Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Filesystem Command-line Simulator

Gruppo-16

Studente/i	Relatore	Correlatore
Nicholas Gherbi Diego Belloni Andrea Perlini		

Corso di laurea	Modulo / Codice Progetto	Anno
Ingegneria Informatica	Ingegneria del software II	2025/2026

Committente	Data
Giancarlo Corti Matteo Besenzoni	17/12/2025

STUDENTSUPSI

Indice

Contesto e motivazione

Questo progetto nasce all'interno del corso di Ingegneria del Software II, con l'obiettivo di applicare i principi di progettazione e sviluppo appresi durante il corso quali:

Sviluppo From Scratch

Realizzazione di un prodotto software completo partendo da zero.

Gestendo l'intero ciclo di vita.

Design Patterns

Utilizzo coerente di strumenti quali i design patterns per garantire modularità e manutenibilità.

Quality Assurance&Testing

Utilizzo di strumenti come **JUnit 5, Mockito, TestFX** e **JaCoCo.**

Per verificare solidità e qualità del codice.

Il Problema: Requisiti e Vincoli

Produrre una stand-alone desktop GUI, che permetta ad un utente di interfacciarsi con un filesystem simulato tramite l'interpretazione di comandi testuali simil UNIX.

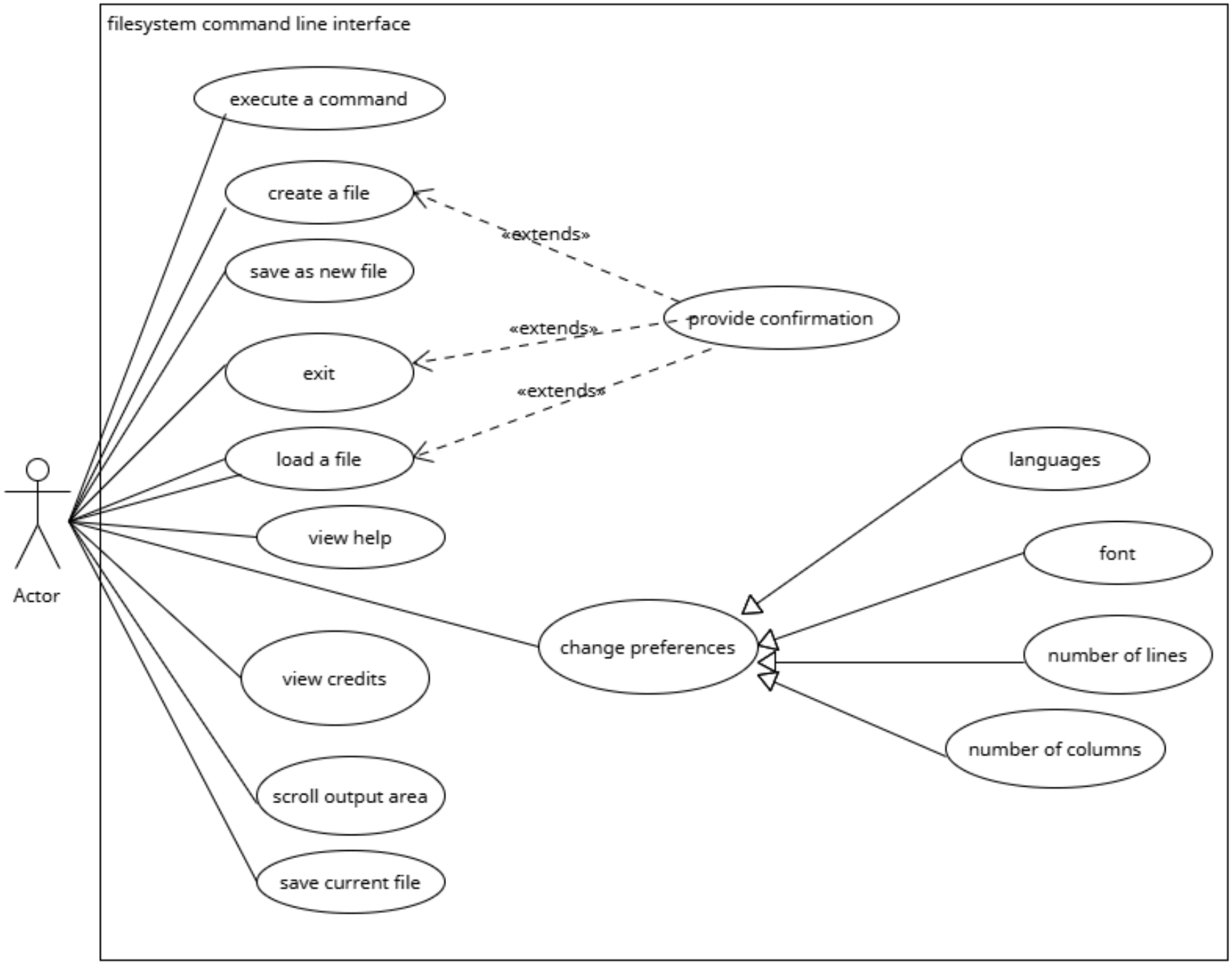
Requisiti Funzionali

- **Core Commands:** ls, mkdir, cd, mv, rm, ln [-s], touch, etc... .
- **Navigazione:** Supporto path assoluti, relativi e speciali (./ ../).
- **Wildcards:** Supporto espansione carattere * (es. ls *).
- **Persistenza:** Salvataggio stato su file JSON.
- **Configurazione:** Preferenze utente (lingua, font).

Vincoli non Funzionali

- Interfaccia Grafica **JavaFX**.
- Architettura **MVC** pura.
- **Testing** estensivo (Unit&Integration).
- Distribuzione via **JAR** self-contained.
- Localizzazione (**I18N**).

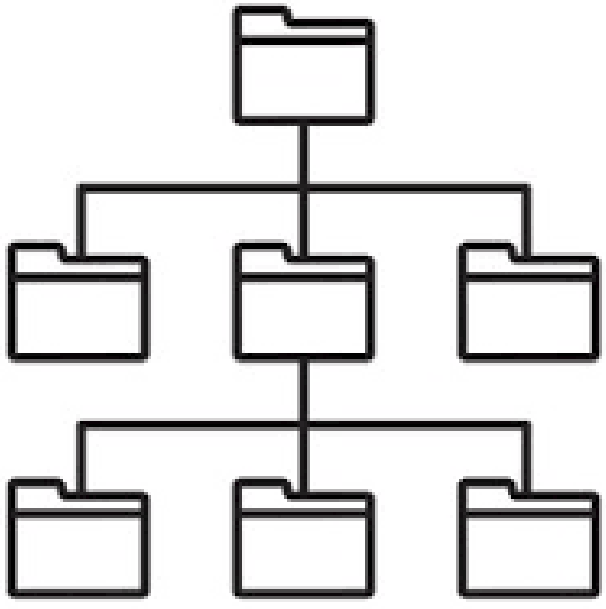
Use Case Diagram



Stato dell'arte

- Google **Jimfs** - In-Memory Filesystems (Java)
- **JSLinux** - Emulazione su Browser (JavaScript)
- **Cowrie** - Honeypots (Python)

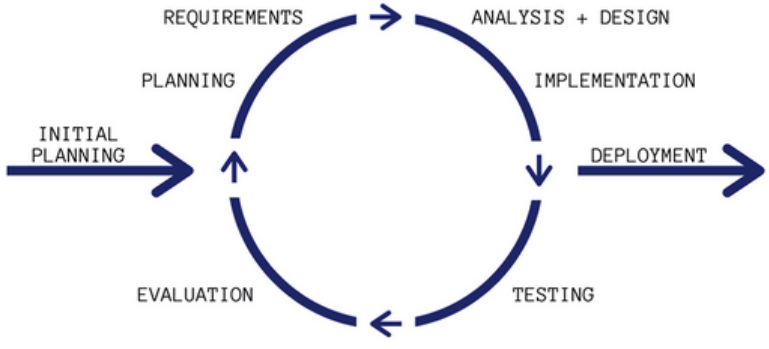
Solo alcune permettono l’opzione salvataggio e la gestione delle preferenze.

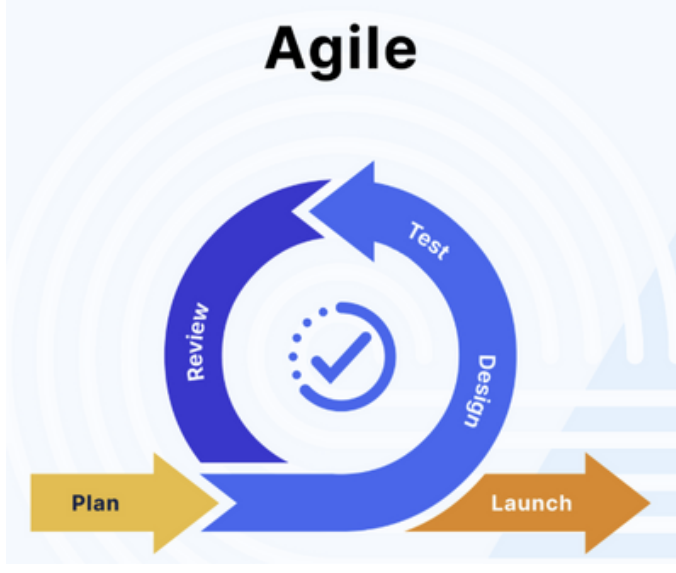


Approccio

1. Stack Tecnologico
   - **Linguaggio**: Java 17
   - **Grafica**: JavaFx
   - **Dipendenze**: Jackson - Guice - Mockito - TestFx
   - **Versioning**: GitLab

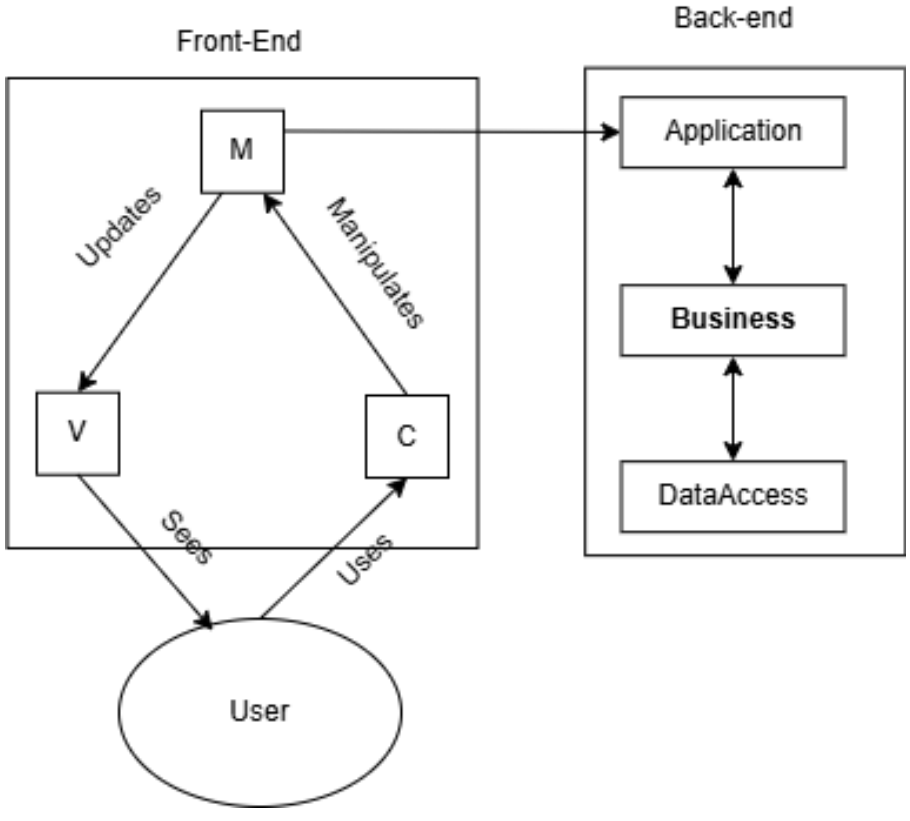
2. Approccio di sviluppo:
   - **Agile**
   - **Durata Sprint:** 2 settimane





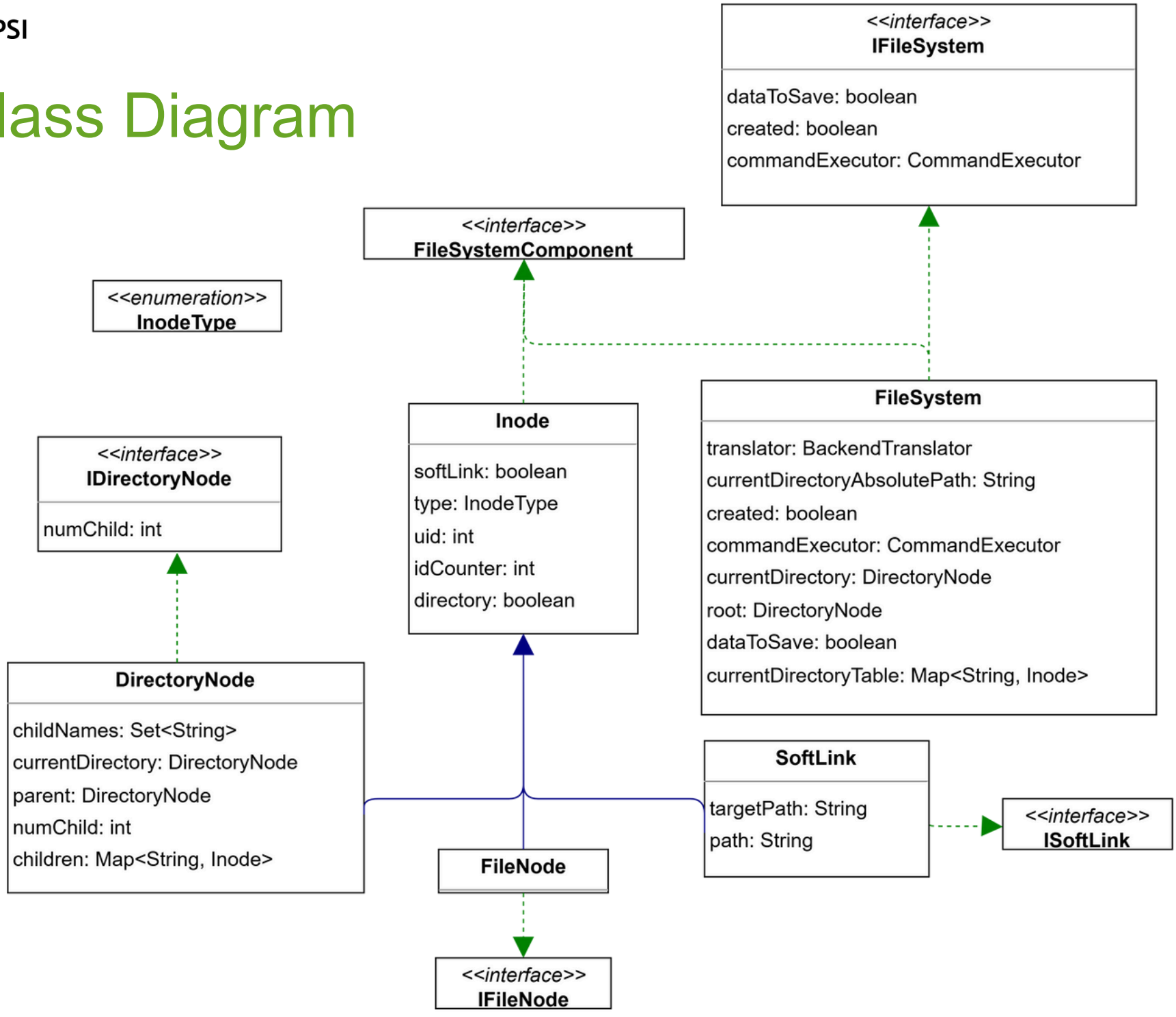
Approccio

3. Architettura: **divisione front-end e back-end**
   - il **Frontend** utilizza l'architettura MVC.
   - il **Backend** utilizza l'architettura a layer.

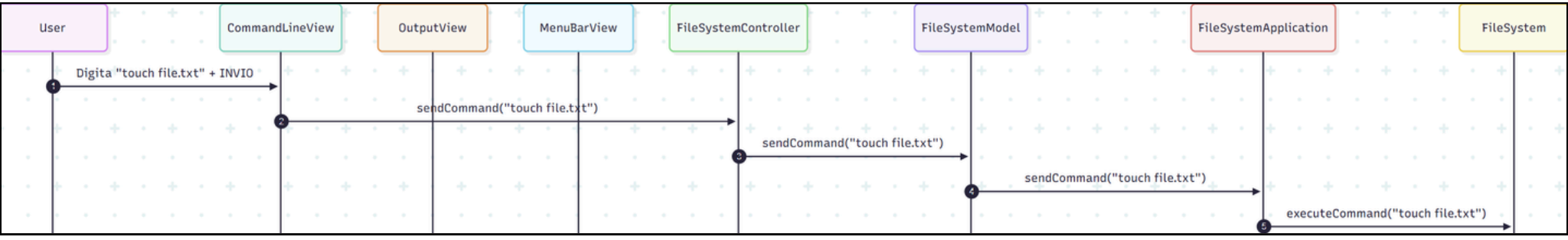


4. Distribuzione ed Esecuzione
   Artifact: JAR (con dipendenze incluse) su Maven.
   Esecuzione: JAR eseguibile con java -jar.

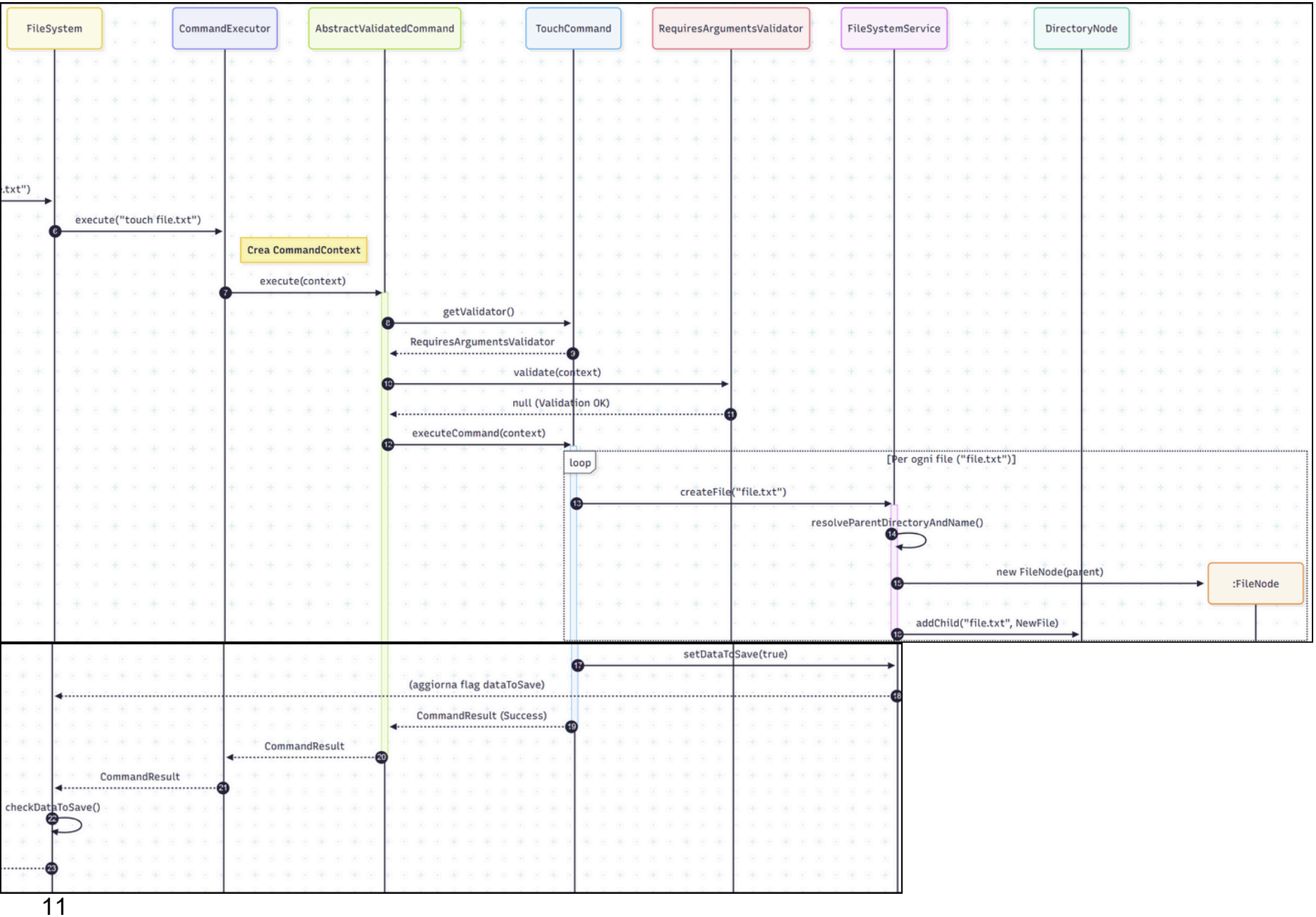
Class Diagram



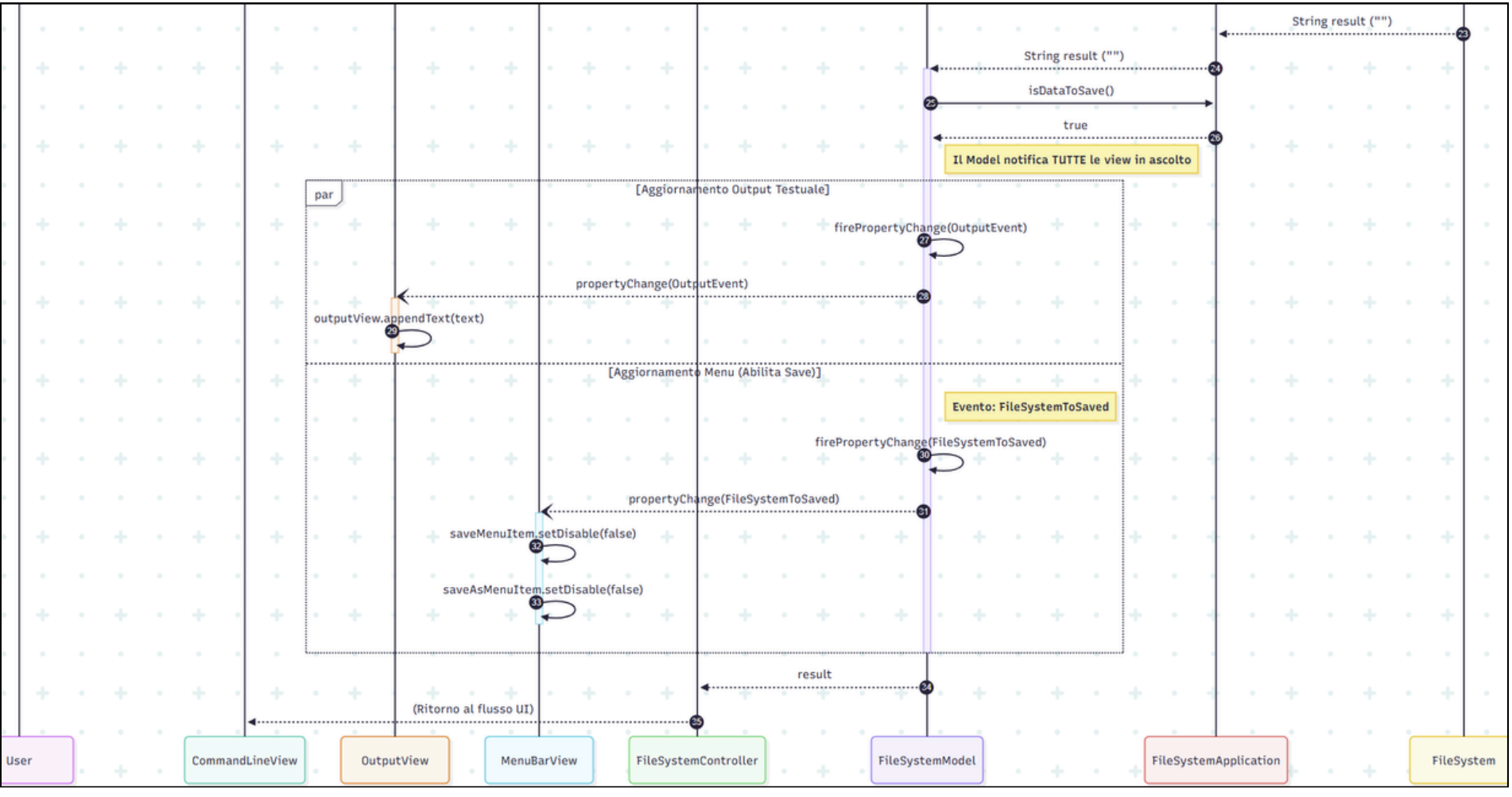
Sequence Diagram – Comando Touch



Sequence Diagram – Comando Touch



Sequence Diagram – Comando Touch



Risultati Raggiunti

- **Copertura completa dei requisiti**
  - Filesystem completo e funzionante
  - Salvataggio
  - Caricamento
  - Modifiche preferenze da GUI
  - Menù help
  - Crediti
  - Messaggi di log
- **Architettura disaccoppiata e scalabile**

Conclusioni

🧠 Cosa abbiamo imparato

- Abbiamo messo in pratica concetti fondamentali di **ingegneria del software II**: gestione dei requisiti, design, sviluppo e testing.
- Abbiamo collaborato efficacemente in team, anche nella **divisione dei compiti e nell'integrazione del codice**.

📉 Difficoltà incontrate

- Capire come strutturare il filesystem
- Implementazione dei design pattern corretti
- Testing