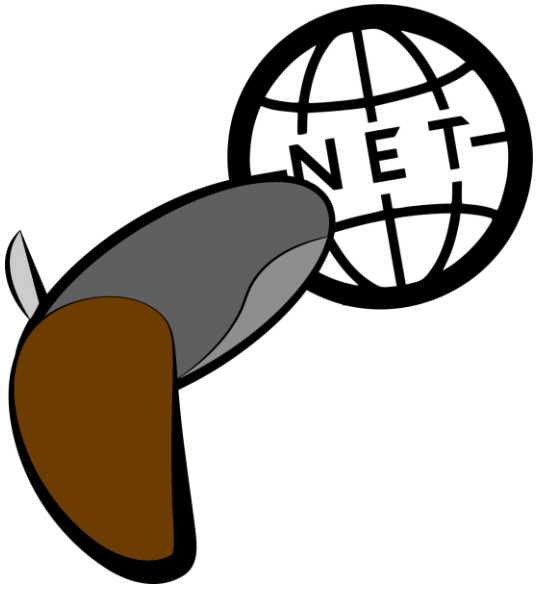


ODD
Object Design Document

NetGun

Versione	0.6
Data	02/02/2023
Destinatario	Professore Carmine Gravino
Presentato da	Carlo Colizzi, Giulio Incoronato, Antonio Mazzarella

Team Members

Nome	Informazioni di contatto
Carlo Colizzi	c.colizzi@studenti.unisa.it
Giulio Incoronato	g.incoronato2@studenti.unisa.it
Antonio Mazzarella	a.mazzarella5@studenti.unisa.it

Revision History

Data	Versione	Descrizione	Autori
15/01/2023	0.1	Stesura della sezione Revision History, Sommario e Team Members	**Giulio Incoronato**
15/01/2023	0.2	Stesura del paragrafo 1 e completamento	**Carlo Colizzi**
16/01/2023	0.3	Completamento del paragrafo 2	**Giulio Incoronato**
18/01/2023	0.4	Definizione delle class interfaces e completamento del paragrafo 3	**Giulio Incoronato**
19/01/2023	0.5	Class Diagram ristrutturato inserito e paragrafo 4 completato	**Carlo Colizzi**
20/01/2023	0.6	Definizione dei Design Pattern utilizzati e Paragrafo 5, 6	**Carlo Colizzi**

Sommario

1 Introduzione

Il sistema NetGun si propone come tool per l'analisi di infrastrutture di rete, con lo scopo di assistere i Penetration Tester e i Network Analyst durante le attività di ricognizione sugli assets IT, facilitando l'individuazione di possibili problematiche relative alla sicurezza di queste infrastrutture.

In questa prima parte del documento vengono descritti gli object design goals e i relativi trade-offs, inoltre vengono stabilite delle convenzioni da attuare in fase di implementazione.

1.1 Object design Trade-offs and Goals

Spazio in memoria vs Tempo di risposta:

Tutti i dati persistenti saranno caricati nella memoria principale all'Avvio del programma, così da garantire un tempo di risposta immediato, a discapito della memoria occupata.

Affidabilità vs Tempo:

Il sistema deve garantire un'affidabilità dei dati raccolti, questa è ottenuta a discapito del tempo. È necessario più tempo affiche' i dati raccolti siano riscontrati e accertati.

Incapsulamento:

Il sistema garantisce il mascheramento dei dettagli implementativi di basso livello attraverso l'uso di interfacce.

1.2 Linee guida per la Documentazione delle Interfacce

Link utili alle convenzioni adottate per l'implementazione:

- **Python Doc:** https://docs.python.org/3.10/
- **Nmap API for Python:** https://pypi.org/project/python-nmap/
- **NVD API:** https://nvd.nist.gov/developers/vulnerabilities
- **CustomTkinter:** https://pypi.org/project/customtkinter/0.3/

1.3 Definizioni, acronimi e abbreviazioni

Pydoc: sistema per la creazione automatica della documentazione attraverso commenti del codice Python

Design Pattern: template di soluzioni a problemi ricorrenti, permettono di standardizzare e semplificare le scelte progettuali.

Package: raggruppamento di classi, interfacce e files.

Interfaccia: insieme di signature rappresentanti le operazioni offerte da una classe.

1.4 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:

- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Object Design Document (ODD)
- Test Plane (TP)
- Test Case Specification (TCS)
- Codice Sorgente
- Matrice di tracciabilità
- Il Documento segue le metodologie presentate nel libro: Object-Oriented Software Engineering, di Bernd Bruegge & Allen H. Dutoit

2 Packages

In questa sezione è mostrata la suddivisione del sistema in package, in base ai sottosistemi definiti in fase di System Design.

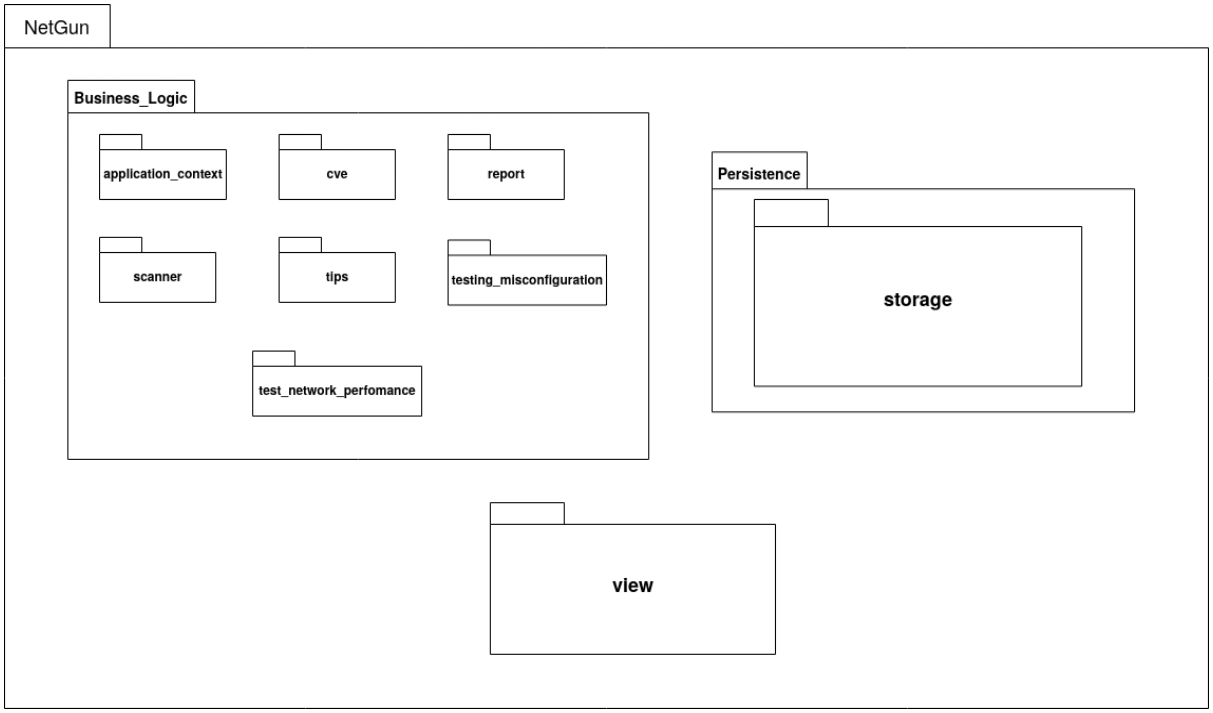
source_code: Package root del codice sorgente

Nella source_code troviamo:

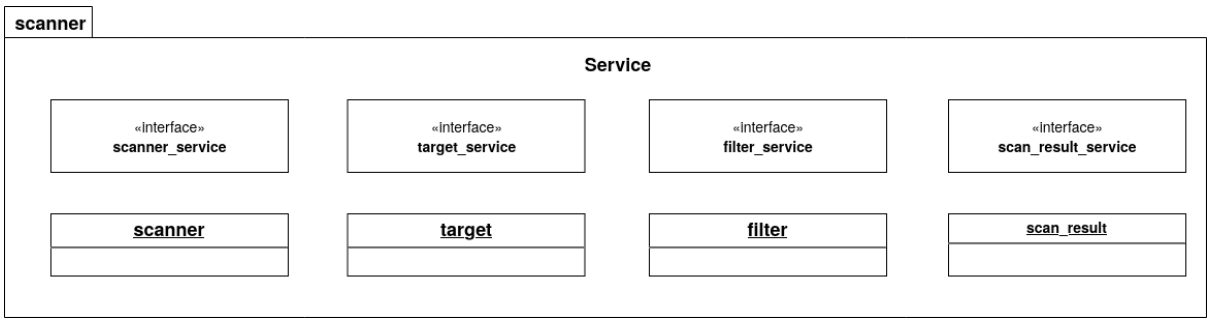
- **business_logic** : contiene le classi relative alla Business Logic
  - **application_context**
  - **cve**
  - **report**
  - **scanner**
  - **test_network_performance**
  - **testing_misconfigurations**
  - **tips**
- **persistence** : contiene le classi relative alla gestione della Persistenza
  - **storage**: contiene i file di persistenza (Dati Strutturati in file xml, file xml serializzati, icone, immagini…)
- **view** : contiene le classi relative alla View

2.1 Packages

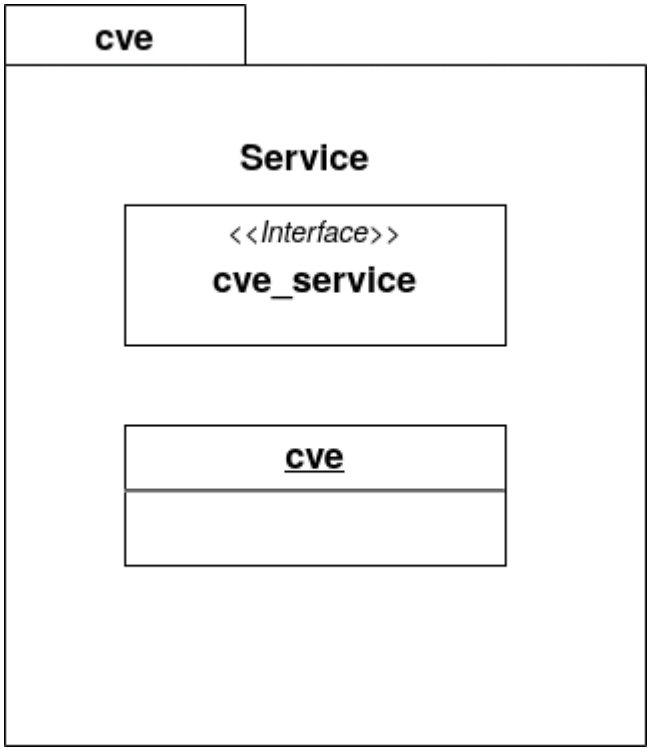
In questa sezione si mostra la struttura del package principale di NetGun, il package “source_code”. La struttura generale è stata costruita tenendo conto della struttura 3 Thier desiderata per questo sistema.



2.1.1 Package scanner

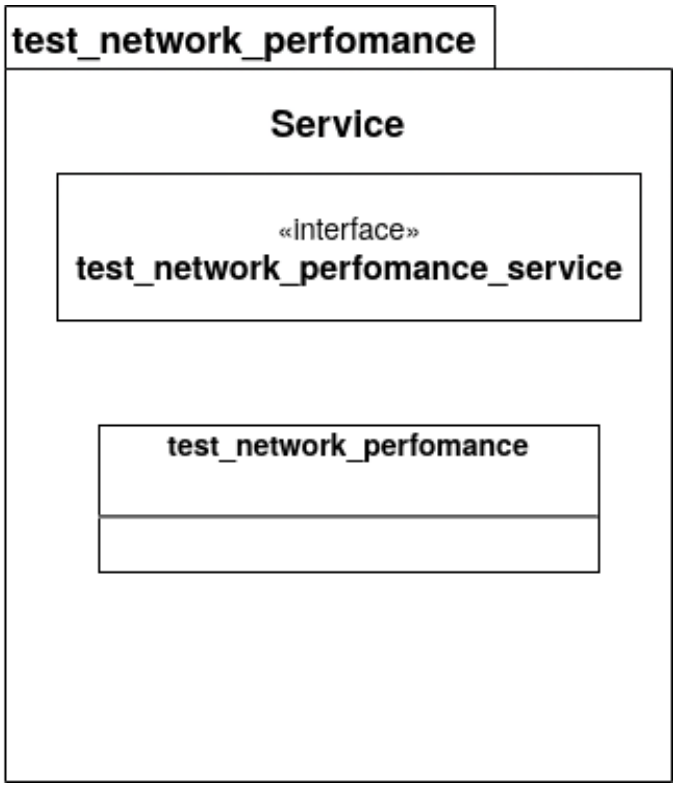


2.1.2 Package CVE

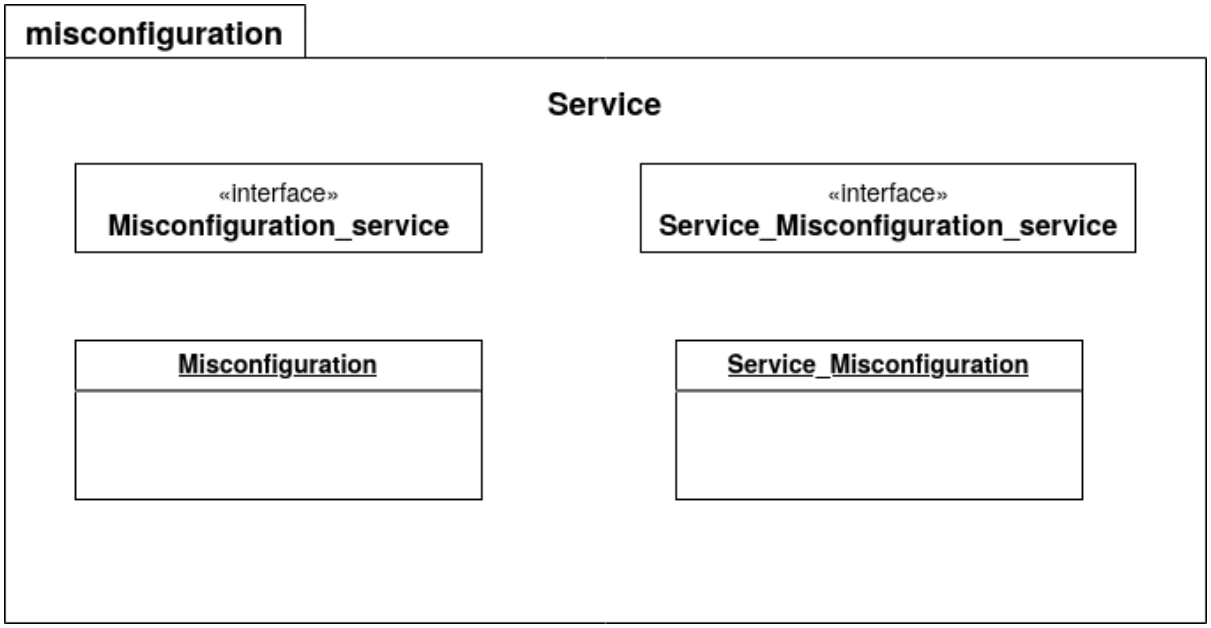


2.1.3 Package report

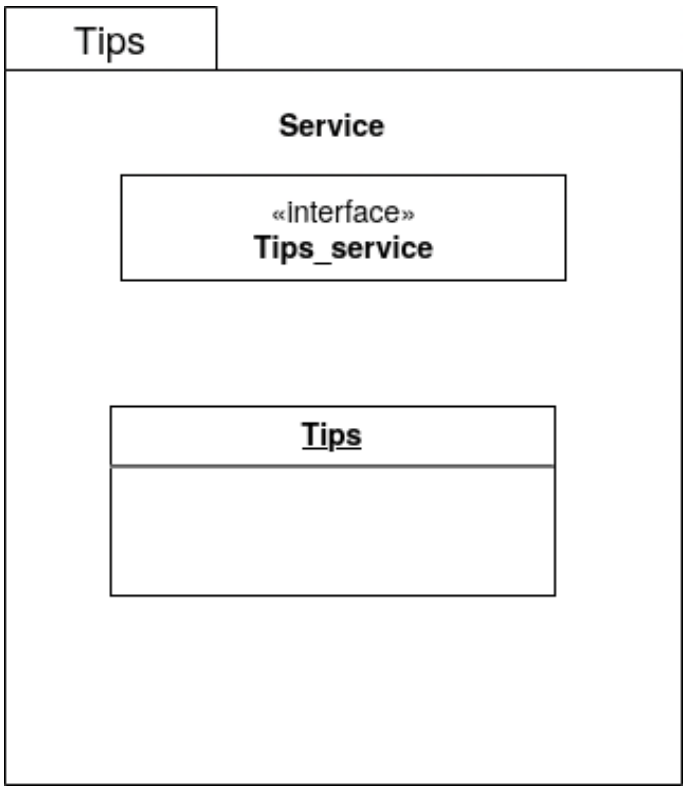
2.1.4 Package test_network_performance



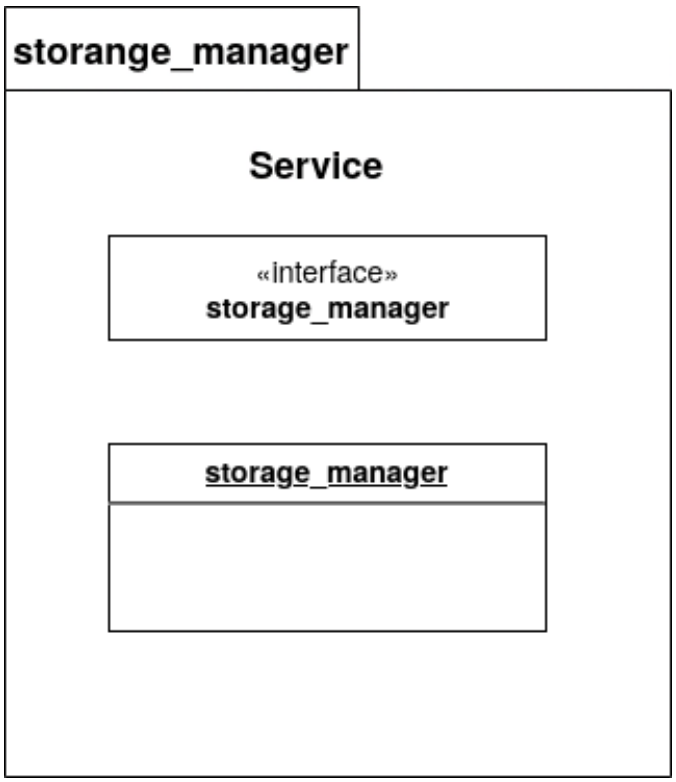
2.1.5 Package misconfigurations



2.1.6 Package tips



2.1.7 Package storage_manager



3 Class Interfaces

Di seguito saranno presentate le interfacce di ciascun package.

3.1 Package scanner

PATH:

- **source_code.business_logic.scanner.Scan**

Nome classe	Scan
Descrizione	Questa classe permette effettuare uno scanner su un Target
Metodi	+ **start_scan**(self): dict + **parse_result_shallow**(self, result: dict): dict + **parse_result_deep**(self, result: dict): dict

Nome Metodo	+ **start_scan**(self): Dict
Descrizione	Questo metodo permette di avviare lo scan
Pre-condizione	**context:** Scan::start_scan(self) **pre:** Target != null && Filter != null && scan_mode != ‘ ’
Post-condizione	**/**

Nome Metodo	+ **parse_result_shallow**(self, result:dict):dict
Descrizione	Questo metodo converte il risultato dello shallow scan in un dizionario in Python

Pre-condizione	**context:** Scan::parse_result_shallow(self, result:dict):dict **pre:** result != null
Post-condizione	/

Nome Metodo	+ **parse_result_deep**(self, result:dict):dict
Descrizione	Questo metodo converte il risultato del deep scan in un dizionario in Python
Pre-condizione	**context:** Scan::parse_result_deep(self, result:dict):dict **pre:** result != null
Post-condizione	/

PATH:

- **source_code.business_logic.scanner.Filter**

Nome classe	Filter
Descrizione	Questa classe permette di impostare i filtri per lo scan
Metodi	+ **advanced_options_to_string**(self, advanced_options : list[str]): str + **advanced_option_convert**(self, advanced_options : list[str]): list[str] + **check_aggressivity**(cls, aggressivity : int): boolean

Nome Metodo	+ **advanced_options_to_string**(self, advanced_options : list[str]): str

Descrizione	Questo metodo converte la lista di opzioni avanzate in una stringa
Pre-condizione	**context:** Scan::advanced_options_to_string(self, advanced_options : list[str]): str **pre:** /
Post-condizione	/

Nome Metodo	+ **advanced_option_convert**(self, advanced_options : list[str]): list[str]
Descrizione	Questo metodo converte una stringa di advanced_options in una lista
Pre-condizione	**context:** Scan::advanced_option_convert(self, advanced_options : list[str]): list[str] **pre:** /
Post-condizione	/

Nome Metodo	+ **check_aggressivity**(cls, aggressivity : int): boolean
Descrizione	Questo metodo verifica se l'aggressività scelta per lo scanning è appropriata
Pre-condizione	**context:** Scan::check_aggressivity(cls, aggressivity : int): boolean **pre:** aggressivity != null
Post-condizione	/

PATH:

- **source_code.business_logic.scanner.Target**

Nome classe	Target
Descrizione	Questa classe permette di impostare il target della scansione
Metodi	+ **check_ip**(cls, ip: str): boolean + **check_ports**(cls, ports:str): boolean

Nome Metodo	+ **check_ip**(cls, ip: str): boolean
Descrizione	Questo metodo controlla la validità dell'IP
Pre-condizione	**context:** Scan::check_ip(cls, ip: str): boolean **pre:** ip != null
Post-condizione	/

Nome Metodo	+ **check_ports**(cls, ports:str): boolean
Descrizione	Questo metodo controlla la validità del range di porte
Pre-condizione	**context:** Scan::check_ports(cls, ports:str): boolean **pre:** ports != null
Post-condizione	/

3.2 Package cve

PATH:

- **source_code.business_logic.cve.Cve**

Nome classe	Cve
Descrizione	Questa classe permette di ricercare le vulnerabilità di un servizio
Metodi	+ **search_cve**(self): dict + **get_vulnerabilities**(self, data: dict): dict

Nome Metodo	+ **search_cve**(self):dict
Descrizione	Questo metodo avvia la ricerca delle vulnerabilità di un servizio
Pre-condizione	**context:** Cve::search_cve(self):dict **pre:** version != null
Post-condizione	/

Nome Metodo	+ **get_vulnerabilities**(cls, data:dict):dict
Descrizione	Questo metodo restituisce un dizionario con le vulnerabilità di un servizio
Pre-condizione	**context:** Cve::get_vulnerabilities(cls, data:dict):dict **pre:** version != null && data != null

Post-condizione	/

3.3 Package report

PATH:

- **source_code.business_logic.report.Report**

Nome classe	Report
Descrizione	Questa classe crea un report in formato PDF, contenente un sommario delle informazioni raccolte dallo scanner e dalla ricerca di CVE
Metodi	+**create_report**(self, result_scan: dict) +**create_table**(cls, results: dict, key: str): list +**create_vertical_table**(cls, elements: dict, element:str): list +**create_horizontal_table**(cls, elements:dict, element:str): list +**draw_table_pdf**(cls, table_data: list, title:str , title_size: int, data_size: int, align_data: str, align_header: str, emphasize_data: list, emphasize_headers: list, emphasize_header_color: (), emphasize_data_color:(), emphasize_header_style: str, emphasize_data_style:str, type_table:str, pdf): FPDF + **add_horizontal_table_headers**(cls, headers: list, pdf: FPDF, col_width: float, line_height: float, align_header, emphasize_headers: list, emphasize_header_color: (), emphasize_header_style:str): FPDF + **add_horizontal_table_data**(cls, data: list, pdf: FPDF(), col_width, line_height, align_data, emphasize_data: list, emphasize_data_color: (), emphasize_data_style: str): FPDF + **add_vertical_table_data**(cls, data: list, pdf: FPDF, col_width: float, line_height: float, align_data: (), emphasize_data: list, emphasize_data_color: (), emphasize_data_style: str): FPDF + **get_indexes**(cls, check_string: str, table_data:list):

	list

Nome Metodo	+ **create_report**(cls, result_scan:dict):dict
Descrizione	Questo metodo crea un report in PDF con i risultati dello scan
Pre-condizione	**context:** Report::create_report(cls, result_scan:dict):dict **pre:** result_scan != null && filename != “ ”
Post-condizione	/

Nome Metodo	+ **create_table**(cls, result:dict, key:str):list
Descrizione	Questo metodo restituisce una lista di liste
Pre-condizione	**context:** Cve::create_table(cls, result:dict, key:str):dict **pre:** result != null && key != null || key != “ “
Post-condizione	/

Nome Metodo	+ **create_vertical_table**(cls, elements: dict, element:str): list
Descrizione	Questo metodo restituisce una lista di liste con due elementi dove il primo elemento è un header e il secondo elemento un valore
Pre-condizione	**context:** Report::create_vertical_table(cls, elements:dict, element:str):list **pre:** elements != null && element != null

Post-condizione	/

Nome Metodo	+ **create_horizontal_table**(cls, elements: dict, element:str): list
Descrizione	Questo metodo restituisce una lista di liste: la prima lista conterrà gli header, mentre le restanti tutti i valori
Pre-condizione	**context:** Report::create_horizontal_table(cls, elements:dict, element:str):list **pre:** elements != null && element != null
Post-condizione	/

Nome Metodo	+ **draw_table_pdf**(cls, table_data: list, title:str , title_size: int, data_size: int, align_data: str, align_header: str, emphasize_data: list, emphasize_headers: list, emphasize_header_color: (), emphasize_data_color:(), emphasize_header_style: str, emphasize_data_style:str, type_table:str, pdf): FPDF
Descrizione	Questo metodo crea una tabella personalizzata per il pdf
Pre-condizione	**context:** Report::**draw_table_pdf**(cls, table_data: list, title:str , title_size: int, data_size: int, align_data: str, align_header: str, emphasize_data: list, emphasize_headers: list, emphasize_header_color: (), emphasize_data_color:(), emphasize_header_style: str, emphasize_data_style:str, type_table:str, pdf): FPDF **pre:** table_data != null

Post-condizione	/

Nome Metodo	+ **add_horizontal_table_headers**(cls, headers: list, pdf: FPDF, col_width: float, line_height: float, align_header, emphasize_headers: list, emphasize_header_color: (), emphasize_header_style:str): FPDF
Descrizione	Questo metodo restituisce un oggetto fpdf con tabella riempita di header dello scan.
Pre-condizione	**context:** Report::**add_horizontal_table_headers**(cls, headers: list, pdf: FPDF, col_width: float, line_height: float, align_header, emphasize_headers: list, emphasize_header_color: (), emphasize_header_style:str): FPDF **pre:** headers != null && pdf != null
Post-condizione	/

Nome Metodo	+ **add_horizontal_table_data**(cls, data: list, pdf: FPDF(), col_width:float, line_height:float, align_data, emphasize_data: list, emphasize_data_color: (), emphasize_data_style: str): FPDF
Descrizione	Questo metodo restituisce un oggetto fpdf con tabella riempita di dati dello scan.
Pre-condizione	**context:** Report::**add_horizontal_table_data**(cls, data: list, pdf: FPDF(), col_width, line_height, align_data, emphasize_data: list, emphasize_data_color: (), emphasize_data_style: str): FPDF

	pre: data != null && pdf != null
Post-condizione	/

Nome Metodo	+ **add_vertical_table_data**(cls, data: list, pdf: FPDF, col_width: float, line_height: float, align_data: (), emphasize_data: list, emphasize_data_color: (), emphasize_data_style: str): FPDF
Descrizione	Questo metodo restituisce un oggetto fpdf con i dati dello scan
Pre-condizione	**context:** Report::**add_vertical_table_data**(cls, data: list, pdf: FPDF, col_width: float, line_height: float, align_data: (), emphasize_data: list, emphasize_data_color: (), emphasize_data_style: str): FPDF **pre:** data != null && pdf != null
Post-condizione	/

Nome Metodo	+ **get_indexes**(cls, check_string: str, table_data:list): list
Descrizione	Questo metodo restituisce una lista di numeri che indicano la posizione in cui si trova una stringa in un

	dizionario
Pre-condizione	**context:** Report::**get_indexes**(cls, check_string: str, table_data:list): list **pre:** check_string != null & table_data != null
Post-condizione	/

3.4 Package test_network_performance

PATH:

- **source_code.business_logic.test_network_perfomance.Network_test**

Nome classe	Network_test
Descrizione	Questa classe permette di fare un test sulle prestazioni della rete
Metodi	+ **test_download**(self): float + **test_upload**(self): float + **bytes_to_mbps**(cls, bytes: float): floats

Nome Metodo	+ **test_download**(self):float
Descrizione	Questo metodo avvia un test di download
Pre-condizione	**context:** Network_test::test_download(self):float **pre:** network != null
Post-condizione	/

Nome Metodo	+ **test_upload**(self):float
Descrizione	Questo metodo avvia un test di upload
Pre-condizione	**context:** Network_test::test_upload(self):float **pre:** network != null

Post-condizione	/

Nome Metodo	+ **bytes_to_mbps**(cls, bytes:float):float
Descrizione	Questo metodo converte il risultato del test il mbps
Pre-condizione	**context:** Network_test::bytes_to_mbps(cls, bytes:float):float **pre:** network != null
Post-condizione	/

3.5 Package misconfigurations

PATH:

- **source_code.business_logic.testing.misconfigurations.Misconfiguration**

Nome classe	Misconfiguration
Descrizione	Questa classe permette di testare mal configurazioni
Metodi	+ **print_toString**(self) + **test_misconfiguration**(self)

Nome Metodo	+ **print_toString**(self)
Descrizione	Questo metodo stampa tutti le variabili dell'oggetto
Pre-condizione	**context:** Misconfiguration::print_toString(self) **pre:** misconfiguration != null
Post-condizione	/

Nome Metodo	+ **test_misconfiguration**(self)
Descrizione	Questo metodo testa le mal configurazioni
Pre-condizione	**context:** Misconfiguration::test_misconfiguration(self) **pre:** /
Post-condizione	/

PATH:

- **source_code.business_logic.testing.misconfigurations.Service_Misconfiguration**

Nome classe	Service_Misconfigurations
Descrizione	Questa classe permette di utilizzare mal configurazioni salvate in un file xml
Metodi	+ **parse_misconfigurations_from_file**(cls, path) + **inizialize_object_from_file_xml**(cls, path)

Nome Metodo	+ **parse_misconfigurations_from_file**(cls, path): dict
Descrizione	Questo metodo converte le mal configurazione da un file xml in un dizionario
Pre-condizione	**context:**Service_Misconfiguration::parse_misconfigurations_from_file(cls, path) **pre:** path != null
Post-condizione	/

Nome Metodo	+ **inizialize_object_from_file_xml**(cls, path): dict
Descrizione	Questo metodo permette di creare un oggetto services_misconfiguration da un file xml
Pre-condizione	**context:**Service_Misconfiguration::inizialize_object_from_file_xml(cls, path) **pre:** /
Post-condizione	/

3.6 Package tips

PATH:

- **source_code.business_logic.testing.tips.tip**

Nome classe	tip
Descrizione	Questa classe permette di raccogliere i tip da un file xml
Metodi	+ **print_toString**(self) + **parse_tips_from_file**(cls, path)

Nome Metodo	+ **print_toString**(self)
Descrizione	Questo metodo permette di stampare tutte le variabili dell'oggetto tip

Pre-condizione	**context:**Tip::print_toString(self) **pre:** tip != null
Post-condizione	/

Nome Metodo	+ **parse_tips_from_file**(cls, path)
Descrizione	Questo metodo permette di convertire i dati da un file per creare un oggetto tip
Pre-condizione	**context:**Tip::parse_tips_from_file(cls, path) **pre:** path != null
Post-condizione	/

3.7 Package storage_manager

PATH:

- **source_code.persistence.storage_manager**

Nome classe	Storage_manager
Descrizione	Questa classe permette di comunicare con lo storage
Metodi	+ **load_resource**(cls, xml_path, pkl_path, create_object_from_file): + **deserialize_object_from_file**(cls,file): + **serialize_object_into_file**(cls,obj,file): + **load_misconfigurations**(cls):

	+ **load_tips**(cls):

Nome Metodo	+ **load_resource**(cls, xml_path, pkl_path, create_object_from_file):
Descrizione	Questo metodo permette di convertire dati da un file xml, per creare una lista di oggetti
Pre-condizione	**context:**Storage_manager::load_resource(cls, xml_path, pkl_path, create_object_from_file): **pre:** xml_path != null && pkl_path != null && create_object_from_file != null
Post-condizione	/

Nome Metodo	+ **deserialize_object_from_file**(cls,file):
Descrizione	Questo metodo permette di deserializzare un file serializzato, ottenendo i valori al suo interno
Pre-condizione	**context:**Storage_manager::deserialize_object_from_file(cls,file): **pre:** file != null && file != ‘’
Post-condizione	/

Nome Metodo	+ **serialize_object_into_file**(cls,obj,file):
Descrizione	Questo metodo permette di serializzare degli oggetti in un file
Pre-condizione	**context:**Storage_manager::serialize_object_into_file

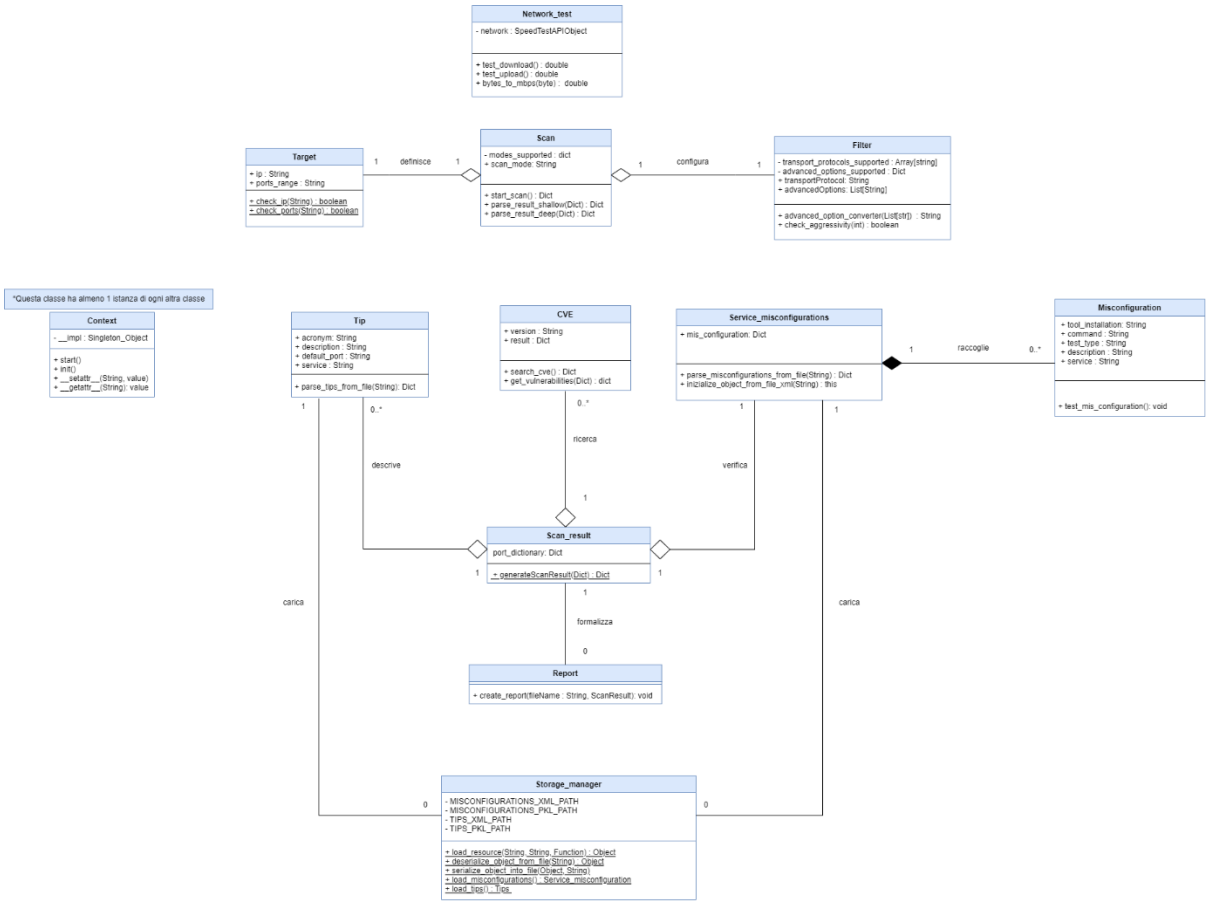
	(cls,obj,file): **pre:** obj != null && file != null
Post-condizione	/

Nome Metodo	+ **load_misconfigurations**(cls):
Descrizione	Questo metodo permette di ottenere le misconfigurazioni presenti nella persistenza, astraendo la logica implementativa
Pre-condizione	**context:**Storage_manager::load_misconfigurations(cls): **pre:** /
Post-condizione	/

Nome Metodo	+ **load_tips**(cls):
Descrizione	Questo metodo permette di ottenere i tips presenti nella persistenza, astraendo la logica implementativa
Pre-condizione	**context:**Storage_manager::load_tips(cls): **pre:** /
Post-condizione	/

4 Class Diagram

- Il nome degli attributi e dei metodi è stato modificato, precedentemente erano in formato CamelCase, ora, rimanendo in linea con quanto scritto nell' SDD, sarà utilizzato lo SnakeCase
- Sono stati aggiunti attributi nelle classi Filter e Scan
- Sono state aggiunte le classi Storage_manager e Context, necessarie per la corretta esecuzione del software
- Sono stati modificati i tipi utilizzati, quelli attuali sono più descrittivi e specifici
- Sono stati aggiunti ulteriori modificatori di accesso, e modificati quelli già esistenti



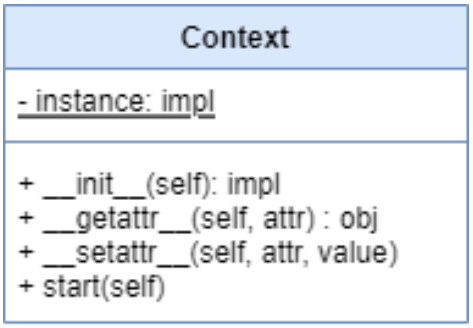
5 Design Patterns

Nella presente sezione si andranno a descrivere e dettagliare i design patterns utilizzati nello sviluppo dell'applicativo NetGun. Per ogni pattern si darà:

- Una breve introduzione teorica.
- Il problema che doveva risolvere in NetGun.
- Una breve spiegazione di come si è risolto il problema in NetGun.
- Un grafico della struttura delle classi che implementano il pattern.

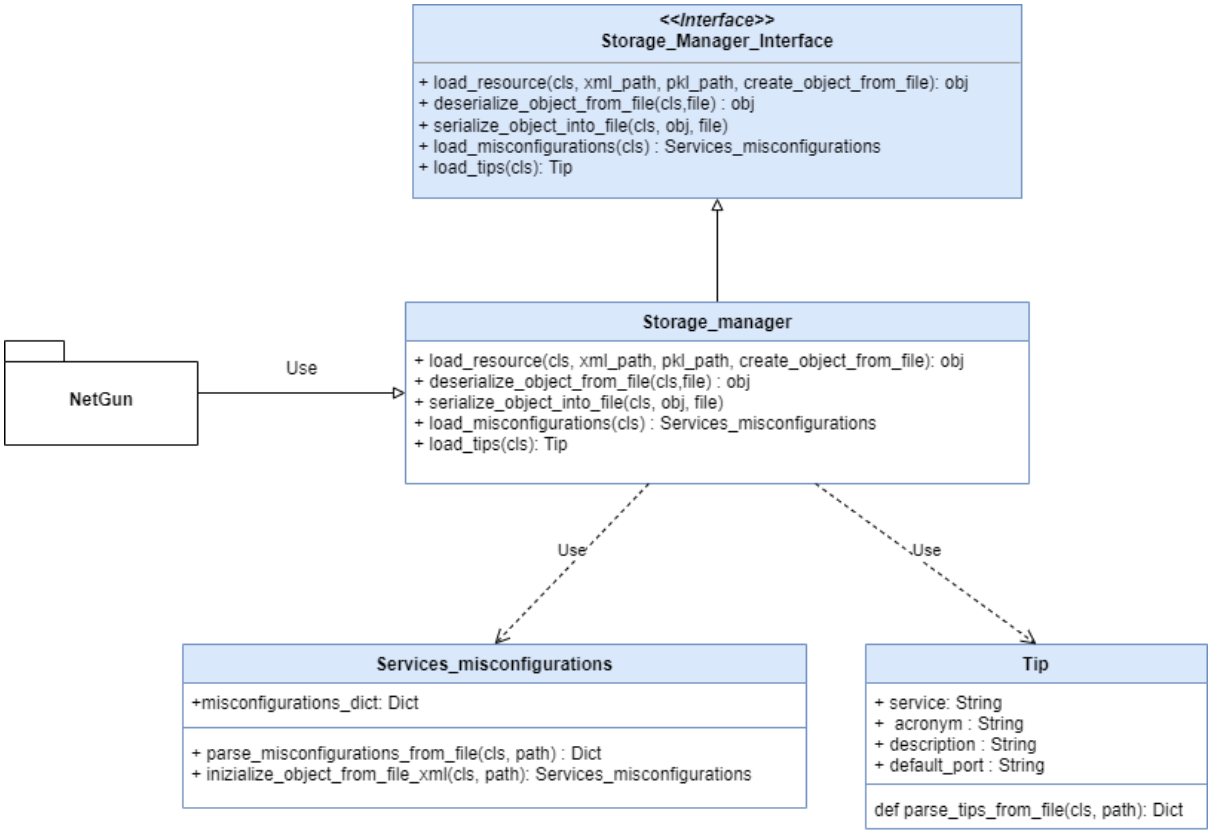
Singleton

Singleton è un design pattern creazionale, ossia un design pattern che si occupa dell'istanziazione degli oggetti, ha lo scopo di garantire, che di una determinata classe venga strutturata una sola istanza, e di fornire un punto di accesso globale a tale istanza.
In NetGun si è ritenuto necessario utilizzare il design pattern Singleton per la creazione del application_context.Context, questa è appunto una classe Singleton, ha il ruolo di gestire e archiviare tutti i dati fondamentali per l'esecuzione di NetGun. Utilizzando questo Context, si ha una migliore gestione dei dati globali, i quali sono immagazzinati nello stesso namespace, cioè il singleton Context.



Facade

Il Facade è un design pattern che permette, implementando una interfaccia semplificata, di accedere a sottosistemi più complessi. In questo modo si può nascondere al sistema la complessità delle librerie, dei framework o dei set di classi che si stanno usando. Si garantisce così un alto disaccoppiamento e si rende la piattaforma più mantenibile e più aggiornabile, poiché basterà cambiare l'implementazione dei metodi dell'interfaccia per implementare le modifiche.
In NetGun si è ritenuto necessario utilizzare il Facade design pattern per la classe Storage Manager. Questa classe fornisce un'interfaccia per la gestione dei dati persistenti, nascondendo l'implementazione a basso livello.



6 Glossario

Sigla/Termine	Definizione
Package	Raggruppamento di classi ed interfacce
DAO	Data Access Object, implementazione dell'omonimo pattern architetturale che si occupa di fornire un accesso in modo astratto ai dati persistenti
Service	Classe che implementa la logica di business, viene utilizzata dall'Handler GUI o da un altro sottosistema
Singleton	Un oggetto con lo scopo di garantire, che di una determinata classe venga strutturata una sola istanza, e di fornire un punto di accesso globale a tale istanza.
Facade	Un oggetto che permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro