SAfeGuard

Object Design Document

Riferimento: C08_ODD_1.0

Versione: 1.0

Data: 14 dicembre 2025

Destinatario: Prof.ssa Filomena Ferrucci, Prof.re Fabio Palomba

Presentato da: C08 - SAfeGuard

Documento generato il 14 dicembre 2025

Team Members

Nome	**Ruolo**	**Acronimo**	**Contatto**
Giuseppe Napolitano	Project Manager	GN	g.napolitano80@studenti.unisa.it
Pasquale Sorrentino	Project Manager	PS	p.sorrentino47@studenti.unisa.it
Alessandro Amendola	Team Member	AA	a.amendola51@studenti.unisa.it
Alessandro Masone	Team Member	AM	a.masone1@studenti.unisa.it
Antonello Castelluccio	Team Member	AC	a.castelluccio15@studenti.unisa.it
Francesco Carbone	Team Member	FC	f.carbone50@studenti.unisa.it
Francesco Zambrino	Team Member	FZ	f.zambrino@studenti.unisa.it
Gianpaolo Aquilone	Team Member	GA	g.aquilone1@studenti.unisa.it
Giorgio Zazzerini	Team Member	GZ	g.zazzerini@studenti.unisa.it
Giovanni Lamberti	Team Member	GL	g.lamberti55@studenti.unisa.it
Matteo Manganiello	Team Member	MM	m.manganiello15@studenti.unisa.it
Thomas Mercadino	Team Member	TM	t.mercandino@studenti.unisa.it
Victor Di Gennaro	Team Member	VDG	v.digennaro5@studenti.unisa.it

Revision History

Data	Ver.	Descrizione	Autori
20/11/25	0.1	Stesura Design Patterns	FC
13/12/25	1.0	Rivisitazione completa	Team

Indice

Capitolo 1

Design Patterns

1.1 Facade

Il Facade è un design pattern particolarmente adatto a un sistema complesso come SAfeGuard, dove la gestione di diversi sottosistemi richiede un alto livello di coordinazione.

SAfeGuard offre diverse funzionalità, come la gestione delle emergenze, la geolocalizzazione in tempo reale, la gestione delle notifiche a numerosi dispositivi, ognuna delle quali coinvolge molteplici classi che collaborano per implementarne i servizi. La complessità dei sottosistemi interni rende fondamentale un'architettura che isoli e semplifichi l'utilizzo di suddette funzionalità.

In questo scenario, il Facade funge da punto di accesso semplificato ai sottosistemi, mascherando i dettagli interni. Questo riduce l'accoppiamento tra i controller API e le classi interne, permettendo alla logica di controllo di ignorare la complessità sottostante. Il risultato è un codice più pulito e facile da mantenere.

Il pattern migliora anche la modularità: ogni sottosistema gestisce le operazioni tramite il proprio Facade, facilitando testing e aggiornamenti. Grazie a questa interfaccia stabile, è possibile modificare le classi interne senza rischiare regressioni o effetti collaterali sugli altri moduli del sistema.

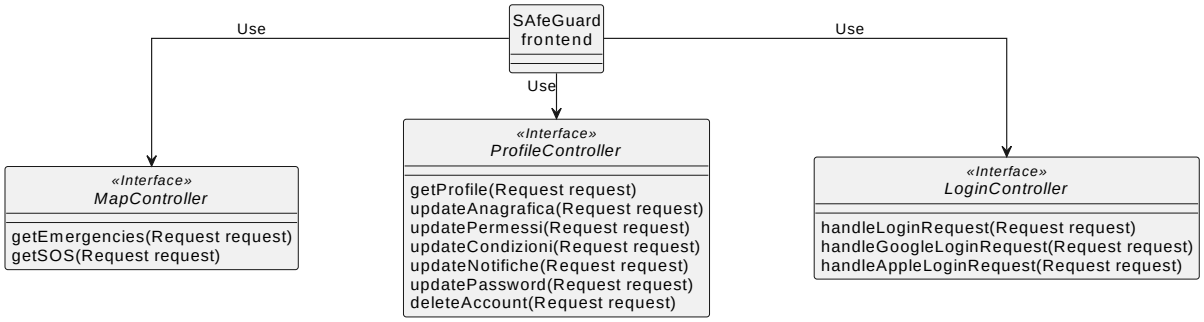

Figura 1.1: Diagramma pattern Facade: SAfeGuard Frontend e Controller

1.2 Adapter

Il design pattern di tipo Adapter può rivelarsi molto utile per integrare in SAfeGuard componenti esterne al sistema, in particolare il modulo di intelligenza artificiale che consente di suddividere la

zona di Salerno in sotto-aree in base all'evento catastrofico e di individuare le zone maggiormente impattate o a rischio.

Essendo che il modulo di AI sarà sviluppato con tool specifici esterni al linguaggio nativo di SAfeGuard (flutter/dart) è necessario un Adapter che consente di convertire l'interfaccia del modulo AI in una interfaccia diversa con cui SAfeGuard possa interagire nonostante le due implementazioni siano di base incompatibili.

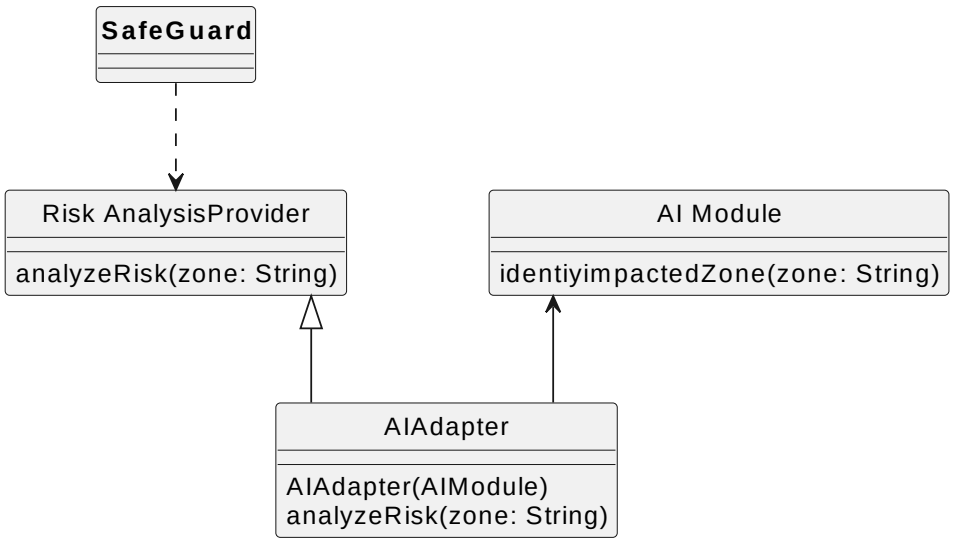

Figura 1.2: Diagramma pattern Adapter per il Modulo AI