Second Sprint

Second Sprint - Development

Sprint Backlog (foglio *User Stories 2nd Sprint Planning*): Tabella user stories

Tasks assegnati ad ogni membro del team

Al seguente link, nel foglio *Tasks 2nd Sprint* è possibile osservare i tasks assegnati ad ogni membro del team e il loro stato al termine della sprint (completati, assegnati ma non completati, non assegnati).

Tabella user stories

User Stories aggiunte o modificate dal Product Owner:

- Nessuna User Story da aggiungere per il Product Owner

Second Sprint - Review

User Stories complete alla fine dello Sprint

Sono state completate tutte le User Stories pianificate per questa seconda Sprint; in più, avendo concluso queste in un tempo minore rispetto a quello stimato inizialmente, sono state aggiunte le seguenti User Stories, anch'esse portate a termine:

2.6) Taglia forma

2.9) Profondità regolabile

3.2) Spazio extra

3.3) Griglia

3.4) Dimensione griglia.

È possibile osservare al link di seguito l'elenco aggiornato di tutte le User Stories portate a termine durante la sprint (foglio User Stories 2nd Sprint Planning):

Tabella user stories

Task per ogni User Story aggiunta

User story 2.6 Taglia forma

2.6.1 Aggiornamento componente grafico bottone taglia

Aggiornamento del menu contestuale della forma con un bottone dedicato all'operazione di taglia in ambiente Scene Builder.

2.6.2 Implementazione della logica dell'operazione taglia

Implementazione della logica dell'operazione taglia forma.

2.6.3 Esecuzione test funzionale operazione taglia

Eseguire functional test cases rispetto alla funzionalità di taglia forma implementata.

User story 2.9 Profondità regolabile

2.9.1 Aggiornamento componente grafico bottoni profondità regolabile

Aggiornare il menu contestuale della forma con due bottoni, rispettivamente per portare in primo piano e sullo sfondo la forma selezionata.

2.9.2 Implementazione logica profondità regolabile

Implementazione della logica per la profondità regolabile delle forme, in particolare per l'operazione di *porta in primo piano* e *porta sullo sfondo*.

2.9.3 Esecuzione test funzionale profondità regolabile

Esecuzione functional test cases per le operazioni di *porta in primo piano* e *porta sullo sfondo* implementate.

User story 3.2 Spazio Extra

3.2.1 Creazione dell'area di disegno estesa

Aggiungere un componente grafico che permetta di disegnare oltre i limiti visibili della finestra.

3.2.2 Logica dell'area estesa

Gestire il posizionamento e l'interazione degli oggetti anche fuori dalla vista iniziale.

3.2.3 Scorrimento dell'area di disegno

Abilitare lo scrolling per navigare nello spazio extra nell'area di disegno.

3.2.4 Test funzionale spazio extra

Verificare che area estesa e scrolling funzionino correttamente.

User story 3.3 Griglia

3.3.1 Aggiunta opzione UI per attivare/disattivare la griglia

Modificare l'interfaccia grafica della toolbar per includere un checkbox per mostrare la griglia.

3.3.2 Implementazione della logica per visualizzare e nascondere la griglia

Implementazione della logica necessaria per far visualizzare e nascondere la griglia.

3.3.3 Eseguire test di attivazione/disattivazione

Eseguire i test funzionali per garantire che la griglia sia correttamente visibile/invisibile in base al flag.

User story 3.4 Dimensione Griglia

3.4.1 Aggiunta componente grafico per selezione dimensione riquadro

Integrare uno slider per permettere la selezione della dimensione del riquadro della griglia.

3.4.2 Implementazione della logica per l'aggiornamento della dimensione

Implementare la logica che gestisce l'evento di aggiornamento della dimensione della griglia.

3.4.3 Esecuzione dei test sulla variazione di dimensione

Eseguire i test funzionali per verificare che la griglia sia visualizzata correttamente con ogni dimensione selezionabile.

User Stories rifiutate dal Product Owner: nessuna

Project Velocity misurata: 45 story points.

Per il prossimo Sprint considereremo una stima di 38 story points, considerando una media aritmetica degli story points completati nelle prime due sprint. Infatti, in questa sprint abbiamo completato molti più story points di quelli previsti.

Durante la Sprint abbiamo ritenuto opportuno, dopo una consultazione con il Product Owner:

- gestire un debito tecnico introdotto dalla precedente Sprint e necessario per rivedere la modellazione del diagramma delle classi, con l'obiettivo di spostare la logica di business dal controller al model, in modo da rispettare correttamente il pattern architetturale MVC e per migliorare la gestione del comportamento dinamico dell'applicazione in base allo stato corrente, è stato necessario introdurre il pattern State. Questa scelta progettuale si è resa opportuna per disaccoppiare la logica condizionale legata agli stati dell'applicazione e rendere il codice più estendibile e manutenibile. Il Controller ora delega il comportamento specifico a oggetti di stato concreti, migliorando la coesione e facilitando eventuali evoluzioni future. In relazione alla risoluzione del debito tecnico, sono state delineate le seguenti tasks:

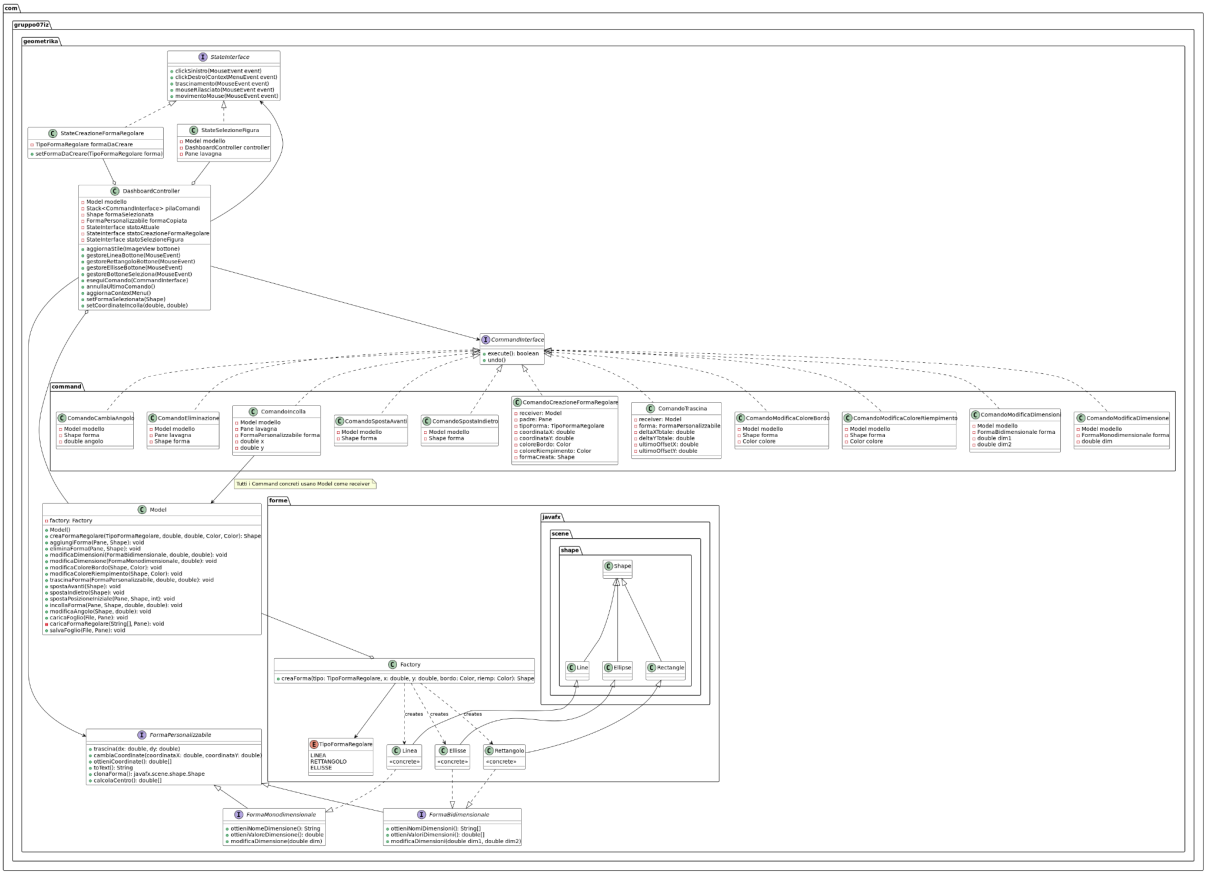
  ## DT 1

  DT.1.1 Aggiornamento design
  Aggiornamento del design per includere le classi necessarie all'implementazione del pattern State relativi alle operazioni di creazione di una forma regolare e della selezione di una forma (per trascinamento e per menu contestuale).
  DT.1.2 Implementazione stato creazione forma regolare
  Implementazione della classe ***StatoCreazioneFormaRegolare*** che va a definire l'operazione delle specifica forma regolare che si vuole creare sulla lavagna, così come il resto delle interazioni del mouse sulla lavagna all'interno di quello stato.
  DT.1.3 Implementazione stato selezione figura
  Implementazione della classe ***StatoSelezioneFigura*** che va a definire le interazioni con il mouse all'interno della lavagna in questo tasto per utilizzare il click sinistro per eseguire il trascinamento della figura e il click destro per definire il menu di interazione.
  DT.1.4 Integrazione dello stato all'interno del Controller
  Aggiunta degli attributi necessari alla gestione degli stati all'interno del controller, così come la loro inizializzazione e la logica di transizione da uno stato all'altro.

- Effettuare un backlog refinement modificando la priorità di alcune User Stories per renderla più conforme con le macro priorità stabilite dal Product Owner all'atto dell'assegnazione del progetto. In particolare:
  - 3.1 Zoom da 13 a 21 value points
  - 3.2 Spazio extra è stata passata da 34 a 21 value points
  - 3.4 Dimensione griglia da 13 a 21 value points
  - 4.1 Inserimento di poligono arbitrari da 34 a 21 value points
  - 4.3 Rotazione di una forma ad angolo arbitrario da 21 a 13 value points
  - 4.5 Allungamento orizzontale o verticale di una forma da 13 a 8 value points

Design aggiornato

Cliccando sull'immagine del Class Diagram aggiornato allegata di seguito è possibile osservare meglio le modifiche apportate sopra citate.



Durante questa sprint sono state apportate due modifiche fondamentali al design dell'applicazione in sviluppo; infatti, sono stati introdotti i pattern State e Command.

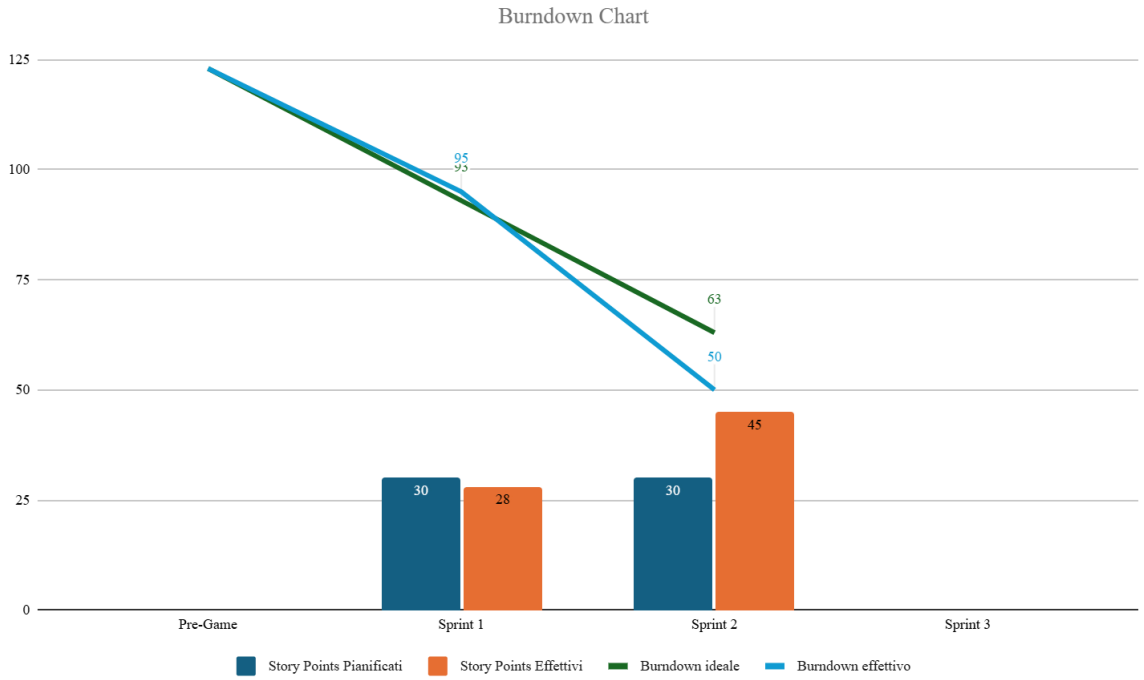
Il pattern State è risultato adatto in quanto nella nostra applicazione sono attualmente possibili due stati, ovvero la creazione di una forma regolare e la selezione della forma (e in futuro si prevede di inserirne altri, quali ad esempio la creazione di un poligono arbitrario e di un testo). Usando il pattern, ogni classe Stato (nel nostro caso ***StateCreazioneFormaRegolare*** e ***StateSelezioneFigura***) gestisce il comportamento specifico di quell'unico stato, riducendo la complessità all'interno della classe principale(***DashboardController***). Nel nostro caso, non esiste una sequenza rigida di transizione tra gli stati: ciascuno può passare liberamente a qualunque altro in base all'interazione dell'utente. Inoltre, è stato scelto come stato iniziale quello della creazione di un'ellisse, ma tale scelta è stata arbitraria, in quanto qualsiasi altro stato iniziale sarebbe stato ugualmente valido, dato che non esiste una dipendenza logica tra gli stati.

Ogni classe Stato effettua l'override dei metodi la cui firma è stata definita nella ***StateInterface*** e che riguardano principalmente il trascinamento della forma e i click effettuati su di essa; infatti, la ***StateInterface*** specifica quali metodi ogni stato deve

implementare, così che il contesto (ovvero l'oggetto che cambia comportamento in base allo stato, nel nostro caso ***DashboardController***) possa interagire con gli stati senza conoscere i dettagli specifici di ciascuno.

Il pattern Command è stato introdotto principalmente per l'implementazione della reversibilità delle operazioni; serve per incapsulare un'operazione come oggetto, separando chi invoca l'operazione da chi la esegue. Nella ***CommandInterface*** è presente la firma dei due metodi di cui ogni ***ConcreteCommand*** deve effettuare l'Override, ovvero ***esegui()*** e ***undo().*** Questi comandi utilizzano il model come receiver: eseguono l'azione direttamente su di esso e sono in grado di annullarla tramite il meccanismo di undo. Il Model e il DashboardController rappresentano rispettivamente il Receiver, ovvero l'oggetto sul quale i comandi agiscono, e l'Invoker, ovvero colui che richiama i comandi. Infine, è importante sottolineare che i comandi eseguiti dall'Invoker sono generati dalle interazioni dell'utente con l'applicazione, ad esempio clic su bottoni o modifiche apportate tramite l'interfaccia grafica.

Burndown Chart



#Sprint	Story Points Pianificati	Story Points Effettivi	Burndown ideale	Burndown effettivo
Pre-Game			123	123
Sprint 1	30	28	93	95
Sprint 2	30	45	63	50
Sprint 3				

Second Sprint - Retrospective

Le decisioni prese durante il retrospective meeting di questa Sprint sono:

Stop (things to stop doing):

- Modifiche grafiche non essenziali che causano sprechi di tempo e risorse.
- Rimandare il refactoring di codice problematico.
- Delegare le task relative ai testing ai membri del team che non si occupano dell'implementazione della funzionalità in esame.

Less of:

- Iniziare nuove task prima di averne concluse altre.
- Daily meeting troppo lunghi.

Keep doing:

- Supporto tra i membri del team.
- Attenzione alle Definition of Done e ai Criteri di Accettazione delineati.

More of (more things to do):

- Scrivere codice pulito sin dall'inizio, riducendo la necessità di refactoring successivo.
- Fare più attenzione nel mantenere la dashboard di Trello aggiornata e consistente(ad esempio archiviando e spostando le schede concluse).

Start (things to start doing):

- Aggiungere i commenti JavaDoc subito dopo aver completamento la logica relativa alla User Story considerata.