mouseReleased),
   Allora una forma del tipo selezionato viene creata sul canvas, definita dai punti di inizio e fine del trascinamento.

Priorità: Alta (5)
Story points: 3
Ruolo: User

3. Selettore Strumenti di Disegno Base

Come utente voglio avere una toolbar o un menu visibile con opzioni chiare per selezionare lo strumento 'Segmento', 'Rettangolo' o 'Ellisse' per poter scegliere quale tipo di forma base iniziare a disegnare.

Criteri di accettazione
AC1: Dato che l'applicazione è avviata,
Quando guardo l'interfaccia principale,
Allora vedo una sezione (menu) con pulsanti chiaramente identificabili per '*Segmento*', '*Rettangolo*' ed '*Ellisse*'.

AC2: Dato che la sezione degli strumenti è visibile,
Quando clicco sull'opzione '*Rettangolo*',
Allora lo strumento '*Rettangolo*' diventa lo strumento di disegno attivo (visivamente indicato, se applicabile) e gli altri strumenti base (Segmento, Ellisse) sono disattivi.

AC3: Dato che lo strumento '*Rettangolo*' è attivo,
Quando clicco sull'opzione '*Segmento*',
Allora lo strumento '*Segmento*' diventa attivo e '*Rettangolo*' si disattiva.

Priorità: Alta (5)
Story points: 1
Ruolo: User

4. Disegnare Segmenti di Linea

Come utente voglio poter disegnare un segmento di linea sul canvas dopo aver selezionato lo strumento 'Segmento' per creare elementi lineari nel mio disegno.

Criteri di accettazione
AC1: Dato che lo strumento '*Segmento*' è attivo,
Quando clicco un punto iniziale (X1, Y1) sul canvas e poi un punto finale (X2, Y2) rilascio,
Allora un segmento di linea visibile con il colore 'Nero' (colore di default) viene disegnato tra (X1, Y1) e (X2, Y2).

Priorità: Alta (5)
Story points: 2
Ruolo: User

5. **Disegnare Rettangoli**

Come utente voglio poter disegnare un rettangolo sul canvas dopo aver selezionato lo strumento 'Rettangolo' per creare aree rettangolari nel mio disegno.

Criteri di accettazione
AC1: Dato che lo strumento '*Rettangolo*' è attivo e ho selezionato colori per bordo ('Rosso') e interno ('Blu'),
Quando clicco e tengo premuto su un angolo (X1, Y1), trascino fino all'angolo opposto (X2, Y2) e rilascio,
Allora un rettangolo viene disegnato con il bordo 'Nero' e l'interno riempito di 'Bianco' (colori di default), definito dai punti (X1, Y1) e (X2, Y2).

Priorità: Alta (5)
Story points: 2
Ruolo: User

6. **Disegnare Ellissi**

Come utente voglio poter disegnare un'ellisse sul canvas dopo aver selezionato lo strumento 'Ellisse' per creare forme curve o circolari nel mio disegno.

Criteri di accettazione
AC1: Dato che lo strumento '*Ellisse*' è attivo,
Quando clicco e tengo premuto su un punto (X1, Y1) (che definisce un angolo del bounding box), trascino fino al punto opposto (X2, Y2) e rilascio,
Allora un'ellisse inscritta nel rettangolo definito da (X1, Y1) e (X2, Y2) viene disegnata con il bordo 'Nero' e l'interno riempito di 'Bianco' (colori di default).

Priorità: Alta (5)
Story points: 2
Ruolo: User

7. **Cambio del colore del bordo**

Come utente voglio selezionare il colore del bordo della forma selezionata per personalizzare l'estetica.

Criteri di accettazione
AC1: Dato che ho selezionato un colore del bordo es. 'Rosso',
Quando creo la figura,
Allora il bordo della forma creata sarà 'Rosso'.

AC2: Dato che ho selezionato una forma (es. un rettangolo) sul canvas,
Quando scelgo il colore 'Rosso' dalla palette dei colori del bordo nella GUI,
Allora il bordo della forma selezionata cambia immediatamente in 'Rosso'.

Priorità: Alta (5)
Story points: 2
Ruolo: User

8. **Impostare colore interno della figura selezionata**

Come utente voglio scegliere il colore interno di una forma chiusa selezionata per evidenziare meglio l'area interna della forma.

Criteri di accettazione
AC1: Dato che ho selezionato un colore interno es. 'Rosso',
Quando creo la figura,
Allora il colore interno della forma creata sarà 'Rosso'.

AC2: Dato che ho selezionato una forma chiusa (es. un'ellisse) sul canvas,
Quando scelgo il colore 'Blu' dalla palette dei colori interni sulla GUI,
Allora il colore di riempimento della forma selezionata cambia immediatamente in 'Blu'.

AC3: Dato che ho selezionato un segmento di linea (forma non chiusa),
Quando tento di applicare un colore dalla palette dei colori interni,
Allora il segmento rimane visibilmente invariato.

Priorità: Alta (5)
Story points: 2
Ruolo: User

9. **Salvataggio disegno**

Come utente voglio salvare il disegno in un file per conservarlo ed eventualmente modificarlo in futuro.

Criteri di accettazione
AC1: Dato che ho completato un disegno,
Quando clicco su '*Salva*',
Allora il file deve contenere tutte le forme e i relativi attributi (tipo, posizione, dimensione, colore bordo, colore interno, etc…).

Priorità: Alta (5)
Story points: 2
Ruolo: User

10. Caricamento disegno

Come utente voglio caricare un disegno da file per poterlo modificare o visualizzare.

Criteri di accettazione
AC1: Dato che ho un file salvato,
Quando clicco su '*Carica*',
Allora il canvas viene prima svuotato e poi tutte le forme dal file vengono disegnate con i loro attributi corretti.

AC2: Dato che ho un file salvato corrotto,
Quando clicco su '*Carica*',
Allora viene mostrato un messaggio di errore in una finestra ma il sistema rimane integro.

Priorità: Alta (5)
Story points: 2
Ruolo: User

11. Eliminazione forma

Come utente voglio eliminare la forma selezionata per correggere il disegno.

Criteri di accettazione
AC1: Dato che ho selezionato una forma,
Quando premo il tasto '*Canc*',
Allora la forma deve essere rimossa dal disegno.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

12. Spostamento forma

Come utente voglio spostare una forma selezionata per riorganizzare il layout del disegno.

Criteri di accettazione
AC1: Dato che ho selezionato una forma,
Quando trascino la forma,
Allora deve seguire il movimento del mouse.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

13. Ridimensionamento forma

Come utente voglio ridimensionare una forma selezionata per modificarne l'estensione.

Criteri di accettazione
AC1: Dato che ho selezionato una forma,
Quando trascino i bordi o modifico le dimensioni dalla GUI,
Allora il vecchio rettangolo viene rimosso e viene immediatamente creato il nuovo con le nuove dimensioni.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

14. Taglia forma

Come utente voglio tagliare una forma selezionata per riorganizzare il disegno.

Criteri di accettazione
AC1: Dato che ho selezionato una forma,
Quando clicco sull'icona '*Taglia*',
Allora la forma deve sparire ma restare in memoria.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

15. Copia forma

Come utente voglio copiare una forma selezionata per duplicarla in altre posizioni.

Criteri di accettazione
AC1: Dato che ho selezionato una forma,
Quando clicco sull'icona '*Copia*',
Allora la forma deve essere memorizzata negli appunti.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

16. Incolla forma

Come utente voglio incollare una forma precedentemente copiata o tagliata per riposizionarla nel disegno.

Criteri di accettazione
AC1: Dato che ho una forma negli appunti,
Quando clicco sull'icona '*Incolla*',
Allora deve apparire una copia nel disegno alle coordinate della figura copiata con un piccolo offset sommato.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

17. Gestione cronologia

Come utente voglio poter gestire una cronologia delle azioni effettuate sulla figura, per poter annullare 'Undo' o ripristinare 'Redo' operazioni.

Criteri di accettazione
AC1: Dato che ci sono delle operazioni modificabili fatte in passato,
Quando clicco sul comando '*Undo*',
Allora il disegno torna allo stato precedente dell'ultima azione modificabile compiuta.

AC2: Dato che sono state fatte delle azioni di '*Undo*',
Quando clicco sul comando '*Redo*',
Allora il disegno torna allo stato precedente dell'ultima azione modificabile compiuta.

AC3: Dato che tutte le operazioni modificabili sono assenti o già annullate,
Quando clicco sul comando '*Undo*',
Allora il disegno mantiene il suo stato attuale e il comando '*Annulla*' appare disabilitato oppure senza produrre alcun cambiamento visibile.

AC4: Dato che ci sono delle operazioni modificabili memorizzate come future rispetto allo stato corrente,
Quando clicco sul comando '*Redo*',
Allora il disegno mantiene il suo stato attuale e il comando '*Annulla*' appare disabilitato oppure senza produrre alcun cambiamento visibile.

Priorità: Medio alta (4)
Story points: 5
Ruolo: User

18. Gestione livelli

Come utente voglio mandare una forma selezionata in primo piano per renderla visibile sopra le altre o in secondo piano per nasconderla sotto ad altre.

Criteri di accettazione
AC1: Dato che due forme si sovrappongono,
Quando clicco su '*Porta avanti*',
Allora la forma selezionata deve apparire sopra.

AC2: Dato che due forme si sovrappongono,
Quando clicco su '*Porta indietro*',
Allora la forma selezionata deve andare sotto.

Priorità: Medio alta (4)
Story points: 2
Ruolo: User

19. Cambio livello zoom

Come utente voglio cambiare il livello di zoom del disegno per osservare meglio i dettagli.

Criteri di accettazione
AC1: Dato che ho aperto un disegno,
Quando modifico lo zoom con i pulsanti +/-,
Allora la visualizzazione deve adattarsi in scala.

AC2: Dato che ho aperto un disegno,
Quando modifico lo zoom con i pulsanti +/-,
Allora sono disponibili almeno 4 livelli di zoom (25%, 50%, 100%, 200%).

Priorità: Media (3)
Story points: 3
Ruolo: User

20. Scorrimento disegno *

Come utente voglio scorrere l'area di disegno quando è più grande della finestra per accedere a tutte le parti.

Criteri di accettazione
AC1: Dato che l'area di disegno è più grande dell'area visibile della finestra,
Quando clicco a vuoto, tengo premuto e muovo il cursore,
Allora la vista deve spostarsi di conseguenza.

Priorità: Media (3)
Story points: 1
Ruolo: User

21. Attivazione griglia

Come utente voglio attivare o disattivare una griglia per aiutarmi nel posizionamento preciso.

Criteri di accettazione
AC1: Dato che voglio precisione,
Quando attivo la griglia,
Allora devono comparire le linee guida sul disegno.

AC2:
Dato che posso fare a meno di precisione e c'è la griglia,
Quando disattivo la griglia,
Allora devono scomparire le linee guida sul disegno.

Priorità: Media (3)
Story points: 2
Ruolo: User

22. Scelta dimensione griglia

Come utente voglio scegliere la dimensione della griglia per adattarla al tipo di disegno.

Criteri di accettazione
AC1: Dato che la griglia è visibile,
Quando cambio dimensione,
Allora la spaziatura tra le linee deve aggiornarsi riducendosi o allargandosi.

Priorità: Media (3)
Story points: 3
Ruolo: User

23. Disegno poligono arbitrario

Come utente voglio disegnare un poligono arbitrario per rappresentare forme personalizzate.

Criteri di accettazione:
AC1: Dato che seleziono la modalità poligono,
Quando clicco più punti,

Allora si deve formare un contorno chiuso (l'ultimo click deve essere fatto sul punto iniziale).

Priorità: Media (3)
Story points: 5
Ruolo: User

24. Inserimento testo

Come utente voglio inserire del testo come forma per aggiungere etichette e annotazioni.

Criteri di accettazione
AC1: Dato che seleziono lo strumento testo e clicco sul canvas per definire la sa posizione,
Quando digito una stringa,
Allora essa deve apparire come forma nel disegno.

Priorità: Medio bassa (2)
Story points: 2
Ruolo: User

25. Dimensione testo

Come utente voglio scegliere la dimensione del testo per renderlo leggibile.

Criteri di accettazione:
AC1: Dato che ho inserito del testo,
Quando seleziono la dimensione dal menu sulla GUI,
Allora il font deve aggiornarsi di conseguenza.

Priorità: Medio bassa (2)
Story points: 2
Ruolo: User

26. Rotazione forma

Come utente voglio ruotare una forma per orientarla secondo necessità.

Criteri di accettazione
AC1: Dato che seleziono una forma,
Quando imposto un angolo in una input box nella GUI,
Allora essa deve ruotare correttamente intorno al centro della figura.

Priorità: Medio bassa (2)

Story points: 3
Ruolo: User

27. Riflessione forma

Come utente voglio riflettere una forma orizzontalmente o verticalmente per modificarne la simmetria.

Criteri di accettazione
AC1: Dato che ho selezionato una forma,
Quando clicco sul comando 'Rifletti Orizzontalmente' (es. da un menu o pulsante dedicato nella GUI),
Allora la forma selezionata deve specchiarsi rispetto al suo asse verticale centrale, invertendo la sua orientazione orizzontale.

AC2: Dato che ho selezionato una forma,
Quando clicco sul comando 'Rifletti Verticalmente' (es. da un menu o pulsante dedicato nella GUI),
Allora la forma selezionata deve specchiarsi rispetto al suo asse orizzontale centrale, invertendo la sua orientazione verticale.

Priorità: Medio bassa (2)
Story points: 3
Ruolo: User

28. Stretching forma

Come utente voglio stirare una forma orizzontalmente o verticalmente per modificarne le proporzioni.

Criteri di accettazione
AC1: Dato che seleziono una forma,
Quando premo il pulsante “S” dalla tastiera e trascino i bordi,
Allora solo l’asse scelto deve essere alterato.

Priorità: Medio bassa (2)
Story points: 3
Ruolo: User

29. Raggruppamento forme

Come utente voglio raggruppare più forme per gestirle come un'unica entità.

Criteri di accettazione

AC1: Dato che ho selezionato almeno due forme distinte sul canvas,
Quando attivo il comando '*Raggruppa*' (tramite pulsante o menu contestuale),
Allora le forme selezionate vengono trattate come un unico gruppo logico e rimangono visivamente selezionate come gruppo (es. con un unico riquadro di selezione che le contiene tutte).

AC2: Dato che esiste un gruppo di forme sul canvas (e non è attualmente selezionato),
Quando clicco su una qualsiasi delle forme che appartengono a quel gruppo,
Allora l'intero gruppo (tutte le forme al suo interno) viene selezionato e indicato visivamente come tale.

AC3: Dato che un gruppo è selezionato,
Quando trascino il gruppo (cliccando e spostando una qualsiasi delle sue forme interne),
Allora tutte le forme appartenenti al gruppo si spostano insieme mantenendo le loro posizioni relative l'una rispetto all'altra.

AC4: Dato che un gruppo è selezionato,
Quando premo il tasto '*Canc*' (o attivo un comando '*Elimina*'),
Allora tutte le forme che appartengono al gruppo vengono rimosse dal disegno.

AC5: Dato che esiste un gruppo e altre forme singole disaggregate,
Quando seleziono e sposto il gruppo,
Allora solo le forme appartenenti al gruppo cambiano posizione, mentre le forme singole mantengono la loro posizione originale.

Priorità: Bassa (1)
Story points: 5
Ruolo: User

30. Separa il raggruppamento forme

Come utente, voglio poter dividere le forme precedentemente raggruppate, per modificarle o spostarle singolarmente in modo indipendente.

Criteri di accettazione
AC1: Dato che seleziono un gruppo di forme precedentemente creato,
Quando attivo il comando 'Separa Gruppo' (tramite GUI),
Allora le forme che componevano il gruppo diventano nuovamente forme individuali e indipendenti, e la selezione di gruppo viene rimossa (nessuna forma deve essere selezionata dopo la separazione).

Priorità: Bassa (1)
Story points: 3
Ruolo: User

31. Creare una Forma Riutilizzabile

Come utente voglio salvare una forma selezionata (o un gruppo) con un nome specifico per poterla riutilizzare rapidamente all'interno dello stesso disegno.

Criteri di accettazione
AC1: Dato che ho selezionato una forma singola (o un gruppo precedentemente creato), Quando attivo l'opzione "Salva come Forma Riutilizzabile" (es. da menu o pulsante), Allora mi viene presentata un'interfaccia per inserire un nome univoco per questa forma.

AC2: Dato che ho inserito un nome valido ("MioLogo") e confermo,
Quando l'operazione di salvataggio termina,
Allora una definizione della forma *così com'è ora* (snapshot di tipo, dimensioni, colori, posizioni relative se gruppo) viene memorizzata internamente associata al nome "MioLogo".

AC3: Dato che ho salvato una forma riutilizzabile con nome "MioLogo",
Quando consulto l'area UI dedicata alle forme riutilizzabili (es. una palette/libreria laterale),
Allora vedo un'icona o voce etichettata "MioLogo" pronta per essere usata.

AC4: Dato che salvo una forma riutilizzabile "MioLogo", e successivamente modifico o elimino la forma originale sul canvas,
Quando uso il comando "MioLogo" per inserire una nuova copia,
Allora la copia inserita corrisponde alla versione della forma al momento del salvataggio, indipendentemente da eventuali modifiche o eliminazioni successive.

Priorità: Bassa (1)
Story points: 5
Ruolo: User

32. Usare una Forma Riutilizzabile

Come utente voglio selezionare una forma riutilizzabile precedentemente creata e inserirne una copia nel disegno per velocizzare la creazione di elementi ripetitivi.

Criteri di accettazione
AC1: Dato che nell'area UI delle forme riutilizzabili è presente "MioLogo",
Quando seleziono "MioLogo" (es. cliccandoci sopra),
Allora lo strumento di inserimento per "MioLogo" diventa attivo.

AC2: Dato che lo strumento "MioLogo" è attivo,
Quando clicco su una posizione (X, Y) nel canvas,
Allora una *nuova copia* della forma (o gruppo) definita da "MioLogo" viene aggiunta al disegno centrata o posizionata in (X, Y) (definire il comportamento esatto).

AC3: Dato che ho inserito una copia tramite il comando "MioLogo",
Quando seleziono la copia appena inserita,
Allora essa si comporta come una normale forma (o gruppo) indipendente, modificabile separatamente dall'originale o da altre copie.

Priorità: Bassa (1)
Story points: 3
Ruolo: User

33. Esportare Forme Riutilizzabili in una Libreria

Come utente, voglio esportare le definizioni delle forme riutilizzabili create in un file di libreria separato, per poterle condividere o riutilizzare in altri disegni.

Criteri di accettazione
AC1: Dato che ho creato delle forme riutilizzabili,
Quando attivo l'opzione "Esporta Libreria Forme",
Allora mi viene presentata un'interfaccia per scegliere quali forme riutilizzabili esportare (es. tutte o una selezione) e per specificare un nome e una posizione per il file di libreria (es. con estensione .geolib).

AC2: Dato che ho selezionato le forme da esportare, fornito un nome file valido e confermato,
Quando l'esportazione termina,
Allora viene creato un file di libreria nel percorso specificato contenente le definizioni delle forme selezionate in un formato riutilizzabile.

Priorità: Bassa (1)
Story points: 3
Ruolo: User

34. Importare Forme Riutilizzabili da una Libreria

Come utente voglio importare definizioni di forme riutilizzabili da un file di libreria per poter usare forme predefinite o create in altri disegni.

Criteri di accettazione
AC1: Dato che voglio usare forme da una libreria esterna,
Quando attivo l'opzione "Importa Libreria Forme",
Allora mi viene richiesto di selezionare un file di libreria valido (es. .geolib).

AC2: Dato che seleziono un file di libreria valido,
Quando l'importazione termina,
Allora l'area UI delle forme riutilizzabili viene aggiornata includendo le forme importate dalla libreria, rendendole disponibili per l'uso nel disegno corrente (storia 29.B).

AC3: Dato che tento di importare un file non valido o corrotto,
Quando l'operazione fallisce,
Allora viene mostrato un messaggio di errore appropriato e nessuna nuova forma viene aggiunta alla UI delle forme riutilizzabili.

AC4: Dato che importo una libreria contenente una forma riutilizzabile con un nome che già esiste nel disegno corrente,
Quando l'importazione avviene,
Allora rinomina automaticamente la libreria importata.

Priorità: Bassa (1)
Story points: 5
Ruolo: User

3. First Sprint Planning

3.1. Calcolo stima della Velocity

La velocity rappresenta il numero di **Story Points (SP)** che il team può completare in uno sprint. Per stimarla, abbiamo considerato:

- Team composto da **4 sviluppatori**
- **8 ore/settimana** per sviluppatore
- Applicazione di un **focus factor del 75%**

La scelta del focus factor leggermente superiore allo standard Scrum (70%) è giustificata dal fatto che le 8 ore settimanali dedicate al progetto sono svolte **durante l'orario di laboratorio**, in un contesto controllato e privo di interruzioni esterne.

Per ottenere quindi:

$$4\ sviluppatori\ \times\ 8\ ore/settimana\ \times\ 0.75\ =\ 24\ h\ reali\ disponibili$$

Abbiamo adottato una stima realistica di **1 Story Point ≈ 40 minuti** di lavoro, basata sull'esperienza concreta del team. In particolare, abbiamo preso come riferimento la user story **#1 – Apertura finestra**, che richiede l'integrazione di un'interfaccia già prototipata in **Figma** e convertita in FXML, stimando un effort reale di circa **40 minuti** per la sua realizzazione end-to-end (integrazione FXML + logica di apertura). Otteniamo che la velocity stimata è quindi pari a:

$$24\ h\ \div\ 0.66\ h/SP\ \approx\ 36\ SP\ per\ sprint$$

Con una velocity stimata di **36 SP/sprint**, e una pianificazione di circa **30 SP per sprint**, riusciamo a completare il backlog in **3 sprint**, come richiesto. La pianificazione sfrutta quasi interamente la capacità disponibile. Con una velocity di **36 SP/sprint**, abbiamo un margine di sicurezza di oltre **15 SP** in totale, che potrà essere utilizzato per testing, bugfixing, polishing o per assorbire eventuali imprevisti. Abbiamo previsto la possibilità di adattare la pianificazione solo nel caso in cui, a fine Sprint 1, la velocity reale si discosti in modo significativo da quella stimata.

3.2. User Stories Selezionate

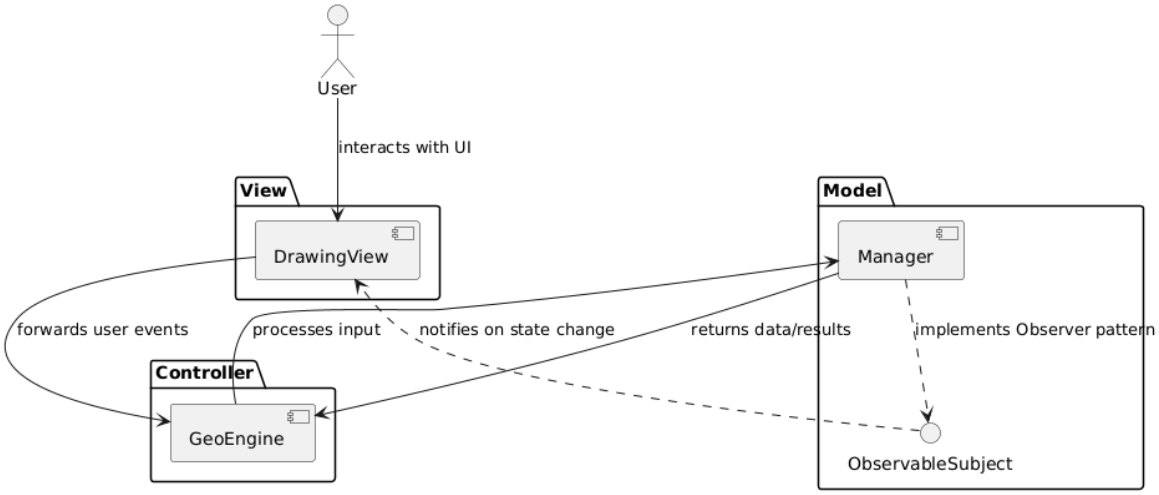
ID US	Titolo User Story	Story Points
1	Apertura finestra	1
2	Aggiunta forma	3
3	Selettore strumenti di disegno base	1
4	Disegnare segmenti di linea	2
5	Disegnare rettangoli	2
6	Disegnare ellissi	2
7	Cambio colore bordo	2
8	Colore interno figura selezionata	2
9	Salvataggio disegno	2
10	Caricamento disegno	2
11	Eliminazione forma	2
12	Spostamento forma	2
13	Ridimensionamento forma	2
14	Taglia forma	2
15	Copia forma	2

3.3. Software Architecture

L'architettura software adottata per il progetto è basata sul pattern **MVC (Model-View-Controller)**, una struttura che suddivide l'applicazione in tre componenti principali:

- **Model**: gestisce la logica di business e lo stato dei dati dell'applicazione.
- **View**: si occupa della presentazione visiva dei dati e dell'interfaccia utente.
- **Controller**: coordina le interazioni dell'utente, aggiornando il Model e la View in modo coerente.

Questa scelta architetturale è motivata dalla necessità di garantire una separazione netta delle responsabilità, facilitando così la manutenibilità, scalabilità e testabilità del sistema. Inoltre, la struttura MVC consente una collaborazione efficiente tra i membri del team, rendendo possibile lo sviluppo parallelo di componenti diversi (interfaccia, logica, gestione dati). In contesti di sviluppo agile con iterazioni rapide, l'adozione di questo pattern risulta particolarmente vantaggiosa.



Il **package View** contiene il componente ***DrawingView***, che espone all'utente l'interfaccia grafica per interagire con la mappa e gli strumenti di disegno. Ogni evento generato dall'utente (click, drag, selezione di strumenti), viene catturato da ***DrawingView*** e inoltrato al Controller, garantendo così un livello di astrazione fra l'interfaccia e la logica di business.

Flusso delle interazioni

Il **package Controller** ospita il componente **GeoEngine**, incaricato di elaborare gli input provenienti dalla View. ***GeoEngine*** riceve gli eventi dell'utente, li traduce in comandi di alto livello e li invia al Model per l'effettiva manipolazione dei dati. Una volta completata l'elaborazione, ***GeoEngine*** riceve indietro i risultati o gli stati aggiornati dal Model e li rende disponibili alla View, che provvede a ridisegnare la mappa in modo sincronizzato.

Gestione dei dati e notifica

Nel **package Model**, il componente **Manager** implementa la logica di gestione dei dati geografici (creazione, modifica, salvataggio). Manager realizza anche l'**ObservableSubject**, un'interfaccia che incapsula il pattern Observer. Ogni volta che lo stato interno del Model cambia (ad esempio, un nuovo oggetto geometrico viene creato o aggiornato), Manager notifica ***DrawingView*** tramite ***ObservableSubject***. In questo modo la View resta sempre coerente con il Model senza dover periodicamente interrogarlo ("polling") e mantenendo un basso grado di accoppiamento.

3.4. Guida per lo sviluppo dei componenti.

3.4.1. Model: Manager

- **Funzione**: Il **Model** rappresenta la logica centrale del sistema, che gestisce i dati e le operazioni di calcolo. In questo caso, il manager è il componente che memorizza e gestisce i dati relativi all'applicazione.
- **Flusso**: Il **Model** riceve input dal controller, che invia i dati per l'elaborazione. Il flusso di dati dal controller al modello implica che il modello esegua un'elaborazione basata sui dati ricevuti.

3.4.2. Controller: GeoEngine

- **Funzione**: Il **Controller** è responsabile per la logica che media l'interazione tra il **Model** e la **View**. In questo caso, il controller **GeoEngine** gestisce l'inserimento di valori o operazioni e coordina l'elaborazione dei dati nel modello. Si occupa di prendere l'input dell'utente, inviarlo al modello per l'elaborazione e successivamente inviare i risultati alla vista per la visualizzazione.
- **Flusso**: Dopo che l'utente inserisce i valori o le operazioni, il controller li riceve e li invia al **Model** (Manager) per elaborazione. Una volta completata l'elaborazione, il controller invia i risultati alla **View** per la visualizzazione.

3.4.3. View: Interfaccia grafica

- **Funzione**: La **View** è il componente che si occupa della rappresentazione grafica dell'applicazione. In questo caso, la **View** visualizza i risultati dell'elaborazione dei dati all'utente. La vista è ciò che l'utente vede e interagisce direttamente con l'interfaccia utente.
- **Flusso**: Dopo che il controller ha elaborato i dati e li ha inviati alla **View**, quest'ultima aggiorna l'interfaccia grafica per mostrare i risultati all'utente.

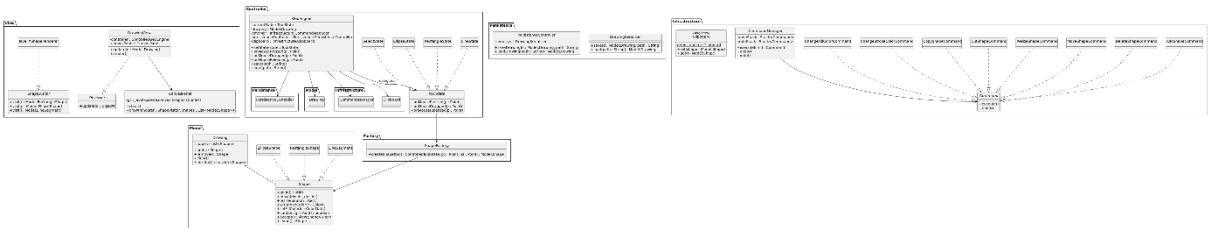
3.4.4. Utente

- **Funzione**: L'utente interagisce con il sistema tramite l'interfaccia grafica (View). L'utente inserisce i valori o le operazioni desiderate, che vengono poi inviati al controller per l'elaborazione.
- **Flusso**: L'utente inserisce un valore o un'operazione, e questi vengono inviati al controller, che li elabora attraverso il modello. Il flusso di dati continua quando il controller invia i risultati all'interfaccia grafica, che aggiorna la vista per mostrare i risultati all'utente.

3.4.5. Flussi di comunicazione nel diagramma

- **Fase 1**: L'utente inserisce un valore o un'operazione tramite l'interfaccia grafica (**View**). Questo input è poi inviato al **Controller** (GeoEngine).
- **Fase 2**: Il **Controller** riceve l'input dall'utente e lo invia al **Model** per l'elaborazione dei dati.
- **Fase 3**: Una volta che il **Model** ha elaborato i dati, i risultati vengono inviati al **Controller**, che poi aggiorna l'interfaccia grafica (**View**) per mostrare i risultati all'utente.

3.5. Class Diagram



Il **class diagram** definitivo della **prima Sprint** rappresenta l'architettura software scelta per il progetto GeoDraw.

3.5.1. Package **View**

La View è composta principalmente da tre elementi:

- **DrawingView**: implementa l'interfaccia Observer per reagire automaticamente ai cambiamenti del Model. Questo consente una netta separazione tra logica e presentazione visiva.
- **CanvasPanel**: rappresenta concretamente l'area grafica su cui vengono disegnate le forme. Contiene un riferimento diretto al contesto grafico (GraphicsContext) di JavaFX.
- **ShapeVisitor (Visitor Pattern)**: introdotto sin dalla progettazione iniziale per disaccoppiare il rendering grafico delle forme dalla logica del modello. La classe concreta JavaFXShapeRenderer implementa questo visitatore ed è responsabile della conversione dei dati delle forme in istruzioni di rendering JavaFX.

3.5.2. Package **Controller**

Il controller è rappresentato dal componente **GeoEngine**, il quale coordina le operazioni utente in modo centralizzato:

- Gestisce gli strumenti di disegno tramite un pattern **State**, delegando ogni evento di input (clic del mouse, trascinamento, rilascio) agli stati concreti (LineState, RectangleState, EllipseState, SelectState), evitando così complessi costrutti condizionali (if-else) e garantendo un'estensione semplice per futuri strumenti.
- Si occupa dell'esecuzione dei comandi tramite il **Command pattern** (ad esempio aggiunta, spostamento, modifica delle forme). I comandi sono gestiti dal CommandManager, che consente anche le future operazioni di undo e redo.
- Il salvataggio e caricamento sono delegati al **PersistenceController**, che utilizza la classe DrawingSerializer.
- Utilizza una clipboard (Singleton) per gestire operazioni di taglia e copia.

3.5.3. Package **Model**

Il modello racchiude le classi che rappresentano i dati e la logica interna delle forme geometriche:

- L'interfaccia **Shape** definisce le operazioni base comuni a tutte le forme (ad esempio spostamento, ridimensionamento, cambio colore). È stato deciso di includere un identificativo univoco (UUID) per facilitare la gestione e l'identificazione delle forme.
- Ogni classe concreta di forma (LineSegment, RectangleShape, EllipseShape) implementa Shape e adotta il pattern **Prototype**, implementando un metodo clone() per facilitare operazioni come copia/incolla.
- La classe Drawing contiene e gestisce una collezione di forme, fornendo metodi per aggiungere, rimuovere e iterare sugli elementi.

È importante notare che, fin dall'inizio, il rendering delle forme è stato previsto utilizzando il pattern **Visitor** (accept() nella classe Shape), mantenendo il modello totalmente indipendente dalla tecnologia di rendering e aumentando così la testabilità e modularità del codice.

3.5.4. Package **Persistence**

Contiene il componente responsabile della persistenza dei dati:

- **DrawingSerializer** fornisce metodi per salvare e caricare disegni da file esterni.
- **PersistenceController** è utilizzato dal controller per effettuare operazioni di salvataggio e caricamento, incapsulando così la logica di persistenza.

3.5.5. Package **Infrastructure**

Infrastruttura comune che supporta gli altri componenti, composta da:

- Il **Command pattern**, con classi concrete come AddShapeCommand, DeleteShapeCommand, MoveShapeCommand e altri, che supportano la futura operazione di undo/redo tramite l'implementazione del metodo undo().
- Il **CommandManager**, che mantiene stack separati per i comandi undo e redo.
- **Clipboard**: implementa il pattern **Singleton** per facilitare le operazioni di copia e taglia delle forme.

3.5.6. Package **Factory**

La factory class **ShapeFactory** è responsabile della creazione delle forme concrete (LineSegment, RectangleShape, EllipseShape), implementando il pattern **Factory Method** per disaccoppiare la creazione degli oggetti dalla loro gestione.

3.6. Design Pattern adottati

L'architettura presentata nel class diagram della Sprint 1 applica un insieme ben definito di **design pattern** per garantire una progettazione modulare, estendibile e manutenibile, in linea con i principi SOLID e con le buone pratiche dell'ingegneria del software.

- Al centro dell'architettura troviamo il pattern **MVC (Model–View–Controller)**, che separa nettamente la logica applicativa (Model), l'interfaccia utente (View) e il coordinamento degli eventi (Controller).

- Il **pattern Observer** è utilizzato per aggiornare automaticamente la DrawingView quando cambia lo stato del modello Drawing, eliminando ogni necessità di polling e mantenendo la sincronizzazione tra i componenti.

- La gestione degli strumenti di disegno è modellata tramite il **pattern State**, che consente di associare ad ogni tool (linea, rettangolo, ellisse, selezione) una classe concreta che incapsula il comportamento associato agli eventi mouse (onMousePressed, onMouseDragged, onMouseReleased). Questo approccio elimina la necessità di strutture condizionali complesse e rende l'estensione dei tool estremamente semplice.

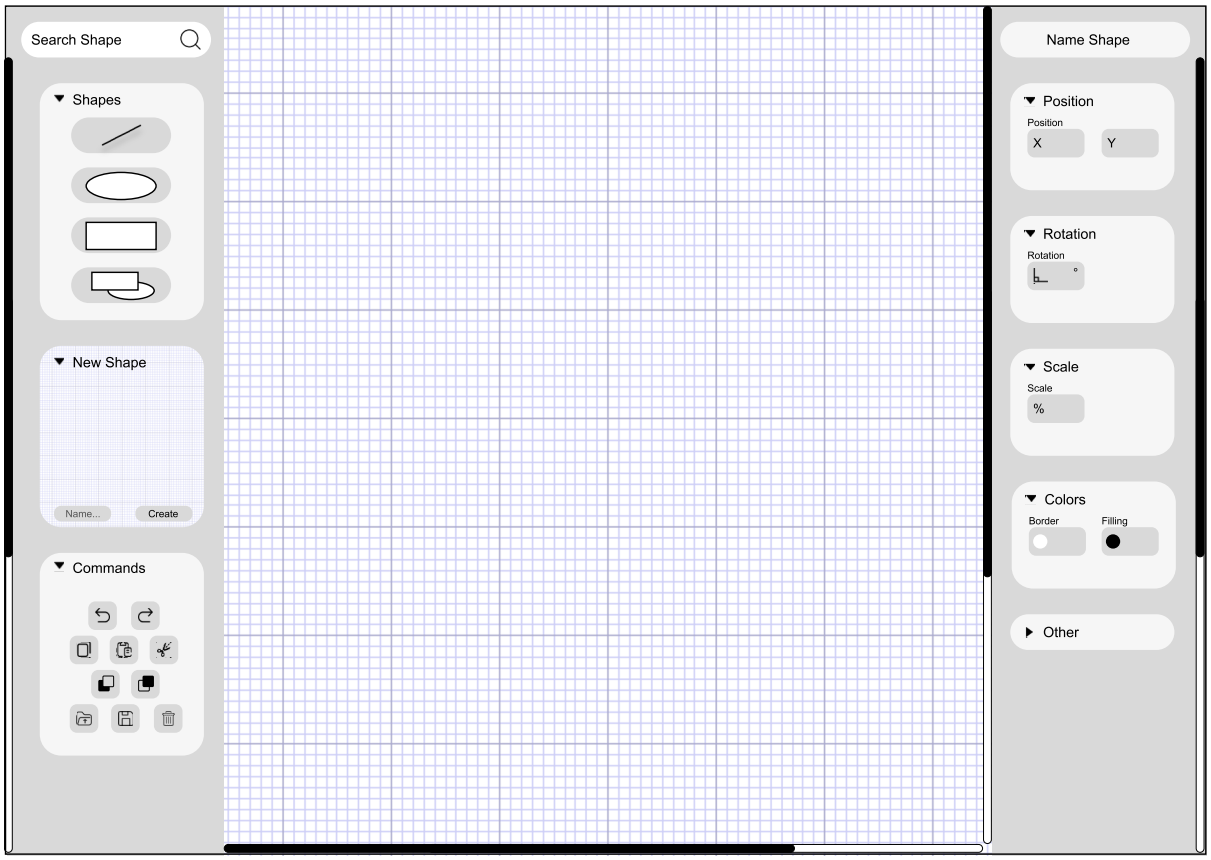
- Ogni operazione che modifica lo stato del disegno è incapsulata in un oggetto comando, secondo il **pattern Command**, il quale supporta anche le operazioni di undo/redo grazie all'introduzione del metodo undo() e alla presenza di un CommandManager dotato di due stack. I comandi implementano azioni atomiche come AddShapeCommand, MoveShapeCommand, ChangeStrokeColorCommand, ecc.

- Il **pattern Singleton** è applicato alla classe Clipboard, che fornisce un'istanza condivisa e centralizzata per gestire le operazioni di taglia/copia/incolla delle forme.

- La creazione delle forme avviene attraverso una **Factory Method** (ShapeFactory), che nasconde la logica di costruzione e consente di creare oggetti Shape concreti (es. RectangleShape) a partire dai parametri forniti dal controller.
- Il **pattern Prototype** è usato per implementare l'operazione clone() su ogni Shape, necessaria per la gestione di copie e duplicazioni, ma anche per clipboard e serializzazione.

- Il rendering delle forme è disaccoppiato dal modello tramite il **pattern Visitor**: ogni Shape implementa un metodo accept(ShapeVisitor), e la View (concretamente JavaFXShapeRenderer) fornisce le implementazioni visit(RectangleShape), visit(EllipseShape), ecc. Questo approccio consente di delegare completamente la logica grafica alla View, mantenendo il modello totalmente libero da riferimenti a JavaFX o ad altre librerie grafiche.

3.7. UI MockUp



L'interfaccia è divisa in tre sezioni principali:

1. **La Colonna di Sinistra (Barra degli Strumenti):**
   In cima c'è una barra **"Search Shape"** con un'icona a forma di lente d'ingrandimento, qui si possono cercare forme specifiche.
   Subito sotto, c'è un pannello espandibile chiamato **"Shapes"**. Qui vediamo delle forme predefinite: una linea, un'ellisse, un rettangolo e una forma più complessa, ossia un rettangolo con un ellisse attaccato sotto. L'idea è che puoi trascinare queste forme sul canvas.
   Poi c'è una sezione interessante chiamata **"New Shape"**. Anche questa è espandibile e al suo interno c'è una piccola griglia, un campo per inserire un nome e un pulsante "*Create*". Questo suggerisce all'utente la possibilità di disegnare una forma personalizzata su quella piccola griglia, nominarla e salvarla per un uso futuro.
   Infine, nella colonna di sinistra, c'è il pannello **"Commands"**. Qui ci sono icone abbastanza standard: annulla, ripeti, copia, incolla, taglia, porta avanti, porta indietro, apri file, salva file ed elimina.
2. **L'Area Centrale (Canvas):**
   Questa è la parte più grande dello schermo, ed è una canvas e come aiuto è stata inserita una griglia. È l'area dove l'utente posizionerà, disegnerà e modificherà le forme. La griglia aiuta con l'allineamento e il posizionamento preciso.

3. **La Colonna di Destra (Pannello delle Proprietà):**
   In alto c'è un campo **"Name Shape"**, che mostra il nome della forma attualmente selezionata e permette di rinominarla.
   Sotto, ci sono vari pannelli espandibili per modificare gli attributi della forma selezionata:
   - **"Position"**: con campi per le coordinate X e Y.
   - **"Rotation"**: con un campo per inserire un angolo di rotazione in gradi.
   - **"Scale"**: con un campo per la percentuale di ridimensionamento.
   - **"Colors"**: con un selettore di colore.
   - **"Other"**: un pannello chiuso presente solo a scopo grafico, per comprendere che sono pannelli espandibili.

3.8. Strumenti Principali

Trello Board	**https://trello.com/gruppo-11-board**
Word	**https://unisalerno-my.sharepoint.com/Soft_Gruppo 11.docx**
GitHub	**https://github.com/Soft_Gruppo11**
Visual Studio Code	

3.9. Tecnologie Utilizzate

Il progetto sarà sviluppato principalmente in Java, scelto per la sua robustezza, portabilità e ampia disponibilità di librerie. Per l'implementazione dell'interfaccia grafica utente utilizzeremo JavaFX, un framework moderno e modulare che consente di realizzare GUI reattive e ben integrate con la logica applicativa.
A supporto dello sviluppo verranno utilizzati i seguenti strumenti e tecnologie:

- **Git** e **GitHub** per il versionamento del codice e la collaborazione in team Maven per la gestione delle dipendenze e l'automazione del build.
- **Trello** per l'organizzazione del Product Backlog e il tracking degli sprint
- **PlantUML** per la modellazione dell'architettura e dei diagrammi
- **Microsoft Word** per la documentazione collaborativa
- **Figma** per la realizzazione di mockup dell'interfaccia utente
- **JUnit** per l'implementazione dei test unitari nel corso dello sviluppo.

FASE 1: First Sprint Delivery

Stima iniziale della velocità del progetto:
Durante la fase di pre-game, la stima della velocity del team è risultata essere pari a 36 Story Points, calcolata sulla base delle ore settimanali effettivamente disponibili e dell'effort medio effettivo per ciascuna User Story. Il primo sprint è stato pianificato con un carico di lavoro di 29 SP, così da garantire un margine di sicurezza adeguato per gestire eventuali imprevisti, attività di testing o rifiniture. Tuttavia, nel corso dell'analisi architetturale e delle prime fasi di implementazione, sono emerse alcune criticità strutturali che hanno reso necessario l'inserimento di 4 nuove User Story fondamentali per garantire la solidità del progetto e l'aderenza ai principi architetturali definiti (come l'indipendenza del Model da JavaFX).

Questa integrazione ha portato il totale degli SP dello sprint a 34, riducendo sensibilmente il margine inizialmente previsto. Di fronte a questa situazione, il team ha preferito destinare lo sforzo residuo a un'esecuzione più accurata delle attività previste, privilegiando la qualità del codice, la coerenza architetturale e una gestione più attenta dei dettagli implementativi. Questa scelta ha permesso di mantenere alta l'affidabilità del progetto, pur lavorando al limite della capacità stimata.

4. User stories sprint

USER STORY: 1 – Apertura Finestra
Story Point: 1

ID TASK: 1 – Creare la finestra principale con canvas
(*Stima: 20 min*)

Realizzare la classe `MainApp.java` con una finestra (`Stage`) contenente un layout JavaFX (`BorderPane`) e un canvas inizialmente vuoto. Il canvas deve essere integrato nella struttura dell'interfaccia seguendo il pattern MVC e predisposto all'interazione futura.

ID TASK: 2 – Inizializzare GeoEngine e canvas all'avvio
(*Stima: 10 min*)

Inizializzare il `GeoEngine` all'avvio dell'applicazione. Collegare il canvas al controller principale e verificare che `GeoEngine.getDrawing().isEmpty()` restituisca `true`, confermando l'assenza di figure nel modello.

ID TASK: 3 – Verifica visiva e test automatico di avvio corretto
(*Stima: 10 min*)

Verificare manualmente e tramite test automatico che la finestra venga avviata correttamente, che il canvas sia visualizzato, vuoto, e senza errori. Usare TestFX per l'inizializzazione GUI nei test.

USER STORY: 2 – Aggiunta Forma

Story Point: 3

ID TASK: 4 – Implementare la selezione dello strumento di disegno (*Stima: 30 min*)

Consentire all'utente di selezionare il tipo di forma (segmento, rettangolo, ellisse) tramite pulsanti della toolbar. Attivare lo stato corrispondente in `GeoEngine` e aggiornare graficamente il pulsante attivo.

ID TASK: 5 – Gestire il disegno della forma su due click (*Stima: 45 min*)

Gestire nel controller (`ToolState`) la pressione del mouse come punto iniziale della forma e il rilascio come punto finale. Creare la forma con i dati di posizione e colori da `GeoEngine`, e aggiungerla al `Drawing` tramite `AddShapeCommand`.

ID TASK: 6 – Aggiornare la View per il rendering della nuova forma (*Stima: 30 min*)

Notificare correttamente la View affinché disegni la nuova forma nel canvas. Assicurarsi che il rendering sia coerente con il Model e che il sistema MVC venga rispettato.

ID TASK: 7 – Scrivere test di integrazione per verifica comportamento (*Stima: 15 min*)

Realizzare test automatici per verificare che la selezione dello strumento sia corretta, che la forma venga aggiunta al modello e che appaia nel canvas come previsto.

USER STORY: 3 – Selettore strumenti di disegno base

Story Point: 1

ID TASK: 8 – Creare la barra degli strumenti con i pulsanti delle forme base (*Stima: 20 min*)

Implementare nella View la barra degli strumenti contenente i pulsanti per disegnare segmento, rettangolo ed ellisse. Posizionare i pulsanti nel layout e assegnare un evento di selezione che attivi il relativo stato (`ToolState`) nel `GeoEngine`.

ID TASK: 9 – Gestire la selezione dello strumento e lo stato attivo (*Stima: 10 min*)

Gestire visivamente la selezione dello strumento, evidenziando il pulsante attivo e disattivando gli altri. Verificare che il controller `GeoEngine` aggiorni correttamente lo stato attuale in base alla scelta dell'utente.

ID TASK: 10 – Scrivere test per la selezione e l'attivazione dello strumento (*Stima: 10 min*)

Realizzare test automatici che verifichino il corretto aggiornamento dello stato attivo in `GeoEngine` al clic sui pulsanti della toolbar, e che l'interfaccia rifletta visivamente la selezione corrente dello strumento.

USER STORY: 4 – Disegnare segmenti di linea

Story Point: 2

ID TASK: 11 – Implementare il comportamento del LineState (*Stima: 30 min*)

Gestire nel controller `LineState` della cattura della pressione del mouse come punto iniziale della forma e il rilascio come punto finale, generando un oggetto `LineSegment` con i due estremi, lo configura con i colori da `GeoEngine`, e lo aggiunge al modello tramite `AddShapeCommand`.

ID TASK: 12 – Disegnare il segmento nella View (*Stima: 25 min*)

Implementare la logica di rendering all'interno della classe `ShapeRenderer`, assicurandosi che il segmento venga tracciato graficamente tra i due punti nel canvas. La logica di disegno deve restare isolata nella View, in linea con il pattern MVC.

ID TASK: 13 – Aggiornare GeoEngine e notificare il disegno (*Stima: 15 min*)

Verificare che, al termine dell'aggiunta del segmento, `GeoEngine` notifichi correttamente le viste osservatrici, causando il ridisegno del canvas con la nuova forma. Controllare che venga utilizzato il meccanismo di notifica standard del progetto.

ID TASK: 14 – Scrivere test per disegno segmento e aggiornamento View (*Stima: 10 min*)

Scrivere test automatici per validare il corretto comportamento di `LineState`, il salvataggio dei punti, la creazione del segmento, e la presenza del rendering nel canvas. Verificare anche che il comando venga registrato nello `UndoManager`.

USER STORY: 5 – Disegnare rettangoli

Story Point: 2

ID TASK: 15 – Implementare il comportamento del RectangleState (*Stima: 30 min*)

Gestire nel controller `RectangleState` della cattura della pressione del mouse come punto iniziale della forma e il rilascio come punto finale, creando un oggetto `RectangleShape` e aggiungerlo al modello tramite `AddShapeCommand`.

ID TASK: 16 – Disegnare graficamente il rettangolo nella View (*Stima: 25 min*)

Estendere la classe `ShapeRenderer` per includere il rendering del rettangolo. Il rettangolo deve essere disegnato nel canvas tra i due punti estremi con colore di bordo predefinito (nero).

ID TASK: 17 – Notificare il disegno e aggiornare la vista (*Stima: 15 min*)

Assicurarsi che `GeoEngine` notifichi la modifica del modello dopo l'aggiunta del rettangolo, in modo che la View venga aggiornata e ridisegni automaticamente il canvas con la nuova figura.

ID TASK: 18 – Scrivere test per disegno rettangolo e corretta integrazione (*Stima: 10 min*)

Scrivere test automatici per verificare il comportamento del `RectangleState`, la creazione corretta del rettangolo, l'uso dei colori e il rendering visivo. Verificare inoltre che la figura venga aggiunta alla lista del disegno e che il comando sia registrato per l'annullamento.

USER STORY: 6 – Disegnare ellissi

Story Point: 2

ID TASK: 19 – Gestire il disegno dell'ellisse (*Stima: 30 min*)

Implementare nella classe `EllipseState` la gestione della pressione del mouse come punto iniziale e del rilascio come punto finale del bounding box. Creare un oggetto `EllipseShape` con i dati raccolti, configurarlo con i colori di bordo e riempimento attuali da `GeoEngine`, e inserirlo nel modello attraverso `AddShapeCommand`.

ID TASK: 20 – Disegnare l'ellisse nel canvas tramite ShapeRenderer (*Stima: 25 min*)

Estendere la classe `ShapeRenderer` per disegnare graficamente l'ellisse all'interno del canvas, utilizzando il rettangolo di delimitazione fornito dall'oggetto `EllipseShape`. Applicare correttamente il colore di bordo e il riempimento, rispettando il pattern MVC.

ID TASK: 21 – Notificare l'aggiornamento della View (*Stima: 15 min*)

Verificare che `GeoEngine` notifichi correttamente la View dopo l'aggiunta dell'ellisse al modello, provocando il ridisegno del canvas e la visualizzazione della nuova forma.

ID TASK: 22 – Scrivere test per il disegno corretto dell'ellisse (*Stima: 10 min*)

Creare test automatici per validare il comportamento di `EllipseState`, dalla gestione degli eventi mouse alla corretta creazione e visualizzazione della forma. Verificare anche l'inserimento nel modello e l'utilizzo del comando per la gestione dell'annullamento.

USER STORY: 7 – Cambio colore bordo

Story Point: 2

ID TASK: 23 – Aggiungere il selettore colore per il bordo nella View (*Stima: 30 min*)

Inserire nella GUI un componente `ColorPicker` per la selezione del colore di bordo. Collegarlo al controller affinché notifichi il cambiamento ogni volta che l'utente seleziona un nuovo colore.

ID TASK: 24 – Applicare il colore di bordo alla forma selezionata tramite comando
(*Stima: 25 min*)

Implementare la logica per aggiornare il colore di bordo della forma attualmente selezionata nel modello. Utilizzare `ChangeStrokeColorCommand` per registrare la modifica.

ID TASK: 25 – Aggiornare la View dopo il cambiamento del bordo
(*Stima: 15 min*)

Notificare correttamente la View al termine del comando di cambio colore. Assicurarsi che la forma venga ridisegnata nel canvas con il nuovo colore di bordo e che il selettore rifletta sempre il colore attuale della forma selezionata.

ID TASK: 26 – Scrivere test per la modifica del colore del bordo
(*Stima: 10 min*)

Scrivere test automatici per verificare che il cambiamento del colore venga applicato correttamente alla forma selezionata e che venga gestito tramite comando. Includere anche il caso in cui nessuna forma è selezionata e viene aggiornato solo il colore di default.

USER STORY: 8 – Colore interno figura selezionata
Story Point: 2

ID TASK: 27 – Aggiungere il selettore colore per il riempimento nella View
(*Stima: 30 min*)

Integrare nella GUI un componente `ColorPicker` dedicato al colore di riempimento. Collegarlo al controller `GeoEngine` e disabilitarlo dinamicamente se la forma selezionata non supporta il riempimento (es. segmenti).

ID TASK: 28 – Applicare il colore interno alla forma selezionata tramite comando
(*Stima: 25 min*)

Implementare nel controller la logica per aggiornare il colore interno della forma selezionata nel modello, tramite `ChangeFillColorCommand`.

ID TASK: 29 – Notificare la View e aggiornare il rendering della forma
(*Stima: 15 min*)

Dopo l'esecuzione del comando, notificare correttamente la View affinché ridisegni la forma con il nuovo colore di riempimento. Aggiornare anche il selettore per riflettere lo stato corrente della selezione.

ID TASK: 30 – Scrivere test per cambio colore interno e casi limite
(*Stima: 10 min*)

Realizzare test automatici che verifichino il corretto funzionamento del cambio colore interno per forme compatibili, il salvataggio del colore nel modello e l'aggiornamento della vista.

USER STORY: 9 – Salvataggio disegno

Story Point: 2

ID TASK: 31 – Serializzare tutte le forme presenti nel disegno
(*Stima: 30 min*)

Implementare in `GeoEngine` un metodo per convertire tutte le forme attualmente presenti nel modello (`Drawing`) in una rappresentazione serializzabile. Includere tutte le proprietà rilevanti come tipo di forma, coordinate, colori di bordo e riempimento.

ID TASK: 32 – Gestire l'evento di salvataggio dalla GUI
(*Stima: 15 min*)

Intercettare il click dell'utente sul pulsante "Salva" nella GUI. Aprire un `FileChooser`, ottenere il percorso di destinazione e invocare la logica di esportazione serializzata dal `GeoEngine`, gestendo eventuali eccezioni e mostrando feedback all'utente.

ID TASK: 33 – Scrivere fisicamente il file sul disco
(*Stima: 15 min*)

Scrivere il contenuto serializzato su file nel formato .ser, assicurandosi che il file venga salvato localmente nella destinazione scelta dall'utente e che il contenuto sia leggibile in un secondo momento per il caricamento.

ID TASK: 34 – Scrivere test per verifica del salvataggio corretto
(*Stima: 12 min*)

Creare test automatici per verificare che la serializzazione produca un file coerente con il contenuto del modello, e che il file possa essere effettivamente generato e salvato. Simulare anche casi d'errore come file non scrivibile o path non valido.

USER STORY: 10 – Caricamento disegno

Story Point: 2

ID TASK: 35 – Gestire la selezione del file da caricare tramite GUI
(*Stima: 15 min*)

Implementare nella GUI la gestione del click sull'icona "Carica". Mostrare un `FileChooser` che consenta all'utente di selezionare un file .ser e inviarne il percorso a `GeoEngine` per l'importazione. Gestire eventuali annullamenti o path null.

ID TASK: 36 – Eseguire il parsing del file e validare il contenuto
(*Stima: 30 min*)

Leggere il file selezionato, deserializzare gli oggetti in esso contenuti e verificarne la validità. In caso di file malformato, incompatibile o contenente dati incompleti, bloccare il caricamento e mostrare un messaggio di errore all'utente.

ID TASK: 37 – Aggiornare il modello con le nuove forme e ridisegnare il canvas
(*Stima: 15 min*)

Dopo un parsing valido, svuotare l'attuale contenuto del `Drawing` e sostituirlo con le forme caricate. Notificare la View in modo che venga effettuato un ridisegno completo del canvas con le nuove forme importate.

ID TASK: 38 – Scrivere test per il caricamento da file
(*Stima: 12 min*)

Realizzare test automatici per verificare la corretta lettura di file validi, la gestione dei casi di errore (file non leggibile o corrotto).

USER STORY: 11 – Eliminazione forma

Story Point: 2

ID TASK: 39 – Rilevare la pressione del tasto "Canc" su una forma selezionata
(*Stima: 30 min*)

Implementare nella GUI la gestione dell'evento di pressione del tasto "Canc". Se una forma è attualmente selezionata nel canvas, inviare un comando al controller (`GeoEngine`) per attivare la cancellazione. Se nessuna forma è selezionata, ignorare l'evento.

ID TASK: 40 – Rimuovere la forma selezionata dal modello tramite comando
(*Stima: 25 min*)

Implementare `DeleteShapeCommand` per rimuovere la forma selezionata dal `Drawing`.

ID TASK: 41 – Aggiornare la View e sincronizzare lo stato della selezione
(*Stima: 15 min*)

Dopo la rimozione della forma dal modello, notificare la View per aggiornare il canvas, rimuovere graficamente la forma e azzerare lo stato di selezione. Verificare che il selettore e i pannelli relativi ai colori siano disattivati o ripristinati.

ID TASK: 42 – Scrivere test automatici per l'eliminazione forma
(*Stima: 10 min*)

Scrivere test per verificare che la pressione del tasto "Canc" su una forma selezionata comporti la sua rimozione effettiva dal modello e il corretto aggiornamento della View.

USER STORY: 12 – Spostamento forma

Story Point: 2

ID TASK: 43 – Gestire evento di trascinamento per forma selezionata
(*Stima: 25 min*)

Nel Controller o nella GUI, intercettare il mouse pressed su una forma selezionata e registrare la posizione iniziale. Gestire l'inizio del trascinamento.

ID TASK: 44 – Calcolare lo spostamento e aggiornare la posizione della forma
(*Stima: 30 min*)

Dopo l'evento mouse dragged, calcolare delta x e delta y rispetto alla posizione iniziale e aggiornare le coordinate della forma selezionata nel Model.

ID TASK: 45 – Notificare la View e aggiornare il canvas in tempo reale
(*Stima: 15 min*)

Durante il trascinamento, notificare costantemente la View per aggiornare la rappresentazione visiva della forma spostata, seguendo il mouse.

ID TASK: 46 – Scrivere test per lo spostamento della forma
(*Stima: 10 min*)

Scrivere test automatici per verificare che una forma venga spostata correttamente con il mouse.

USER STORY: 13 – Ridimensionamento forma

Story Point: 2

ID TASK: 47 – Implementare maniglie di ridimensionamento nella GUI
(*Stima: 25 min*)

Visualizzare maniglie o punti interattivi attorno alla forma selezionata per consentire l'interazione diretta con il mouse.

ID TASK: 48 – Gestire evento di trascinamento sulle maniglie
(*Stima: 30 min*)

Rilevare il trascinamento su una maniglia e calcolare le nuove dimensioni della forma in base alla direzione e alla distanza del trascinamento.

ID TASK: 49 – Sostituire la forma nel Model con una nuova ridimensionata
(*Stima: 15 min*)

Rimuovere la forma esistente dal Model e crearne una nuova con le dimensioni aggiornate.

ID TASK: 50 – Scrivere test per il ridimensionamento
(*Stima: 10 min*)

Verificare tramite test che il ridimensionamento produca la forma corretta con le nuove dimensioni e che il comportamento della GUI sia reattivo e corretto.

USER STORY: 14 – Taglia forma

Story Point: 2

ID TASK: 51 – Gestire la selezione della forma e il click sull'icona "Taglia"
(*Stima: 25 min*)

Implementare il comportamento del pulsante "Taglia" nella GUI. Verificare che una forma sia effettivamente selezionata, e se presente, iniziare l'operazione di taglio. Collegare l'evento al controller `GeoEngine` per avviare l'azione.

ID TASK: 52 – Rimuovere la forma selezionata e salvarla negli appunti interni
(*Stima: 30 min*)

Realizzare un comando `CutShapeCommand` che rimuova la forma selezionata dal `Drawing` e la salvi nel Clipboard interno. La forma tagliata deve essere clonata e resa pronta per un eventuale incolla, senza essere definitivamente eliminata.

ID TASK: 53 – Notificare la View della modifica e aggiornare il canvas
(*Stima: 15 min*)

Dopo l'esecuzione del comando, notificare la View per aggiornare il canvas, assicurandosi che la forma tagliata venga effettivamente rimossa dalla visualizzazione. Aggiornare la selezione e eventuali proprietà mostrate nella GUI.

ID TASK: 54 – Scrivere test per la funzionalità di taglio
(*Stima: 10 min*)

Scrivere test automatici per verificare che una forma selezionata possa essere tagliata correttamente, che venga rimossa dal disegno, memorizzata nel Clipboard.

USER STORY: 15 – Copia forma
Story Point: 2

ID TASK: 55 – Implementare la selezione della forma e il rilevamento del comando di copia
(*Stima: 30 min*)

Verificare che l'utente abbia selezionato una forma nel canvas e intercettare la pressione della combinazione da tastiera o il clic sull'icona "Copia" nella GUI. Integrare la logica nel controller (`GeoEngine`) e abilitare il comportamento solo in presenza di una selezione attiva.

ID TASK: 56 – Clonare la forma selezionata e copiarla negli appunti interni
(*Stima: 25 min*)

Implementare il metodo `cloneWithNewId()` per ogni tipo di forma, in modo da creare una copia indipendente della forma selezionata. Memorizzare questa copia nel Clipboard interno di `GeoEngine`, pronta per una futura operazione di incolla.

ID TASK: 57 – Aggiornare il sistema di feedback utente sulla copia
(*Stima: 15 min*)

Fornire un riscontro visivo o testuale all'utente quando la copia viene effettuata con successo. Ad esempio, stampare un messaggio nella console o mostrare una notifica temporanea nella GUI, indicando che la forma è stata copiata negli appunti.

ID TASK: 58 – Scrivere test per la funzionalità di copia forma
(*Stima: 10 min*)

Creare test automatici per verificare che solo le forme selezionate possano essere copiate, che la copia venga effettivamente memorizzata negli appunti, e che il clone non sia legato all'istanza originale ma sia una copia autonoma e correttamente identificabile.

USER STORY: 16 – (35*) Separazione del rendering dalla logica del modello

Story Point: 2

ID TASK: 59 – Rimuovere i riferimenti a JavaFX dal package model (*Stima: 25 min*)

Identificare tutte le dipendenze da JavaFX (es. `GraphicsContext`) nelle classi del Model (es. `Shape`) ed eliminarle.

ID TASK: 60 – Creare interfaccia `ShapeRenderer` nel package View o Adapter (*Stima: 30 min*)

Definire una classe o interfaccia `ShapeRenderer` in un package della View responsabile del rendering, con metodi per disegnare le diverse forme.

ID TASK: 61 – Implementare il rendering delegato per ogni forma nel renderer (*Stima: 15 min*)

Per ogni tipo di forma (es. `Rectangle`), implementare i metodi di rendering all'interno di `ShapeRenderer`, usando `GraphicsContext`.

ID TASK: 62 – Scrivere test e verificare separazione MVC (*Stima: 10 min*)

Verificare tramite test o analisi statica che il Model non importi più classi JavaFX. Testare che la View renda correttamente le forme tramite il renderer.

USER STORY: 17 – (36*) Sostituzione di Point2D con Point Custom

Story Point: 5

ID TASK: 63 – Identificare tutte le classi e i metodi che usano Point2D (*Stima: 40 min*)

Effettuare una scansione completa del progetto (in particolare dei pacchetti Model, Controller e Infrastructure) per individuare tutte le classi, metodi, costruttori e variabili che dipendono da Point2D. Preparare una lista completa per guidare la sostituzione con il tipo custom Point.

ID TASK: 64 – Sostituire Point2D con Point nel dominio (*Stima: 60 min*)

Aggiornare tutte le classi del modello (es. `Rect`, `Shape`, `EllipseShape`) per utilizzare Point al posto di Point2D. Aggiornare costruttori, metodi, equals/hashCode, mantenendo compatibilità e integrità del comportamento.

ID TASK: 65 – Aggiornare il controller e gli stati interattivi (ToolState) (*Stima: 40 min*)

Adattare le classi del controller (come `LineState`, `RectangleState`, `EllipseState`) per usare il nuovo tipo Point in tutti i metodi relativi a eventi mouse (press, drag, release). Includere conversioni se necessario e garantire il corretto flusso di dati tra View e Model.

ID TASK: 66 – Aggiornare i comandi e la serializzazione
(*Stima: 30 min*)

Adeguare le classi Command
(es. `AddShapeCommand`, `ResizeShapeCommand`, `MoveShapeCommand`) e il meccanismo di serializzazione per supportare Point. Verificare che tutte le trasformazioni di coordinate siano coerenti e persistibili.

ID TASK: 67 – Scrivere test per validare la migrazione a Point
(*Stima: 30 min*)

Scrivere test automatici per verificare che tutte le operazioni basate su coordinate (disegno, spostamento, ridimensionamento, hit detection) funzionino correttamente con il nuovo tipo `Point`.

USER STORY: 18 – (37*) Rimozione di Color dal modello
Story Point: 2

ID TASK: 68 – Creare la classe ColorData nel modello
(*Stima: 20 min*)

Definire una classe `ColorData` all'interno del package Model, contenente i campi r, g, b e α.
Fornire metodi costruttori, getter, `equals()`, `hashCode()`, e metodi di conversione da/verso `javafx.scene.paint.Color` (usati solo nella View/Controller).

ID TASK: 69 – Rimuovere javafx.scene.paint.Color dal dominio
(*Stima: 35 min*)

Eliminare tutte le occorrenze della classe Color di JavaFX dal Model (in particolare da classi come `Shape`, `EllipseShape`, `RectangleShape`). Sostituire le proprietà di colore con istanze della nuova `ColorData`. Aggiornare tutti i metodi, costruttori e strutture che utilizzavano Color.

ID TASK: 70 – Adeguare GeoEngine e le conversioni colore nella View
(*Stima: 15 min*)

Aggiungere metodi ausiliari nella View e in `GeoEngine` per convertire `ColorData` in Color, limitando l'utilizzo della libreria JavaFX esclusivamente alla parte grafica. Assicurarsi che le operazioni di colore restino compatibili con `ChangeStrokeColorCommand` e `ChangeFillColorCommand`.

ID TASK: 71 – Scrivere test per verificare la sostituzione di Color
(*Stima: 10 min*)

Realizzare test automatici per verificare che la nuova `ColorData` sia usata correttamente al posto di Color e che tutte le forme mantengano il colore corretto durante creazione, modifica e rendering. Includere test di conversione tra Color e `ColorData`.

USER STORY: 19 – (38*) Rimozione delle classi JavaFx (Rectangle,Ellipse,etc.) dal Model

Story Point: 3

ID TASK: 72 – Rilevare tutte le dipendenze JavaFX residue nel Model (*Stima: 30 min*)

Eseguire un'analisi completa del package Model per identificare ogni importazione e utilizzo di classi JavaFX residue (es. `Rectangle`, `Ellipse`, `Color`, `Point2D`, `GraphicsContext`). Mappare le sostituzioni necessarie per ciascun tipo.

ID TASK: 73 – Rimuovere definitivamente le classi JavaFX dal dominio (*Stima: 50 min*)

Eliminare ogni riferimento a classi JavaFX nelle classi di dominio, sostituendole con strutture dati custom come `RectData`, `EllipseData`, `ColorData`, `Point`. Completare la transizione già iniziata nelle user story precedenti, garantendo un dominio completamente agnostico rispetto alla GUI.

ID TASK: 74 – Adeguare la View e il controller per adattarsi al nuovo Model (*Stima: 25 min*)

Modificare la View (`ShapeRenderer`) e il controller (`GeoEngine`, `ToolState`) affinché convertano correttamente i tipi interni del modello in oggetti JavaFX temporanei per il rendering (es. conversione `ColorData → Color`, `Point → double`).

ID TASK: 75 – Scrivere test per validare l'assenza completa di JavaFX nel Model (*Stima: 15 min*)

Scrivere test automatici e statici per verificare che nessuna classe del Model importi più pacchetti JavaFX. Includere test funzionali che garantiscano il corretto funzionamento delle operazioni (creazione, disegno, modifica) con i nuovi tipi interni.

4.1. Sviluppo del primo sprint

Ad ogni membro sono stati assegnati dei task

	Ferrara Francesco	Luongo Leonardo	Monda Francesco	Parente Alessia
Tasks assegnati e completati	TASK 6 TASK 12 TASK 13 TASK 17 TASK 21 TASK 30 TASK 31 TASK 33 TASK 37 TASK 40 TASK 49 TASK 52 TASK 56 TASK 63 TASK 64 TASK 68	TASK 3 TASK 7 TASK 10 TASK 14 TASK 18 TASK 22 TASK 26 TASK 34 TASK 38 TASK 39 TASK 42 TASK 46 TASK 50 TASK 54 TASK 58 TASK 62 TASK 67 TASK 71 TASK 72 TASK 75	TASK 1 TASK 8 TASK 9 TASK 16 TASK 20 TASK 23 TASK 25 TASK 27 TASK 29 TASK 32 TASK 35 TASK 41 TASK 45 TASK 47 TASK 53 TASK 57 TASK 60 TASK 61 TASK 70 TASK 74	TASK 2 TASK 4 TASK 5 TASK 11 TASK 15 TASK 19 TASK 24 TASK 28 TASK 36 TASK 43 TASK 44 TASK 48 TASK 51 TASK 55 TASK 59 TASK 65 TASK 66 TASK 69 TASK 73
Tasks assegnati e non completati				
Tasks ancora non assegnati				

4.1.1. Sprint release

Lo Sprint 1 si è concluso con il rilascio di una versione iniziale funzionante dell'applicazione GeoDraw, che include le funzionalità fondamentali per la creazione e manipolazione di figure geometriche base. L'obiettivo principale di questo sprint era impostare l'architettura dell'applicazione, implementare l'infrastruttura base del disegno e permettere all'utente di interagire con l'interfaccia grafica.

Funzionalità rilasciate:

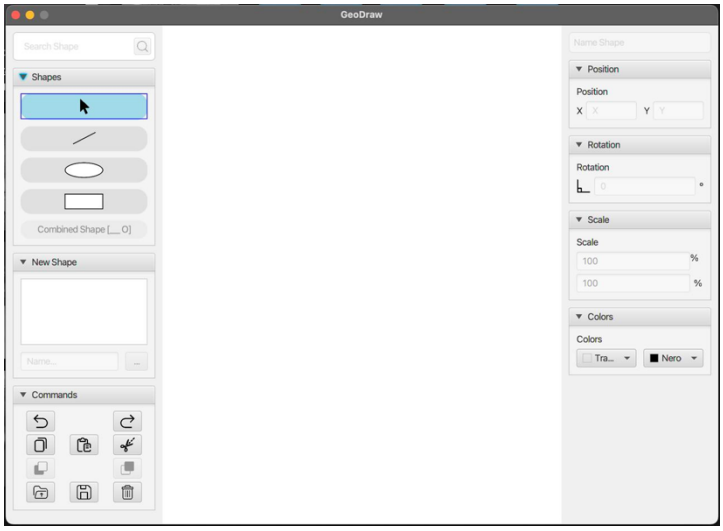
Le funzionalità rilasciate in questa Sprint sono le seguenti:

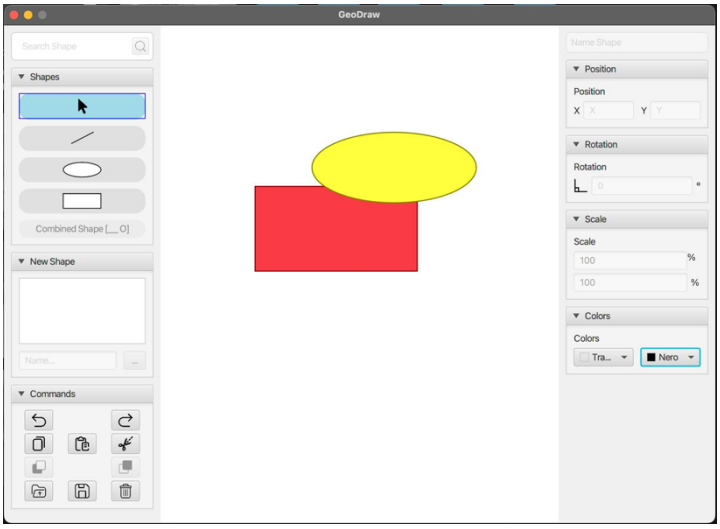
- Struttura MVC implementata con Observer per l'aggiornamento delle viste.
- Creazione di nuove figure geometriche (linea, rettangolo, ellisse) tramite barra degli strumenti;
- Visualizzazione delle figure nel canvas;
- Selezione di una figura con il mouse;
- Cancellazione di una figura selezionata.

Design Pattern applicati:

- ***Singleton*** per il controller;
- ***Factory Method*** per la creazione delle figure;
- ***Command*** per l'annullamento delle operazioni base;
- ***Strategy*** per il disegno.

4.1.2. Anteprima base





La funzionalità di anteprima base rappresenta uno dei primi risultati tangibili dello sviluppo dell'applicazione.
Le due schermate mostrano rispettivamente lo stato iniziale dell'applicativo (a sinistra) e una prima interazione utente (a destra), in cui è stato possibile disegnare e personalizzare forme geometriche.
Nella prima schermata, si evidenzia la corretta inizializzazione dell'interfaccia grafica: il canvas è vuoto, in accordo con i criteri di accettazione della User Story 1 ("Apertura finestra"), e tutti i componenti dell'interfaccia sono caricati correttamente (tool bar, pannello comandi, proprietà).
La seconda schermata mostra il risultato delle seguenti interazioni:

- Selezione dello strumento "Rettangolo" dalla toolbar laterale;
- Creazione interattiva della forma tramite click e trascinamento;
- Selezione dello strumento "Ellisse" e creazione di una seconda figura sovrapposta;
- Personalizzazione dei colori di bordo e riempimento tramite il pannello delle proprietà.

4.1.3. Update product backlog

Durante lo sviluppo del progetto, sono emerse alcune criticità di natura architetturale che hanno reso necessario un intervento strutturale sul design del software. In particolare, si è evidenziata una forte dipendenza tra il Model e JavaFX, in contrasto con i principi dell'architettura MVC. Per affrontare questi problemi e garantire un'adeguata separazione delle responsabilità tra Model, View e Controller, il team ha introdotto 4 nuove User Story. Queste storie aggiuntive hanno l'obiettivo di disaccoppiare il dominio dalla View e migliorare la manutenibilità e testabilità del codice.

35. Separazione del rendering dalla logica del modello

Come sviluppatore, voglio che il metodo draw(GraphicContent) sia spostato fuori dalla classe shape, per rispettare il pattern MVC ed evitare dipendenze tra il Model e JavaFX.

AC1: Dato che sviluppo il Model,
Quando implemento le classi delle forme,
Allora il package model è privo riferimenti a java.fx

AC2: Dato che disegno una forma,
Quando viene invocato il rendering,
Allora la View gestisce il disegno tramite una classe ShapeRender

AC3: Dato che lavoro sul Controller o sul Model,
Quando gestisco le operazioni sulle forme,
Allora né il Model né il Controller conoscono GraphicsContent.

Priorità: Alta (5)
Story points: 2
Ruolo: Sviluppatore

36. Sostituzione di Point2D con Point custom

Come sviluppatore, voglio utilizzare una classe Point autonoma nel dominio al posto di javafx.geometry.point2d, per rendere il Model indipendente da JavaFX.

Criteri di accettazione

AC1: Dato che implemento una forma nel Model,
Quando definisco i punti,
Allora il Model utilizza una classe Point al posto di utilizzare Point2D di JavaFX.

AC2: Dato che descrivo una forma nel dominio,
Quando uso coordinate,
Allora utilizzo una classe Point con campi x e y.

AC3: Dato che la View riceve dati dal Model,
Quando deve disegnare,
Allora la View converte Point nel formato compatibile con JavaFX.

Priorità: Alta (5)
Story points: 1
Ruolo: Sviluppatore

37. Rimozione di Color dal modello

Come sviluppatore, voglio sostituire javafx.scene.paint.color con una struttura ColorData, per disaccoppiare il dominio dalla View.

Criteri di accettazione

AC1: Dato che implemento il Model,
Quando assegno un colore a una forma,
Allora il Model utilizza la classe ColorData al posto di utilizzare Color di JavaFX.

AC2: Dato che definisco un colore nel dominio,
Quando rappresento una tinta,
Allora utilizzo una classe ColorData con campi r, g, b e alpha.

AC3: Dato che la View riceve un ColorData,
Quando esegue il rendering,
Allora la View converte ColorData in un oggetto Color JavaFX.

Priorità: Alta (5)
Story points: 1
Ruolo: Sviluppatore

38. Rimozione delle classi JavaFX (Rectangle, Ellipse, etc.) dal Model

Come sviluppatore, voglio che le classi RectangleShape, EllipseShape siano implementate senza l'utilizzo di classi JavaFX, per mantenere il Model autonomo e testabile.

Criteri di accettazione

AC1: Dato che definisco una forma nel dominio,
Quando scrivo la sua classe,
Allora le classi delle forme siano implementate senza l'utilizzo di javafx.scene.shape.

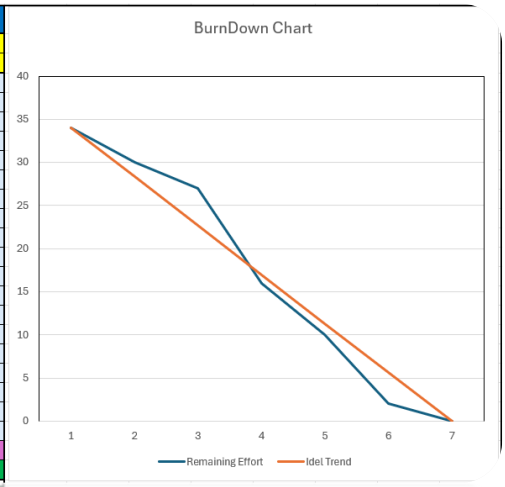
AC2: Dato che descrivo le forme nel Model,
Quando ne rappresento i dati,
Allora ogni forma è descritta con strutture come Point[], RectData e ColorData.

AC3: Dato che voglio visualizzare una forma,
Quando viene richiesta la visualizzazione,
Allora il rendering è gestito interamente dalla View.

Priorità: Alta (5)
Story points: 1
Ruolo: Sviluppatore

4.2. Burndown Chart

BurnDown Chart							
	Initial Story Points Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
User Stories ID	14/05/2025	15/05/2025	16/05/2025	17/05/2025	18/05/2025	19/05/2025	20/05/2025
1	1	1					
2	3	1		2			
3	1	1					
4	2		2				
5	2		2				
6	2			2			
7	2	1		1			
8	2			1	1		
9	2		-1			1	2
10	2					1	1
11	2			2			
12	2			1	1		
13	2				2		
14	2					2	
15	2				1	1	
16	2					2	
17	1					1	
18	1			1			
19	1			1	1		
Remaining Effort	34	30	27	16	10	2	0
Idel Trend	34	28,333333	22,6666667	17	11,333333	5,6666667	0



Il burndown chart mostra come il team, inizialmente, abbia incontrato un rallentamento dovuto all'approccio per la prima volta a problematiche implementative di natura complessa. In particolare, si sono verificati alcuni ostacoli tecnici che hanno inciso sul progresso:

- Cambio del pattern di rendering, che ha richiesto una ristrutturazione significativa del codice.
- Problemi di salvataggio dei file, dove l'applicativo restituiva un messaggio positivo di avvenuto salvataggio, ma in realtà scriveva soltanto 4 byte, corrompendo il file. Il problema è stato individuato e risolto correggendo un errore di sintassi: da "`Serializable{`" a "`Serializable {`".
- Figure invisibili dopo il caricamento del file, dovute al fatto che il main manteneva un riferimento alla vecchia Scene. Questo è stato corretto rimuovendo il riferimento obsoleto e ottenendo dinamicamente la nuova scena tramite il controller.

Nonostante le difficoltà iniziali, il team ha mostrato un netto miglioramento nel ritmo di sviluppo nella fase centrale/finale dello sprint, riuscendo quindi, oltre che a recuperare quello che era stato il ritardo nella fase iniziale, anche ad anticipare anche i tempi previsti.

Componenti principali del grafico:

Linea guida ideale (rossa): Spiega che rappresenta la linea teorica di completamento costante e lineare del lavoro.

Linea reale (blu): Indica l'andamento reale del team.

4.3. First Sprint - Review

User stories complete alla fine di questo sprint:

US1: Apertura Finestra **(1 SP)**
US2: Aggiunta Forma **(3 SP)**
US3: Selettore strumenti di disegno base (**1 SP)**
US4: Disegnare segmenti di linea **(2 SP)**
US5: Disegnare rettangoli **(2 SP)**
US6: Disegnare ellissi **(2 SP)**
US7: Cambio colore bordo **(2 SP)**
US8: Colore interno figura selezionata **(2 SP)**
US9: Salvataggio disegno **(2 SP)**
US10: Caricamento disegno **(2 SP)**
US11: Eliminazione forma **(2 SP)**
US12: Spostamento forma **(2 SP)**
US13: Ridimensionamento forma **(2 SP)**
US14: Taglia forma **(2 SP)**
US15: Copia forma **(2 SP)**
US16: (35*) Separazione del rendering dalla logica del modello **(2 SP)**
US17: (36*) Sostituzione di Point2D con Point Custom **(1 SP)**
US18: (37*) Rimozione di Color dal modello **(1 SP)**
US19: (38*) Rimozione delle classi JavaFx (Rectangle,Ellipse,etc.) dal Model **(1 SP)**

Eventuali user stories rifiutate dal Product Owner:

Nessuna. Tutte le User Story pianificate e quelle aggiuntive sono state completate.

Velocità del progetto misurata:
Somma degli Story Point delle US completate:

$$1 + 3 + 1 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1 = \boldsymbol{34\ SP}$$

4.4. First Sprint – Retrospective

FASE 2: Second Sprint Delivery

Stima iniziale della velocità del progetto:

Durante la fase di pre-game la stima della velocity del team è risultata essere pari a 36 Story Points, calcolata sulla base delle ore settimanali effettivamente disponibili e dell'effort medio effettivo per ciascuna User Story. Il primo sprint è stato pianificato con un carico di lavoro di 30 SP, così da garantire un margine di sicurezza adeguato per gestire eventuali imprevisti, attività di testing o rifiniture. Tuttavia, nel corso dell'analisi architetturale e delle prime fasi di implementazione, sono emerse alcune criticità strutturali che hanno reso necessario l'inserimento di 1 nuova User Story fondamentale per garantire all'utente una buona esperienza durante l'uso del tool.

Questa integrazione ha portato il totale degli SP dello sprint a 32, riducendo lievemente il margine inizialmente previsto. Di fronte a questa situazione, il team ha deciso comunque di lasciare eventuale effort rimanente per l'implementazione di ulteriori user story/ ottimizzazione e revisione del codice.

5. User stories sprint

USER STORY: 16 – Incolla forma

Story Point: 2

ID TASK: 76 – Gestione clipboard nel modello GeoEngine (*Stima: 20 min*)

Estendere il modello `GeoEngine` per includere una proprietà di clipboard in grado di memorizzare una figura copiata o tagliata. La clipboard deve gestire una figura singola, accessibile tramite metodi `setClipboardShape(Shape s)` e `getClipboardShape()`, dove il getter restituisce una copia tramite `clone()`.

ID TASK: 77 – Implementazione comando PasteCommand (*Stima: 30 min*)

Implementare la classe `PasteCommand` che recupera la figura dalla clipboard, crea una copia indipendente e la riposiziona nel disegno con un offset predefinito. Il comando deve essere eseguibile tramite `execute()` e annullabile tramite `undo()`, mantenendo un riferimento interno alla forma incollata per la rimozione successiva. Il comando va registrato nel `CommandManager`.

ID TASK: 78 – Collegare bottone 'Incolla' al controller (*Stima: 15 min*)

Modificare il controller della toolbar per gestire il click sul pulsante 'Incolla'. Il controller deve istanziare `PasteCommand` e invocarlo tramite il `CommandManager`, verificando che l'operazione venga eseguita correttamente e che il canvas venga aggiornato con la nuova figura incollata.

ID TASK: 79 – Scrivere test per PasteCommand
(*Stima: 15 min*)

Scrivere test automatici per verificare il funzionamento completo della feature di incolla. I test devono validare che la figura venga effettivamente incollata nel disegno con l'offset atteso, che `PasteCommand` si comporti correttamente in esecuzione e annullamento, e che il modello rifletta correttamente lo stato aggiornato.

USER STORY: 17 – Gestione cronologia

Story Point: 5

ID TASK: 80 – Progettare la struttura dati per undo/redo nel CommandManager
(*Stima: 50 min*)

Progettare l'architettura interna del `CommandManager` per supportare due pile (stack) separate: una per i comandi eseguiti e una per i comandi annullati. Definire chiaramente le regole per il comportamento delle pile in base all'uso delle operazioni `execute()`, `undo()` e `redo()`.

ID TASK: 81 – Implementare undo() e redo() nel CommandManager
(*Stima: 60 min*)

Implementare i metodi `undo()` e `redo()` nel `CommandManager`, assicurandosi che ogni operazione modifichi correttamente le pile e richiami i metodi `undo()` o `execute()` dei singoli comandi. Gestire casi limite come stack vuoti e nuove esecuzioni dopo un undo.

ID TASK: 82 – Integrazione con l'interfaccia grafica e aggiornamento dinamico dei bottoni
(*Stima: 45 min*)

Collegare i pulsanti 'Undo' e 'Redo' nel controller dell'interfaccia alla logica del `CommandManager`. Aggiornare dinamicamente lo stato dei bottoni in base alla disponibilità delle operazioni, disattivandoli se gli stack sono vuoti.

ID TASK: 83 – Scrivere test automatici per la gestione della cronologia
(*Stima: 45 min*)

Scrivere test unitari e di integrazione per il `CommandManager` che coprano sequenze di `execute()`, `undo()` e `redo()` con più tipi di comandi.

USER STORY: 18 – Gestione livelli

Story Point: 2

ID TASK: 84 – Modificare il modello per supportare il riordinamento delle forme

(*Stima: 25 min*)

Estendere il modello (`GeoEngine`) per fornire metodi `bringToFront(Shape s)` e `sendToBack(Shape s)`. Tali metodi devono spostare la figura selezionata rispettivamente alla fine o all'inizio della lista, così da influenzarne il rendering (che rispetta l'ordine della lista).

ID TASK: 85 – Implementare i comandi BringToFrontCommand e SendToBackCommand

(*Stima: 25 min*)

Creare due classi comando (`BringToFrontCommand` e `SendToBackCommand`) che implementano `Command` e utilizzano i metodi del modello per spostare la figura selezionata in avanti o indietro. Implementare `undo()` per ciascuno dei comandi, memorizzando la posizione originale della figura prima dello spostamento.

ID TASK: 86 – Collegare i comandi al controller e alla UI

(*Stima: 15 min*)

Collegare i pulsanti o le voci di menu 'Porta avanti' e 'Porta indietro' al controller. All'attivazione, il controller deve creare e invocare il comando appropriato tramite il `CommandManager`, aggiornando il canvas per riflettere il nuovo ordine visivo.

ID TASK: 87 – Scrivere test per il riordinamento delle forme nel disegno

(*Stima: 15 min*)

Scrivere test automatici che verificano l'effettivo cambio di posizione nella lista delle forme nel modello dopo l'esecuzione dei comandi. I test devono anche controllare che l'ordine venga ripristinato correttamente tramite `undo()`.

USER STORY: 19 – Cambio livello zoom

Story Point: 3

ID TASK: 88 – Progettare e implementare la gestione dei livelli di zoom

(*Stima: 35 min*)

Implementare un sistema di gestione dello zoom nella parte di visualizzazione attraverso il trackpad. Creare una variabile di stato nel controller o nella vista per mantenere il livello corrente e calcolare il fattore di scala da applicare al rendering.

ID TASK: 89 – Applicare la trasformazione di zoom al canvas
(*Stima: 30 min*)

Integrare il fattore di zoom nella logica di disegno delle forme nel canvas. Utilizzare trasformazioni grafiche JavaFX (`GraphicsContext.scale()`) per ridimensionare la visualizzazione senza alterare i dati del modello. Verificare che lo zoom sia centrato e coerente.

ID TASK: 90 – Collegare i pulsanti + e – al cambio zoom
(*Stima: 25 min*)

Aggiungere i pulsanti + e – nell'interfaccia utente e collegarli al controller. Ogni pressione deve aggiornare il livello di zoom, limitandolo ai valori predefiniti. Aggiornare il canvas ogni volta che il livello cambia.

ID TASK: 91 – Scrivere test e verificare il comportamento grafico dello zoom
(*Stima: 30 min*)

Scrivere test automatici per verificare che il livello di zoom venga applicato correttamente. Eseguire test manuali per controllare che la resa visiva sia corretta e che il canvas risponda bene ai cambi di scala nei vari livelli previsti.

USER STORY: 20 – Scorrimento disegno

Story Point: 1

ID TASK: 92 – Integrare ScrollPane per abilitare lo scorrimento del canvas
(*Stima: 25 min*)

Modificare l'interfaccia grafica inserendo il canvas all'interno di un componente `ScrollPane` di JavaFX. Configurare il contenitore per attivare le barre di scorrimento quando necessario.

ID TASK: 93 – Scrivere test per verifica del comportamento dello scroll
(*Stima: 15 min*)

Scrivere test automatici con TestFX per verificare la presenza e il comportamento delle barre di scorrimento. Simulare l'interazione utente con scroll verticale e orizzontale e verificare che il canvas risponda correttamente aggiornando la vista visualizzata.

USER STORY: 21 – Attivazione griglia

Story Point: 2

ID TASK: 94 – Progettare la visualizzazione della griglia sul canvas
(*Stima: 20 min*)

Definire la logica per disegnare una griglia regolare sopra il canvas, con linee orizzontali e verticali a intervalli costanti. Il disegno della griglia deve avvenire solo quando la modalità è attiva, senza interferire con le figure del disegno.

ID TASK: 95 – Implementare toggle griglia nel controller e nella vista (*Stima: 20 min*)

Aggiungere un pulsante per attivare/disattivare la griglia. Il controller dovrà gestire lo stato della griglia (booleano) e aggiornare il canvas ogni volta che lo stato cambia. Il valore va mantenuto coerente anche dopo ridisegni o aggiornamenti.

ID TASK: 96 – Integrare la griglia nel metodo di rendering del canvas (*Stima: 20 min*)

Modificare il metodo di rendering del canvas per includere, prima del disegno delle forme, la visualizzazione delle linee guida se la modalità griglia è attiva. Gestire correttamente lo zoom e lo scroll per mantenere coerenza visiva.

ID TASK: 97 – Scrivere test automatici e verifica grafica della griglia (*Stima: 20 min*)

Scrivere test automatici e visivi per verificare che l'attivazione e la disattivazione della griglia producano il comportamento atteso. Usare TestFX per simulare il click sul toggle e verificare la presenza o assenza delle linee guida nel canvas.

USER STORY: 22 – Scelta dimensione griglia

Story Point: 3

ID TASK: 98 – Aggiungere parametro per la spaziatura della griglia nel controller (*Stima: 25 min*)

Definire una variabile di stato nel controller per la dimensione attuale della griglia. Aggiungere una `TextField` nell'interfaccia per l'inserimento del parametro. Assicurarsi che questa variabile venga letta dal metodo di rendering quando la griglia è attiva.

ID TASK: 99 – Aggiornare il disegno della griglia con la spaziatura selezionata (*Stima: 30 min*)

Modificare il metodo di disegno della griglia nel canvas affinché utilizzi la nuova spaziatura come passo per le linee. Verificare che il rendering si adatti dinamicamente alla spaziatura, anche in presenza di zoom e scroll.

ID TASK: 100 – Collegare input dimensione griglia all'interfaccia utente (*Stima: 35 min*)

Aggiungere un controllo nella UI per selezionare la dimensione della griglia. Collegare l'evento al controller per aggiornare la variabile di spaziatura e ridisegnare immediatamente il canvas con la nuova configurazione.

ID TASK: 101 – Scrivere test automatici e verifica visiva del cambio dimensione
(*Stima: 30 min*)

Scrivere test automatici (TestFX) per verificare che il cambiamento della dimensione della griglia abbia effetto sul canvas. Includere test con diversi valori di spaziatura e con griglia attiva/disattiva.

USER STORY: 23 – Disegno poligono arbitrario

Story Point: 5

ID TASK: 102 – Gestire lo stato interattivo del disegno di poligoni
(*Stima: 45 min*)

Creare una modalità interattiva per il disegno di poligoni arbitrari. Quando l'utente seleziona la modalità "Poligoni arbitrari", il controller deve iniziare a raccogliere i punti cliccati.

ID TASK: 103 – Visualizzare il contorno del poligono durante il disegno
(*Stima: 40 min*)

Durante la fase di raccolta dei punti, aggiornare dinamicamente il canvas per mostrare una polilinea che collega i punti già cliccati, seguendo il mouse. Il contorno deve essere visibile ma non ancora parte del modello fino alla chiusura del poligono.

ID TASK: 104 – Rilevare la chiusura del poligono e creare la figura
(*Stima: 40 min*)

Chiudere automaticamente il contorno quando l'utente clicca abbastanza vicino al punto iniziale. A quel punto, creare una nuova istanza di `PolygonShape`, composta dalla lista di punti. Inserire la figura nel `GeoEngine` tramite un `DrawPolygonCommand`.

ID TASK: 105 – Integrare la modalità poligono nella toolbar e nel controller
(*Stima: 35 min*)

Aggiungere un pulsante nella toolbar per attivare la modalità poligono. Gestire l'attivazione nel controller principale, assicurandosi che la modalità sia esclusiva e disattivabile, e che si possa annullare l'azione cambiando modalità.

ID TASK: 106 – Scrivere test per il disegno di poligoni arbitrari
(*Stima: 40 min*)

Scrivere test automatici per validare il comportamento della modalità poligono: raccolta dei punti, chiusura, creazione della figura e inserimento nel modello.

USER STORY: 24 – Inserimento testo

Story Point: 2

ID TASK: 107 – Definire la classe TextShape per rappresentare un'etichetta (*Stima: 20 min*)

Creare la classe `TextShape` che estende, con attributi per posizione (x, y), contenuto testuale e stile base. Implementare il metodo `draw(GraphicsContext gc)` per disegnare il testo sul canvas con JavaFX.

ID TASK: 108 – Gestire l'interazione utente per il posizionamento e l'inserimento del testo (*Stima: 20 min*)

Aggiungere la modalità "inserimento testo" nel controller: al click sul canvas, mostrare un campo testuale temporaneo per l'inserimento. Una volta confermato, creare un `TextShape` con il contenuto e la posizione.

ID TASK: 109 – Creare il comando DrawTextCommand per integrare la forma nel modello (*Stima: 20 min*)

Implementare `DrawTextCommand` che incapsula la creazione e l'inserimento di un `TextShape` nel modello. Integrare il comando nel `CommandManager` per supportare undo/redo anche per i testi.

ID TASK: 110 – Scrivere test per l'inserimento e la visualizzazione del testo (*Stima: 20 min*)

Scrivere test automatici per verificare la creazione corretta della forma `TextShape`, il comportamento del comando e il posizionamento nel disegno. Testare anche casi limite come testo vuoto, clic multiplo o caratteri speciali.

USER STORY: 25 – Dimensione testo

Story Point: 2

ID TASK: 111 – Aggiungere l'attributo fontSize a TextShape (*Stima: 20 min*)

Estendere la classe `TextShape` con un attributo fontSize. Aggiornare il metodo `draw(GraphicsContext gc)` per impostare dinamicamente la dimensione del font tramite `gc.setFont(new Font(fontSize))`. Impostare un valore di default iniziale (es. 12 pt).

ID TASK: 112 – Aggiungere selettore dimensione font nella GUI (*Stima: 20 min*)

Aggiungere un controllo (menu a tendina) nella toolbar per la scelta della dimensione del testo. Collegare l'interazione al controller in modo che aggiorni il valore selezionato e lo applichi alla figura selezionata.

ID TASK: 113 – Implementare comando ChangeFontSizeCommand (*Stima: 20 min*)

Creare il comando `ChangeFontSizeCommand` che permette di modificare la dimensione del font di una `TextShape`.

ID TASK: 114 – Scrivere test automatici per la modifica del font
(*Stima: 20 min*)

Scrivere test unitari per verificare che la modifica del `fontSize` venga applicata correttamente e che `ChangeFontSizeCommand` funzioni come previsto. Verificare che il testo venga ridisegnato con la nuova dimensione.

USER STORY: 26 – Rotazione forma

Story Point: 3

ID TASK: 115 – Aggiungere angolo di rotazione alle forme e supporto nel disegno
(*Stima: 30 min*)

Estendere la classe `Shape` con una proprietà `rotationAngle`. Modificare il metodo `draw(GraphicsContext gc)` per applicare una trasformazione grafica (`gc.rotate()`) intorno al centro della figura prima del disegno.

ID TASK: 116 – Aggiungere input per angolo di rotazione nella GUI e collegarlo al controller
(*Stima: 30 min*)

Aggiungere una `TextField` nell'interfaccia per l'inserimento dell'angolo di rotazione in gradi. Collegare l'input al controller per applicare la trasformazione alla figura selezionata in tempo reale o al cambio di focus.

ID TASK: 117 – Implementare comando RotateShapeCommand
(*Stima: 30 min*)

Creare il comando `RotateShapeCommand`, che aggiorna l'angolo di rotazione della figura selezionata.

ID TASK: 118 – Scrivere test automatici per la rotazione
(*Stima: 30 min*)

Scrivere test automatici per verificare che le figure ruotino correttamente attorno al proprio centro, che `RotateShapeCommand` funzioni correttamente e che il disegno rifletterà visivamente l'angolo di rotazione applicato.

USER STORY: 27 – (39*) Spostamento figura seguendo il cursore

Story Point: 3

ID TASK: 119 – Gestire la selezione della figura al click sul canvas
(*Stima: 20 min*)

Modificare il controller per rilevare il click su una figura presente nel disegno. Identificare la figura selezionata e impostarla come attiva, pronta per essere trascinata.

ID TASK: 120 – Implementare il trascinamento della figura con il cursore (*Stima: 25 min*)

Gestire gli eventi `onMouseDragged` e `onMouseReleased` nel canvas. Durante il trascinamento, aggiornare dinamicamente la posizione della figura selezionata seguendo il movimento del cursore in tempo reale.

ID TASK: 121 – Integrare lo spostamento nel sistema di comandi (MoveShapeCommand) (*Stima: 20 min*)

Creare il comando `MoveShapeCommand` per salvare lo spostamento eseguito, con coordinate iniziali e finali.

ID TASK: 122 – Scrivere test per spostamento interattivo delle figure (*Stima: 15 min*)

Scrivere test automatici per verificare che: la figura venga selezionata correttamente, segua il cursore durante il drag.

5.1. Sviluppo del secondo sprint

Ad ogni membro sono stati assegnati dei task

	Ferrara Francesco	Luongo Leonardo	Monda Francesco	Parente Alessia
Tasks assegnati e completati	TASK 78 TASK 82 TASK 86 TASK 90 TASK 98 TASK 100 TASK 102 TASK 105 TASK 108 TASK 112 TASK 116 TASK 119 TASK 120	TASK 79 TASK 83 TASK 87 TASK 89 TASK 91 TASK 92 TASK 93 TASK 94 TASK 96 TASK 97 TASK 99 TASK 101 TASK 103 TASK 106 TASK 110 TASK 114 TASK 118 TASK 122	TASK 77 TASK 81 TASK 85 TASK 95 TASK 109 TASK 113 TASK 117 TASK 121	TASK 76 TASK 80 TASK 84 TASK 88 TASK 104 TASK 107 TASK 111 TASK 115
Tasks assegnati e non completati				
Tasks ancora non assegnati				

5.1.1. Sprint release

Durante il secondo sprint, il team ha introdotto importanti funzionalità di modifica, riutilizzo e inserimento avanzato di elementi nel disegno. Queste novità hanno esteso le possibilità offerte all'utente, rendendo l'applicazione più flessibile e interattiva. Le funzionalità sono state sviluppate mantenendo piena aderenza all'architettura MVC e integrando nuove componenti nel sistema esistente senza introdurre accoppiamenti indesiderati.

Funzionalità rilasciate:

Le funzionalità rilasciate in questa Sprint sono le seguenti:

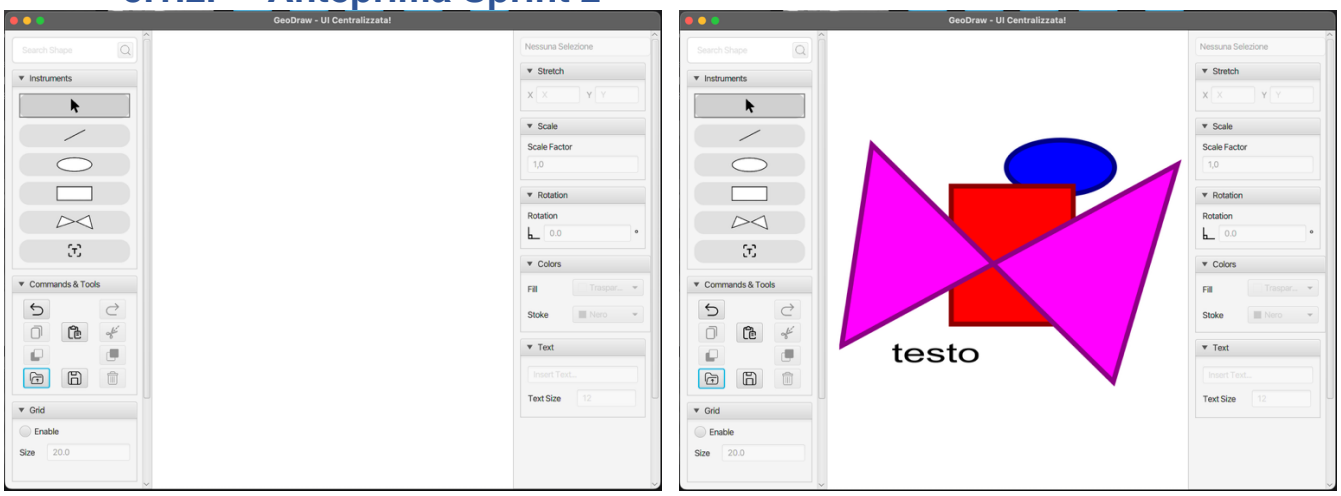
- **Incolla forma**
- **Gestione cronologia**
- **Gestione livelli**
- **Spostamento figura seguendo il cursore**
- **Cambio livello zoom**
- **Scorrimento disegno**
- **Attivazione griglia**
- **Scelta dimensione griglia**
- **Disegno di poligoni arbitrari**

- **Inserimento testo**
- **Dimensione testo**
- **Rotazione forma**

Design Pattern applicati:

- **MVC** per la separazione tra logica, interfaccia e gestione eventi.
- **Command** per incapsulare operazioni modificabili e abilitarne l'annullamento.
- **State** per gestire gli strumenti attivi di disegno in modo modulare.
- **Prototype** per clonare forme durante le operazioni di copia/incolla.
- **Singleton** per la gestione centralizzata della clipboard.
- **Observer** per aggiornare automaticamente la vista a ogni modifica del modello.
- **Visitor** per delegare alla View il rendering delle nuove forme, mantenendo il Model indipendente da JavaFX.

5.1.2. Anteprima Sprint 2



Le due schermate rappresentano l'evoluzione dell'interfaccia utente durante la seconda sprint del progetto. L'immagine a sinistra mostra lo stato iniziale dell'applicazione, con la UI centralizzata pronta all'uso ma ancora priva di contenuti sul foglio di disegno. Tutti gli strumenti sono visibili e ordinati nella barra laterale, compresi quelli per il disegno delle forme, la gestione dei comandi, i colori, la scala, la rotazione e la griglia. Nella seconda schermata, a destra, vediamo invece un esempio pratico di utilizzo: sono state disegnate varie forme geometriche sovrapposte, è stato aggiunto un testo, e sono stati applicati colori differenti.

5.1.3. Update product backlog

Durante lo sviluppo del progetto, è emersa una criticità legata all'interazione grafica nella fase di spostamento delle forme. In particolare, si è osservato che, al momento del trascinamento, la figura selezionata non seguiva visivamente il cursore del mouse durante il movimento: veniva invece visualizzata solo nella posizione iniziale e in quella finale, rendendo l'interazione poco reattiva.
Per risolvere questa limitazione e migliorare l'esperienza utente, è stata aggiunta una nuova User Story al Product Backlog. L'obiettivo è consentire all'utente di trascinare attivamente una forma selezionata sul canvas, visualizzandone lo spostamento in tempo reale mentre segue il cursore del mouse.

39. Spostamento figura seguendo il cursore

Come utente, voglio poter trascinare una forma selezionata sul canvas spostandola con il cursore del mouse, per controllarne direttamente la posizione

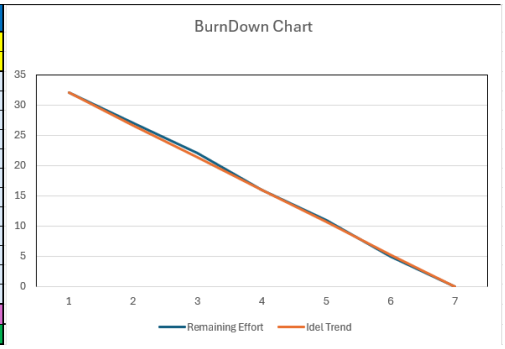
Criteri di accettazione
AC1:
Dato che una forma è presente sul canvas,
Quando clicco su di essa,
Allora la forma deve risultare selezionata e pronta per essere spostata.

AC2:
Dato che ho selezionato una forma sul canvas,
Quando trascino il cursore del mouse,
Allora la forma deve seguire in tempo reale il movimento del cursore fino al rilascio del tasto.

Priorità: Medio alta (4)
Story points: 2
Ruolo: Utente

5.2. Burndown Chart

BurnDown Chart							
	Initial Story Points Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
User Stories ID	21/05/2025	22/05/2025	23/05/2025	24/05/2025	25/05/2025	26/05/2025	27/05/2025
16	2	1	1				
17	5	3	1	1			
18	2		1	1			
19	3		1	-1	2		
20	1				-1	1	
21	2			2			
22	3			1	2		
23	5	1	1			2	1
24	2					1	1
25	2				1	1	
26	3			2		1	
27	2				1		1
Remaining Effort	32	27	22	16	11	5	0
Idel Trend	32	26,666667	21,3333333	16	10,666667	5,3333333	0



All'inizio dello sprint, il carico di lavoro stimato era di 32 story points, distribuiti sulle funzionalità pianificate e le relative attività di sviluppo.

Durante l'implementazione sono emersi tre problemi tecnici imprevisti che hanno richiesto interventi correttivi:

1. Click su forme ruotate: le forme risultavano ruotate solo a livello grafico, ma non logicamente, impedendo il rilevamento corretto dei click. È stato necessario riscrivere la logica del metodo contains, con un impatto non trascurabile sul tempo stimato.
2. Aliasing durante zoom out: un forte aliasing si presentava con griglie molto dense. Per gestirlo è stato imposto un limite allo zoom e alla densità, bilanciando qualità visiva e prestazioni.

3. Conflitto tra scrollbar e movimento delle forme: il sistema di scroll del pannello interferiva con il movimento delle figure selezionate. Si è implementata una gestione più intelligente dell'input, distinguendo tra scroll e move.

Per gestire questi problemi, sono stati aggiunti 2 story points alla stima iniziale, portando il totale effettivo dello sprint a 34 story points.

Questo si riflette nel burndown chart, che avrebbe presentato un'andamento leggermente in anticipo nella prima parte della sprint, che, a causa dei problemi riscontrati, ha comunque portato ad un andamento costante e puntuale durante l'intero sprint.

Componenti principali del grafico:

Linea guida ideale (rossa): Spiega che rappresenta la linea teorica di completamento costante e lineare del lavoro.

Linea reale (blu): Indica l'andamento reale del team.

5.3. Second Sprint - Review

User stories complete alla fine di questo sprint:

US16: Incolla forma **(2 SP)**
US17: Gestione cronologia **(5 SP)**
US18: Gestione livelli (**2 SP)**
US19: Cambio livello zoom **(3 SP)**
US20: Scorrimento disegno **(1 SP)**
US21: Attivazione griglia **(2 SP)**
US22: Scelta dimensione griglia **(3 SP)**
US23: Disegno poligono arbitrario **(5 SP)**
US24: Inserimento testo **(2 SP)**
US25: Dimensione testo **(2 SP)**
US26: Rotazione forma **(3 SP)**
US27(39*): Spostamento figura seguendo il cursore **(2 SP)**

Eventuali user stories rifiutate dal Product Owner:

Nessuna. Tutte le User Story pianificate e quelle aggiuntive sono state completate.

Velocità del progetto misurata:
Somma degli Story Point delle US completate:

$$2 + 5 + 2 + 3 + 1 + 2 + 3 + 5 + 2 + 2 + 3 + 2 = \mathbf{32\ SP}$$

5.4. Second Sprint – Retrospective

FASE 3: Third Sprint Delivery

Stima iniziale della velocità del progetto:

La velocity stimata per questa sprint è stata di 30 Story Points. Durante la fase iniziale di analisi del lavoro, non è emersa la necessità di aggiungere nuove user story.

Tuttavia, nel corso dello sviluppo, sono emerse alcune criticità tecniche impreviste, che hanno richiesto piccoli interventi correttivi. Nonostante ciò, il team è riuscito a mantenere un buon ritmo, lavorando in modo coerente con la stima iniziale e completando tutte le attività pianificate nei tempi previsti.

6. User stories sprint

USER STORY: 27 – Riflessione forma

Story Point: 3

ID TASK: 123 – Aggiungere comandi per riflessione orizzontale e verticale (*Stima: 35 min*)

Creare due comandi distinti `ReflectHorizontalCommand` e `ReflectVerticalCommand` che eseguano la trasformazione della forma selezionata invertendo le coordinate rispetto all'asse centrale (X o Y).

ID TASK: 124 – Aggiungere metodi di riflessione nelle shape (*Stima: 30 min*)

Estendere le classi delle forme con metodi `reflectHorizontally()` e `reflectVertically()` che modificano correttamente le coordinate.

ID TASK: 125 – Integrare comandi di riflessione nella GUI (*Stima: 30 min*)

Aggiungere due pulsanti nell'interfaccia per attivare i comandi di riflessione, abilitati solo se è selezionata una forma. Collegare l'azione agli appositi comandi.

ID TASK: 126 – Test per riflessione (*Stima: 25 min*)

Scrivere test automatici per verificare la corretta applicazione della riflessione e la modifica delle coordinate.

USER STORY: 28 – Stretching forma

Story Point: 3

ID TASK: 127 – Mappare i campi di input per stretching X e Y
(*Stima: 40 min*)

Identificare e collegare i TextField `shapeStretchXField` e `shapeStretchYField` della GUI al controller. I campi devono consentire all'utente di inserire manualmente nuove dimensioni per la forma selezionata.

ID TASK: 128 – Implementare la logica di stretching nella forma
(*Stima: 40 min*)

Aggiungere metodi nelle classi delle shape per aggiornare larghezza o altezza in base ai valori forniti.

ID TASK: 129 – Aggiornare il modello e ridisegnare la vista
(*Stima: 20 min*)

Dopo l'inserimento dei valori, aggiornare il modello e notificare la View per ridisegnare correttamente la forma.

ID TASK: 130 – Scrivere test per stretching con input numerici
(*Stima: 20 min*)

Creare test per validare l'applicazione di input X e Y. I test devono coprire sia input corretti che casi errati e simulare l'interazione utente.

USER STORY: 29 – Raggruppamento forme

Story Point: 5

ID TASK: 131 – Definire la classe ShapeGroup per contenere forme multiple
(*Stima: 60 min*)

Creare la classe ShapeGroup che implementa l'interfaccia Shape e contiene una lista ordinata di forme. Deve propagare le trasformazioni (traslazione, selezione, rendering) a tutte le forme contenute.

ID TASK: 132 – Implementare il comando GroupShapesCommand
(*Stima: 60 min*)

Creare il comando che prende le forme selezionate, le rimuove dal modello e le sostituisce con un nuovo oggetto ShapeGroup.

ID TASK: 133 – Aggiornare selezione, spostamento e delete per i gruppi
(*Stima: 45 min*)

Aggiornare il controller per selezionare il gruppo. Esso deve rispondere a drag, delete e altri comandi come una singola entità.

ID TASK: 134 – Testare creazione, selezione e comportamento del gruppo
(*Stima: 35 min*)

Scrivere test automatici per verificare che la creazione del gruppo avvenga correttamente, che le forme al suo interno restino coerenti.

USER STORY: 30 – Separa il raggruppamento forme

Story Point: 3

ID TASK: 135 – Definire il comando UngroupCommand
(*Stima: 40 min*)

Creare il comando `UngroupCommand` che prende un `ShapeGroup`, ne estrae le forme interne e le reinserisce nel modello come entità separate.

ID TASK: 136 – Implementare logica di separazione nel controller
(*Stima: 35 min*)

Aggiornare il controller per riconoscere un gruppo selezionato e sostituirlo con le singole forme. La selezione viene svuotata dopo la separazione.

ID TASK: 137 – Aggiornare GUI e gestione selezione post-separazione
(*Stima: 25 min*)

Dopo la separazione, aggiornare la GUI e la View per mostrare le forme slegate. Nessuna forma deve risultare selezionata automaticamente.

ID TASK: 138 – Testare comportamento del comando di separazione
(*Stima: 20 min*)

Verificare che il gruppo venga rimosso correttamente e che tutte le forme interne siano ripristinate.

USER STORY: 31 – Creare una Forma Riutilizzabile

Story Point: 5

ID TASK: 139 – Definire la struttura ReusableShapeDefinition
(*Stima: 25 min*)

Creare una classe che rappresenti una forma salvata, contenente nome, tipo, dimensioni, colori, posizione e struttura interna (anche gruppi). Deve essere serializzabile per l'uso in libreria.

ID TASK: 140 – Implementare comando SaveReusableShapeCommand
(*Stima: 15 min*)

Realizzare un comando che, dato un gruppo selezionato, crea un oggetto `ReusableShapeDefinition` e lo salva in una collezione gestita dal `GeoEngine`.

ID TASK: 141 – Integrare salvataggio nella GUI con nome personalizzato
(*Stima: 25 min*)

Aggiungere un'interfaccia per permettere all'utente di assegnare un nome alla forma da salvare. Impedire il salvataggio se il campo è vuoto.

ID TASK: 142 – Scrivere test su creazione e registrazione della forma
(*Stima: 15 min*)

Testare che la forma venga salvata correttamente, sia visibile nella libreria e mantenga dimensioni, colori e struttura anche se la forma originale viene modificata o rimossa.

USER STORY: 32 – Usare una Forma Riutilizzabile

Story Point: 3

ID TASK: 143 – Creare comando InsertReusableShapeCommand (*Stima: 40 min*)

Implementare un comando che genera una copia indipendente di una `ReusableShapeDefinition` selezionata e la inserisce nel modello. La copia deve mantenere geometria e stile.

ID TASK: 144 – Attivare lo strumento di inserimento nella GUI (*Stima: 30 min*)

Al clic su una forma nella libreria laterale, il sistema entra in modalità di inserimento. Il cursore cambia stato e si attende un click sul canvas per posizionarla.

ID TASK: 145 – Posizionare la forma copiata sul canvas (*Stima: 30 min*)

Al click sul canvas, la forma viene inserita seguendo il cursore, come per gli altri strumenti di disegno.

ID TASK: 146 – Testare inserimento, indipendenza e visibilità (*Stima: 20 min*)

Verificare che la copia sia indipendente dall'originale, che possa essere selezionata, modificata e cancellata come una normale forma, e che sia visibile nel canvas.

USER STORY: 33 – Esportare Forme Riutilizzabili in una Libreria

Story Point: 3

ID TASK: 147 – Definire formato .geolib per salvataggio (*Stima: 40 min*)

Progettare un formato serializzabile per rappresentare `ReusableShapeDefinition`, includendo nome, tipo, dimensioni e colori. Il formato deve essere compatibile con l'import.

ID TASK: 148 – Implementare comando ExportReusableLibraryCommand (*Stima: 30 min*)

Creare il comando che consente all'utente di scegliere quali forme esportare dalla libreria interna e di salvarle in un file .geolib.

ID TASK: 149 – Creare dialogo per selezione forme e nome file (*Stima: 30 min*)

Integrare nella GUI una finestra che permetta all'utente di selezionare più forme dalla libreria e di specificare il percorso e nome del file da generare.

ID TASK: 150 – Testare generazione e contenuto del file (*Stima: 20 min*)

Scrivere test per assicurarsi che il file esportato contenga correttamente le definizioni scelte e che possa essere riaperto senza perdita di dati.

USER STORY: 34 – Importare Forme Riutilizzabili da una Libreria

Story Point: 5

ID TASK: 151 – Implementare parser per file .geolib
(*Stima: 60 min*)

Scrivere una classe che legga file .geolib, validi il formato e ricostruisca oggetti `ReusableShapeDefinition` pronti per l'uso nel progetto corrente.

ID TASK: 152 – Creare comando ImportReusableLibraryCommand
(*Stima: 60 min*)

Realizzare un comando per selezionare un file .geolib, lo carica e aggiunge le forme riutilizzabili nella libreria interna del sistema.

ID TASK: 153 – Gestire conflitti di nome e aggiornare la GUI
(*Stima: 60 min*)

In caso di nomi duplicati, mostrare un messaggio di errore.

ID TASK: 154 – Scrivere test per importazione, errori e conflitti
(*Stima: 20 min*)

Testare casi validi, file corrotti e nomi duplicati. Verificare che l'import non modifichi la libreria esistente in caso di errore e che le forme importate siano utilizzabili.

6.1. Sviluppo del terzo sprint

Ad ogni membro sono stati assegnati dei task

	Ferrara Francesco	Luongo Leonardo	Monda Francesco	Parente Alessia
Tasks assegnati e completati	TASK 125 TASK 127 TASK 137 TASK 141 TASK 144 TASK 149 TASK 150 TASK 153	TASK 126 TASK 129 TASK 130 TASK 133 TASK 134 TASK 136 TASK 138 TASK 142 TASK 145 TASK 146 TASK 154	TASK 123 TASK 132 TASK 135 TASK 140 TASK 143 TASK 148 TASK 152	TASK 124 TASK 128 TASK 131 TASK 139 TASK 147 TASK 151
Tasks assegnati e non completati				
Tasks ancora non assegnati				

6.1.1. Sprint release

Durante il terzo sprint, il team ha completato e rilasciato un insieme significativo di funzionalità legate alla gestione avanzata delle forme e alla riusabilità degli elementi grafici. Queste funzionalità migliorano notevolmente l'esperienza utente, offrendo strumenti di manipolazione.

Funzionalità rilasciate:

Le funzionalità rilasciate in questa Sprint sono le seguenti:

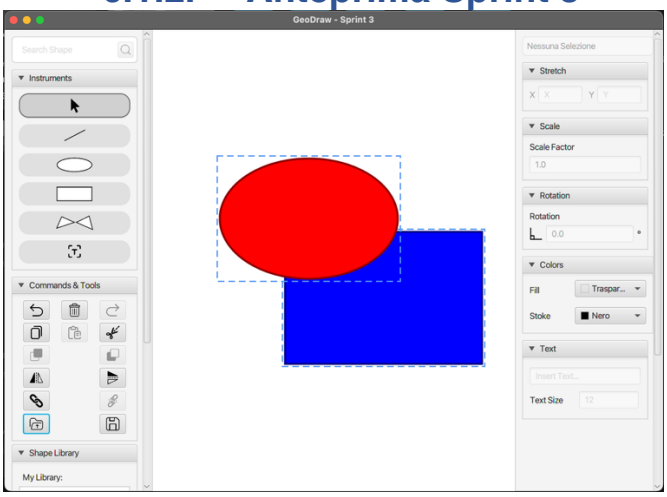
- **Riflessione forma**
- **Stretching forma**
- **Raggruppamento forme**
- **Separa il raggruppamento forme**
- **Creare una Forma Riutilizzabile**
- **Usare una Forma Riutilizzabile**
- **Esportare Forme Riutilizzabili in una Libreria**
- **Importare Forme Riutilizzabili da una Libreria**

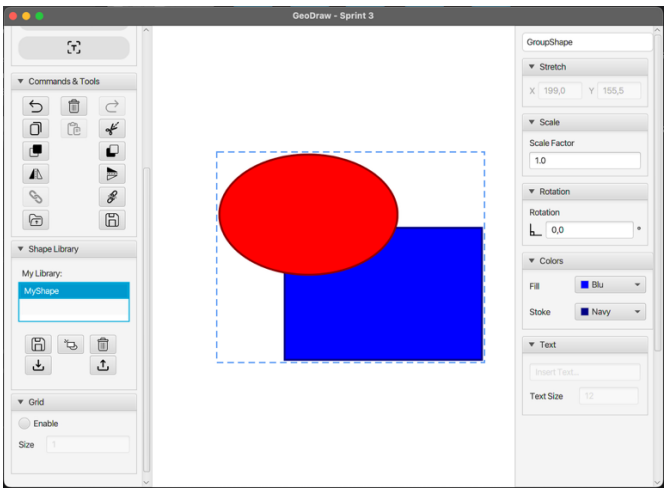
Design Pattern applicati:

- **MVC** per la separazione tra logica, interfaccia e gestione eventi.
- **Command** per incapsulare operazioni modificabili e abilitarne l'annullamento.
- **State** per gestire gli strumenti attivi di disegno in modo modulare.
- **Prototype** per clonare forme durante le operazioni di copia/incolla.
- **Singleton** per la gestione centralizzata della clipboard.
- **Observer** per aggiornare automaticamente la vista a ogni modifica del modello.
- **Visitor** per delegare alla View il rendering delle nuove forme, mantenendo il Model indipendente da JavaFX.

- **Facade** per fornire un'interfaccia semplificata alla gestione delle forme riutilizzabili, nascondendo la complessità sottostante.

6.1.2. Anteprima Sprint 3





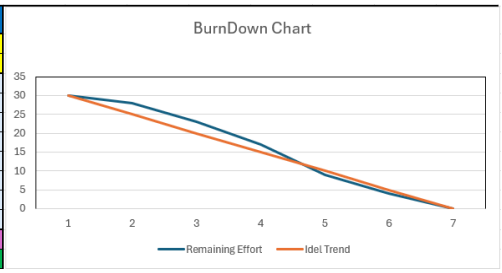
La terza e ultima sprint segna il completamento del progetto GeoDraw, introducendo funzionalità avanzate. Le schermate evidenziano il pieno supporto al raggruppamento di forme, alla loro manipolazione tramite stretching e riflessione, e alla possibilità di salvare e riutilizzare elementi personalizzati all'interno di una libreria dedicata. L'impiego di molteplici design pattern (MVC, Command, State, Prototype, Singleton, Observer, Visitor, Facade) ha permesso di mantenere il codice modulare, estendibile e indipendente dalla View.

6.1.3. Update product backlog

Nel corso di questa sprint non sono emerse nuove User Story da parte del Product Owner. Il Product Backlog rimane invariato rispetto alla versione precedente.

6.2. Burndown Chart

BurnDown Chart							
	Initial Story Points Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
User Stories ID	28/05/2025	29/05/2025	30/05/2025	31/05/2025	01/06/2025	02/05/2025	03/05/2025
27	3	1	2				
28	3	1	1	1			
29	5		1	2	2		
30	3		1	1	1		
31	5			1	1	1	2
32	3			1	1	1	
33	3				1	1	1
34	5				2	2	1
Remaining Effort	30	28	23	17	9	4	0
Idel Trend	30	25	20	15	10	5	0



L'andamento del team, seppur con un leggero ritardo iniziale rispetto alla velocity stimata, ha mostrato una buona capacità di recupero, riuscendo a rientrare nei tempi previsti per il completamento dello sprint.

Durante lo sviluppo sono emerse alcune complessità architetturali che hanno richiesto riflessioni più approfondite:

1. **Disaccoppiamento tra Controller e Model**: si è dovuto ristrutturare il meccanismo di notifica degli observers, spostando la responsabilità dalla logica di controllo alla classe Drawing all'interno del Model, così da garantire una maggiore coerenza e indipendenza tra i componenti.
2. **Implementazione del pattern Facade:** `PersistenceController` potrebbe essere visto come una Facade molto semplice che nasconde i dettagli specifici di `IDrawingSerializer` e `IReusableShapeLibrarySerializer` a `GeoEngine` per le operazioni di salvataggio/caricamento. Questo beneficia `GeoEngine` che non deve preoccuparsi di gestire due serializer diversi direttamente.

6.3. Third Sprint - Review

User stories complete alla fine di questo sprint:

US27: Riflessione forma **(3 SP)**
US28: Stretching forma (**3 SP)**
US29: Raggruppamento forme **(5 SP)**
US30: Separa il raggruppamento forme **(3 SP)**
US31: Creare una Forma Riutilizzabile **(5 SP)**
US32: Usare una Forma Riutilizzabile **(3 SP)**
US33: Esportare Forme Riutilizzabili in una Libreria **(3 SP)**
US34: Importare Forme Riutilizzabili da una Libreria **(5 SP)**

Eventuali user stories rifiutate dal Product Owner:
Nessuna. Tutte le User Story pianificate e quelle aggiuntive sono state completate.

Velocità del progetto misurata:
Somma degli Story Point delle US completate:

$$3 + 3 + 5 + 3 + 5 + 3 + 3 + 5 = \mathbf{30}\ \boldsymbol{SP}$$

6.4. Third Sprint – Retrospective