



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Corso di Laurea Magistrale in Ingegneria Informatica

Progetto per il corso di Progettazione, Algoritmi e
Computabilità

CleanUp

Docente:

Chiar.ma Prof. Patrizia Scandurra

Componenti del gruppo:

Filippo Bordogna

Xhorxho Papallazi

Simone Ronzoni

Anno Accademico 2021-2022

Immagini

1	Diagramma degli use case	4
2	Topologia del sistema	12
3	Onion architecture	14
4	Model-view-viewmodel pattern	14
1.1	Diagramma dei componenti 1	17
1.2	Diagramma dei componenti 2	17
1.3	Diagramma dei componenti 3	18
1.4	Diagramma delle classi (interfacce)	18
1.5	Diagramma delle classi (tipi di dati)	19
1.6	Diagramma di deployment	20
1.7	Diagramma dei package	22
2.1	Diagramma dei componenti	27
2.2	Diagramma entità relazioni	28
2.3	Diagramma delle classi (tipi di dati)	29
2.4	Diagramma delle classi (interfacce)	29
2.5	Diagramma dei package	30
2.6	Unit test Login	31
2.7	Test API Login	32
2.8	Test API RefreshToken	32
2.9	Metriche di qualità del codice	33
2.10	Login API	34
2.11	Risposta Login API	35
2.12	Refresh Token API	35
2.13	Risposta Refresh Token API	35

3.1	Diagramma dei componenti	40
3.2	Diagramma entità relazioni	41
3.3	Diagramma delle classi (tipi di dati)	42
3.4	Diagramma delle classi (interfacce)	42
3.5	Diagramma dei package	43
3.6	Test API Get User	44
3.7	Test API Get Users	44
3.8	Test API Create User	44
3.9	Test API Update Users	45
3.10	Test API Delete Users	45
3.11	Metriche di qualità del codice	46
3.12	Get User API	47
3.13	Risposta Get User API	47
3.14	Get Users API	48
3.15	Risposta Get Users API	48
3.16	Create User API	48
3.17	Risposta Create User API	49
3.18	Update Users API	49
3.19	Risposta Update User API	49
3.20	Delete Users API	50
3.21	Risposta Delete User API	50
4.1	Diagramma dei componenti	53
4.2	Diagramma entità relazioni	54
4.3	Diagramma delle classi (tipi di dati)	55
4.4	Diagramma delle classi (interfacce)	55
4.5	Diagramma dei package	56
4.6	Unit test Upload Events	57
4.7	Test API Get Events	57
4.8	Test API Get Event	57
4.9	Test API Create Event	58
4.10	Test API Update Event	58
4.11	Test API Delete Event	58

4.12 Test API Upload Events	58
4.13 Test API Get Classrooms	58
4.14 Metriche di qualità del codice	59
4.15 Get Events API	60
4.16 Risposta Get Events API	61
4.17 Get Event API	61
4.18 Risposta Get Event API	62
4.19 Create Event API	62
4.20 Risposta Create Event API	62
4.21 Update Event API	63
4.22 Risposta Update Event API	63
4.23 Delete Event API	63
4.24 Risposta Delete Event API	64
4.25 Upload Events API	64
4.26 Risposta Upload Events API	64
4.27 Get Classrooms API	65
4.28 Risposta Get Classrooms API	65
5.1 Diagramma dei componenti	75
5.2 Diagramma entità relazioni	76
5.3 Diagramma delle classi (tipi di dati)	77
5.4 Diagramma delle classi (interfacce)	78
5.5 Diagramma dei package	79
5.6 Unit test	80
5.7 Test API Get Schedule	80
5.8 Test API Compute Schedule	80
5.9 Test API Get Work Days	80
5.10 Test API Create Work Day	81
5.11 Test API Update Work Day	81
5.12 Metriche di qualità del codice	82
5.13 Get Schedule API	83
5.14 Risposta Get Schedule API	84
5.15 Compute Schedule API	84

5.16 Risposta Compute Schedule API	84
5.17 Get Work Days API	85
5.18 Risposta Get Work Days API	85
5.19 Create Work Day API	85
5.20 Risposta Create Work Day API	86
5.21 Update Work Day API	86
5.22 Risposta Update Work Day API	86
6.1 Diagramma dei componenti	88
6.2 Diagramma entità relazioni	89
6.3 Diagramma delle classi (tipi di dati)	90
6.4 Diagramma delle classi (interfacce)	91
6.5 Diagramma dei package	92
6.6 Test API Get Report	93
6.7 Metriche di qualità del codice	93
6.8 Get Report API	94
6.9 Risposta Get Report API (file .xlsx)	94
7.1 Schermata login	95
7.2 Schermata utente	96
7.3 Schermata eventi	96
7.4 Schermata interventi	97

Indice

0 Iterazione 0	2
0.1 Contesto e obiettivi	2
0.2 Analisi dei requisiti	3
0.2.1 Sistema e casi d'uso	3
0.2.2 Requisiti dell'algoritmo	10
0.2.3 Requisiti non funzionali	11
0.2.4 Architettura del sistema	12
0.2.5 Design patterns	12
0.2.6 Toolchain	14
1 Iterazione 1	16
1.1 Descrizione	16
1.2 Diagramma dei componenti	16
1.2.1 Raggruppamento use cases	16
1.3 Diagramma delle classi (interfacce)	18
1.4 Diagramma delle classi (tipi di dati)	19
1.5 Diagramma di deployment	19
1.6 Diagramma dei package	20
2 Iterazione 2	23
2.1 Descrizione	23
2.2 Casi d'uso	23
2.2.1 UC1 - Login	23
2.2.2 UC1.2 - Refresh del Token	24
2.2.3 UC2 - Logout	25

2.3	Diagramma dei componenti	26
2.4	Diagramma entità relazioni (ER)	28
2.5	Diagramma delle classi	29
2.6	Diagramma delle interfacce	29
2.7	Diagramma dei package	29
2.8	Testing	31
2.8.1	Unit test	31
2.8.2	Test delle API	32
2.9	Analisi statica del codice	33
2.10	Documentazione API	34
3	Iterazione 3	36
3.1	Descrizione	36
3.2	Casi d'uso	36
3.2.1	UC4.1.1 - Visualizzazione account autenticato	36
3.2.2	UC4.1.2 - Visualizzazione di tutti gli account	36
3.2.3	UC4.2 - Aggiunta account operatore	37
3.2.4	UC4.3 - Aggiunta account amministratore	37
3.2.5	UC4.4 - Modifica account operatore	38
3.2.6	UC4.5 - Modifica account amministratore	38
3.2.7	UC4.6 - Eliminazione account	39
3.3	Diagramma dei componenti	40
3.4	Diagramma entità relazioni (ER)	41
3.5	Diagramma delle classi	42
3.6	Diagramma delle interfacce	42
3.7	Diagramma dei package	43
3.8	Testing	44
3.8.1	Test delle API	44
3.9	Analisi statica del codice	46
3.10	Documentazione API	47
4	Iterazione 4	51
4.1	Descrizione	51

4.2	Casi d'uso	51
4.2.1	UC6.1 - Importazione eventi	51
4.2.2	UC6.2 - Modifica eventi	52
4.3	Diagramma dei componenti	53
4.4	Diagramma entità relazioni (ER)	54
4.5	Diagramma delle classi	55
4.6	Diagramma delle interfacce	55
4.7	Diagramma dei package	56
4.8	Testing	57
4.8.1	Unit test	57
4.8.2	Test delle API	57
4.9	Analisi statica del codice	59
4.10	Documentazione API	60
5	Iterazione 5	66
5.1	Descrizione	66
5.2	Casi d'uso	66
5.2.1	UC5 - Assegnazione giornaliera operatori	66
5.2.2	UC7 - Schedulazione interventi pulizia	67
5.2.3	UC8 - Visualizzazione elenco attività pulizia	67
5.3	Scheduler	68
5.3.1	Algoritmo	70
5.3.2	Analisi della complessità	73
5.4	Diagramma dei componenti	75
5.5	Diagramma entità relazioni (ER)	76
5.6	Diagramma delle classi	77
5.7	Diagramma delle interfacce	78
5.8	Diagramma dei package	79
5.9	Testing	80
5.9.1	Unit test	80
5.9.2	Test delle API	80
5.10	Analisi statica del codice	82
5.11	Documentazione API	83

6 Iterazione 6	87
6.1 Descrizione	87
6.2 Casi d'uso	87
6.2.1 UC3 - Visualizzazione report	87
6.3 Diagramma dei componenti	88
6.4 Diagramma entità relazioni (ER)	89
6.5 Diagramma delle classi	90
6.6 Diagramma delle interfacce	91
6.7 Diagramma dei package	92
6.8 Testing	93
6.8.1 Test delle API	93
6.9 Analisi statica del codice	93
6.10 Documentazione API	94
7 Manuale utente	95
8 Conclusioni	98
8.1 Sviluppi futuri	98

Iterazione 0

0.1 Contesto e obiettivi

Con l'avvento del Corona Virus, si sono resi necessari interventi di pulizia e sanificazione degli ambienti con maggior frequenza e miglior efficacia. L'ambiente universitario ha richiesto che al termine di ogni lezione venisse effettuata la sanificazione dell'aula. Questo ha comportato una serie di ritardi dovuti al fatto che l'organizzazione degli interventi di pulizia è a discrezione dei singoli operatori, che giornalmente guardano l'orario e programmano gli interventi senza alcun criterio. Se durante la giornata si verificano ritardi o cambiamenti, ad esempio a causa del prolungamento delle lezioni, allora gli operatori stessi variano il programma per riuscire a coprire tutte le pulizie necessarie. Di conseguenza molti professori sono costretti a ritardare l'inizio delle lezioni e le sanificazioni richieste potrebbero non essere rispettate del tutto. Il problema che andremo a risolvere riguarda quindi l'ottimizzazione degli interventi di pulizia nel contesto universitario, partendo da un insieme di lezioni giornaliere, contraddistinte da un'aula, orario di inizio, orario di fine e numero di studenti partecipanti, e da un certo numero di addetti, con l'obiettivo di assegnargli gli interventi di pulizia in modo da evitare ritardi tra una lezione ed un'altra. L'algoritmo, inoltre, sarà applicabile in un qualsiasi altro scenario, anche al di fuori di quello universitario. Per questo motivo di seguito si utilizzerà il termine evento per generalizzare una lezione.

Abbiamo individuato diverse tipologie di pulizie che si differenziano per velocità di esecuzione, frequenza e profondità di intervento:

- Tra un evento e il successivo si vede necessaria la sanificazione dell'ambiente coinvolto nel modo più rapido possibile così da evitare ritardi. Nei casi di due lezioni consecutive senza tempo libero tra le due, questi interventi di pulizia

avranno priorità massima.

- Gli interventi di pulizia non hanno tutti la stessa durata: questa varia in base alla dimensione dell'aula, dalla durata dell'evento e dall'urgenza della pulizia in caso di due eventi contigui.

Ogni addetto del personale pulizia sarà dotato di un dispositivo mobile che rende possibile lo sviluppo di una soluzione software per la ricezione di notifiche e visualizzazione delle pulizie, notificando variazioni della stessa in caso di ri-programmazioni.

0.2 Analisi dei requisiti

All'interno del sistema sono stati individuati due tipi di attori:

1. l'amministratore per la gestione dell'intero sistema: si occuperà della parte relative la gestione degli utenti, gestione degli eventi e visualizzazione completa delle operazioni di pulizia.
2. l'addetto alle pulizie: avrà solo la possibilità di visualizzare i propri interventi di pulizia pianificati per la giornata.

0.2.1 Sistema e casi d'uso

Come operatore/amministratore si devono poter effettuare varie operazioni:

- UC1 - Effettuare la login per visualizzare gli interventi di pulizia/gestire il sistema di pulizie
- UC2 - Effettuare la logout per disconnettersi dal sistema
- UC3 - Visualizzare un report per verificare lo stato degli interventi giornalieri
- UC4 - Gestire gli account per
 - UC4.1 - Visualizzare gli account degli operatori e amministratori
 - UC4.2 - Aggiungere gli account degli operatori
 - UC4.3 - Aggiungere gli account amministratori
 - UC4.4 - Modifica account operatore
 - UC4.5 - Modifica account amministratore
 - UC4.6 - Eliminazione account operatore e amministratore
- UC5 - Assegnare giornalmente gli operatori che effettueranno le pulizie

- UC6 - Gestire gli eventi per
 - UC6.1 - Importare gli eventi programmati per permettere lo scheduling degli interventi di pulizia
 - UC6.2 - Modificare gli eventi programmati
 - UC7 - Richiamare lo scheduler degli eventi per aggiornare la schedule
- Come operatore voglio poter:
- UC8 - Visualizzare l'elenco delle attività da svolgere per effettuare le pulizie

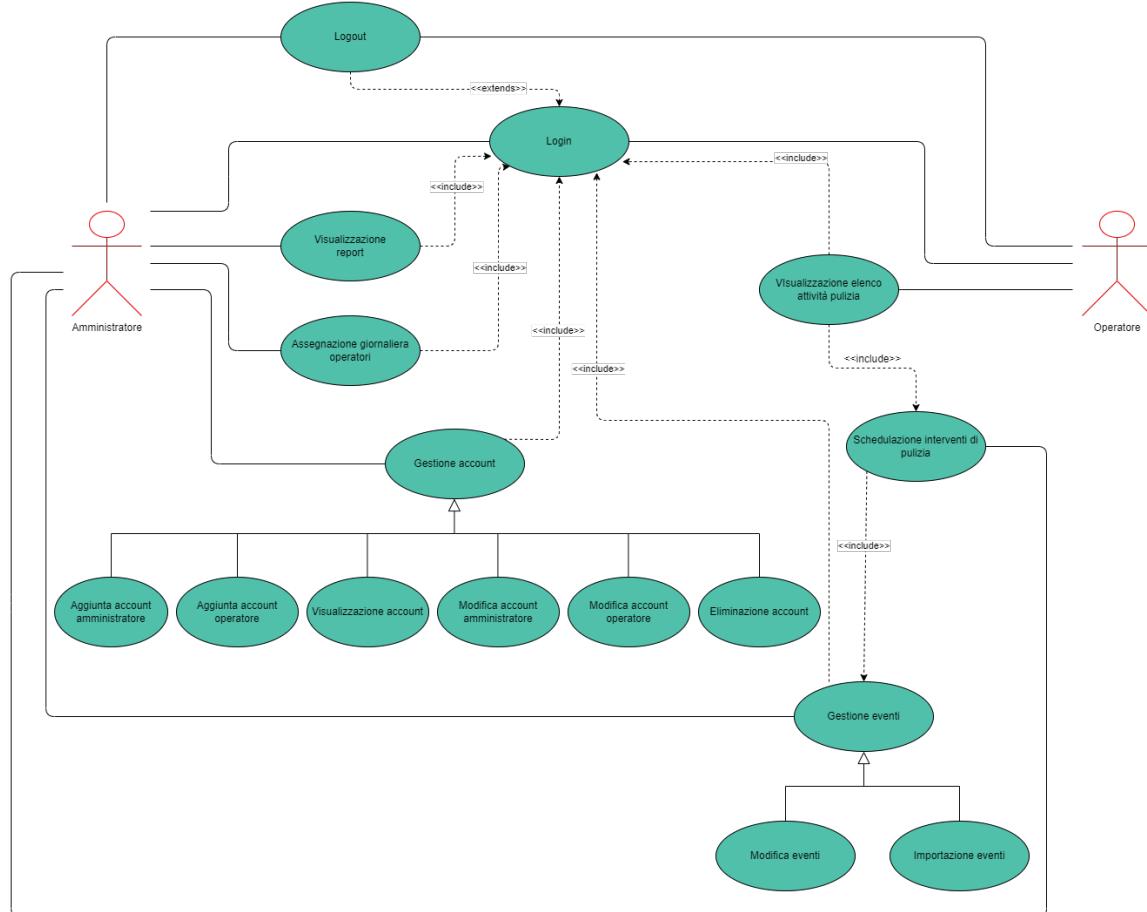


Figura 1: Diagramma degli use case

Di seguito si riporta il dettaglio di ciascun caso d'uso.

UC1	Login
<i>Descrizione:</i>	Autenticazione di un utente al sistema
<i>Attori:</i>	Amministratore, operatore
<i>Precondizioni:</i>	L'utente è in possesso delle credenziali di un account creato da un amministratore

Flusso eventi:

1. L'utente inserisce le proprie credenziali
2. Il sistema controlla i dati inseriti e autentica l'utente come amministratore o come operatore

Postcondizioni: Reindirizzamento alla pagina di amministrazione o a una pagina operatore

Eccezioni: L'utente inserisce credenziali errate

UC2 Logout

Descrizione: Disconnessione dal proprio account

Attori: Amministratore, operatore

Precondizioni: L'utente è già autenticato

Flusso eventi:

1. L'utente richiede la disconnessione

Postcondizioni: Reindirizzamento alla pagina di accesso

Eccezioni: L'utente inserisce credenziali errate

UC3 Visualizzazione report

Descrizione: Visualizzazione dei report giornalieri delle attività

Attori: Amministratore, operatore

Precondizioni: L'utente è già autenticato

Flusso eventi:

1. L'amministratore richiede il report di un giorno

2. Generare il report relativo

Postcondizioni: Report generato

Eccezioni: In quel giorno non sono presenti eventi o attività

UC4.1 Visualizzazione account

Descrizione: Visualizzazione degli account di amministrazione e operatori

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

UC4.2 Aggiunta account operatore

Descrizione: Aggiunta di un account relativo a un operatore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore inserisce i dati dell'operatore
2. L'amministratore inserisce una email e password relativa a quell'account
3. Il sistema salva tale account

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

UC4.3 Aggiunta account amministratore

Descrizione: Aggiunta di un account relativo a un amministratore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore inserisce i dati del nuovo utente
2. L'amministratore inserisce una email e password relativa a quell'account
3. Il sistema salva tale account

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

UC4.4 Modifica account operatore

Descrizione: Modifica di un account relativo ad un operatore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore inserisce i dati del nuovo utente

2. L'amministratore inserisce una email e password relativa a quell'account

3. Il sistema salva tale account

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

UC4.5 Modifica account amministratore

Descrizione: Modifica di un account relativo ad un amministratore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore inserisce i dati del nuovo utente

2. L'amministratore inserisce una email e password relativa a quell'account

3. Il sistema salva tale account

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

UC4.6 Eliminazione account

Descrizione: Eliminazione di un account relativo ad un operatore o ad un amministratore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore seleziona l'account da eliminare e lo rimuove

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

UC5 Assegnazione giornaliera operatori

Descrizione: Caricamento dell'orario di lavoro degli operatori

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'utente specifica l'orario nel quale gli operatori sono disponibili in un dato giorno
2. Il sistema effettua una verifica dei dati inseriti dall'utente
 - a. Se i dati sono corretti, il sistema salva lo stato degli operatori
 - b. Se i dati contengono errori, essi vengono mostrati all'utente

Eccezioni: L'utente inserisce un piano di disponibilità errato

UC6.1 Importazione eventi

Descrizione: Importazione degli eventi che si svolgono in una giornata

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore carica l'elenco degli eventi
2. Il sistema computa gli elenchi di interventi di pulizia
3. Il sistema avvisa mediante notifica push gli operatori che sono stati assegnati ad una lista di pulizia

Postcondizioni: Visualizzazione dell'elenco di interventi di pulizia

Eccezioni: L'amministratore aggiunge eventi con data passata

UC6.2 Modifica eventi

Descrizione: Modifica evento

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore.
Ci sono già degli eventi per quel giorno.

Flusso eventi:

1. L'amministratore carica l'elenco degli eventi aggiornati
2. Il sistema elabora le modifiche agli elenchi di interventi di pulizia
3. Il sistema avvisa mediante notifica push solo gli operatori interessati dalle modifiche

Postcondizioni: Visualizzazione elenco interventi pulizia aggiornati

Eccezioni: L'amministratore va a modificare eventi in corso: in questo caso non si può fare nulla

UC7 Schedulazione interventi pulizia

Descrizione: Richiamare lo scheduler per forzare l'aggiornamento della schedule

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. Viene richiamato il componente che implementa l'algoritmo di scheduling

Eccezioni: L'algoritmo non ha eventi da rischedulare

UC8 Visualizzazione elenco attività pulizia

Descrizione: Visualizzazione dell'elenco degli interventi di pulizia di un certo operatore

Attori: Amministratore, operatore

Precondizioni: L'utente è autenticato come operatore o amministratore

Flusso eventi:

1. Viene caricato e mostrato l'elenco degli interventi di pulizia di un operatore specificato
 - a. Se l'account è di un amministratore può richiedere qualsiasi utente
 - b. Se l'account è di un operatore può accedere solo al suo

Eccezioni: L'utente che richiede non ha interventi di pulizia da effettuare.

Un operatore cerca di richiedere l'elenco attività non sue.

0.2.2 Requisiti dell'algoritmo

Ad inizio giornata l'algoritmo può stabilire il numero ottimale di operatori in base al numero di interventi necessari in quel giorno. In input riceve i dati di quella giornata:

- il numero di operatori effettivamente disponibili
- l'elenco degli interventi di pulizia, ciascuno contraddistinto da orario di inizio coincidente con l'orario di fine della lezione, orario di fine coincidente con l'orario

di inizio della lezione seguente, nome dell'aula e altri possibili dati utili per calcolare la durata dell'intervento

In output restituisce l'elenco degli interventi per ogni operatore.

Se in un ambiente si è verificato almeno un evento allora va effettuata una pulizia di fine giornata. Alla fine di ogni evento se tale ambiente viene ancora utilizzato all'interno della stessa giornata, è necessaria una sanificazione potendo seguire alcune euristiche: se l'evento ha durata maggiore di una certa soglia si dovrà intervenire con una sanificazione approfondita, altrimenti con una sanificazione breve. Gli ambienti che presentano due eventi consecutivi hanno la precedenza rispetto agli altri scenari.

0.2.3 Requisiti non funzionali

In caso di variazioni degli eventi pianificati nel corso della stessa giornata, il sistema dovrà avvisare gli operatori almeno un'ora prima. Questo comporterà una modifica delle attività di pulizia relativa a determinati operatori, che dovrà essere inviata mediante notifica push sul dispositivo mobile in dotazione.

Per quanto riguarda la portabilità, l'applicazione in dotazione agli operatori dovrà poter funzionare sia su dispositivi Android che iOS. Inoltre, il back-end fornirà della api REST indipendenti dal linguaggio e dalla tecnologia di implementazione lato operatore.

Dal punto di vista della sicurezza, gli operatori dovranno essere autenticati e ognuno potrà visualizzare solo la propria pianificazione di attività giornaliere. L'amministratore che avrà accesso alla pagina web di gestione avrà anche esso un account dedicato, con permessi differenti rispetto agli account operatore.

Al fine di garantire la qualità di manutenibilità si dovranno parametrizzare il più possibile informazioni come l'euristica utilizzata per calcolare la durata degli interventi di pulizia. Inoltre la progettazione e realizzazione di un sistema per componenti favorisce eventuali modifiche future.

0.2.4 Architettura del sistema

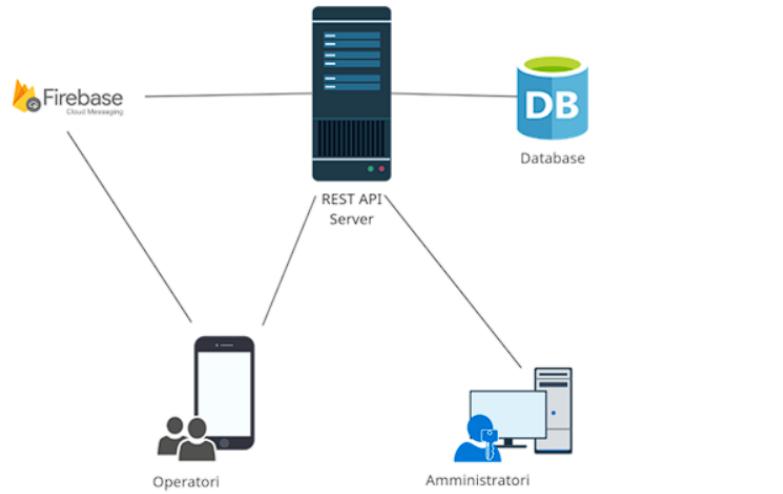


Figura 2: Topologia del sistema

Il sistema sarà costituito da 3 componenti principali:

1. un'applicazione web che espone tramite api REST le risorse necessarie, memorizzate in un database sulla stessa macchina
2. una single page application che fungerà da backoffice per l'amministrazione
3. un'applicazione mobile per gli operatori su cui visualizzare gli interventi di pulizia assegnati

Per notificare agli operatori eventuali variazioni riguardo le proprie operazioni di pulizia della giornata, l'applicativo a backend si interfaccia a Firebase.

0.2.5 Design patterns

Il sistema si baserà su una architettura client-server. La soluzione a back-end comprende due progetti, uno che implementa ed espone le api REST in rete e uno per l'applicazione web client per l'amministrazione. Il progetto delle web api è strutturato secondo il pattern Onion Architecture, che sviluppa un'architettura a strati:

- Domain che comprende i modelli dei dati
- Application che contiene la logica di business

- Infrastructure per i componenti che si occupano dell'accesso ai dati e di configurazione

Per l'accesso ai dati si utilizza pattern Repository.

Nella realizzazione di api in .NET è molto utilizzato l'approccio CQRS, acronimo di Command Queries Separation of Concerns, che suddivide quelle che sono le operazioni di lettura e scrittura nella base dati rendendole indipendenti tra loro con l'obiettivo di ridurre le dipendenze interne. Operativamente questo si traduce che ad ogni api è associato un command (nel caso in cui questa modifichi dei dati) o una query (nel caso in cui debba solamente recuperare dei dati) dedicati.

Essendo i command e le queries indipendenti tra di loro, per generalizzare la logica comune si utilizzano dei servizi. Per implementare il modello CQRS, si utilizza il pattern Mediator che permette di far interagire componenti diversi del sistema che condividono una interfaccia comune introducendo un unico punto di controllo centrale (chiamato mediatore) per garantire basso accoppiamento.

Nel progetto Client Web l'interazione tra interfaccia e dati avviene mediante pattern Model View ViewModel, con repository per l'accesso ai dati mediante chiamate HTTPS. Il funzionamento sarà il seguente: a seguito dell'interazione dell'utente con la view, verrà richiamato il metodo relativo del proprio ViewModel che si occuperà di eseguire la logica ed eventualmente recuperare i dati mediante repository da api. Una volta ricevuti i dati, il.viewmodel li elaborerà e li renderà disponibili alla view.

L'applicazione mobile implemterà il pattern MVVM e anch'essa sarà implem-tata rispettando la separazione dei layer.

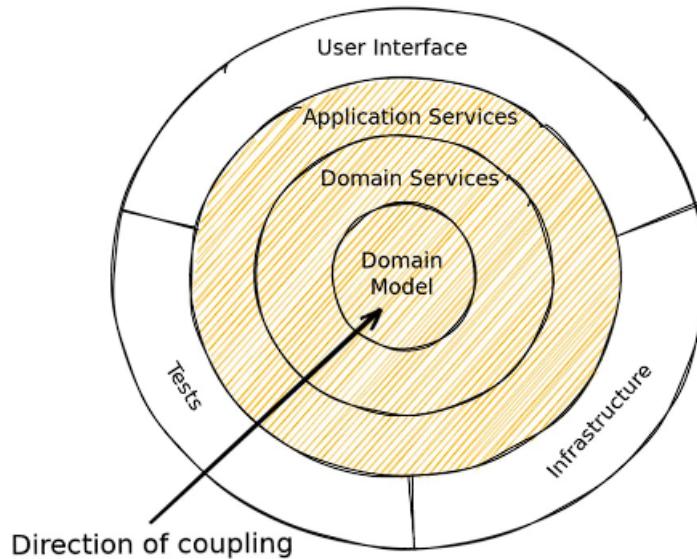


Figura 3: Onion architecture

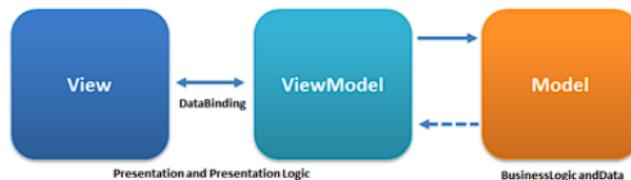


Figura 4: Model-view-viewmodel pattern

0.2.6 Toolchain

Strumenti utilizzati per la realizzazione del progetto:

- Modellazione diagrammi: draw.io
- Implementazione software back-end:
 - Tecnologia: ASP.NET Core 6, linguaggio C#
 - Framework e librerie: Entity Framework (ORM), Serilog (logging), Swagger
 - IDE: Visual Studio 2022
 - DMBS: SQL Server
 - Gestione DB: SQL Server Management Studio (SSMS)

- Implementazione client web:
 - Tecnologia: Blazor Web Assembly, linguaggio C#
 - Framework e librerie: MudBlazor (ui), Blazored (local storage)
 - IDE: Visual Studio 2022
- Implementazione software app:
 - Tecnologia: Flutter 2.8.1, linguaggio Dart
 - Framework e librerie: Provider, http, Firebase messaging
 - IDE: Visual Studio Code
 - Emulatore Android
- Analisi del software:
 - Analisi statica: Visual Studio code analysis
- Testing
 - Unit test: xUnit
 - Postman
- Documentazione, versioning:
 - Versioning: Git con GitHub
 - Documentazione: Latex con Overleaf
 - Pianificazione: Azure Boards con Azure DevOps

Iterazione 1

1.1 Descrizione

In questa iterazione si rappresenta il progetto nel suo complesso, lasciando la progettazione in dettaglio dei singoli componenti per le iterazioni successive.

1.2 Diagramma dei componenti

1.2.1 Raggruppamento use cases

Per comprendere i componenti che costituiranno il sistema, sono stati raggruppati gli use case in gruppi di affinità in base alle funzionalità:

- Gestione degli account operatori (UC4.1, UC4.2, UC4.3, UC4.4, UC4.5, UC4.6)
- Gestione eventi (UC6.1, UC6.2)
- Scheduler (UC5, UC7, UC8)
- Autenticazione (UC1, UC2)
- Report (UC3)

Lo sviluppo del diagramma è stato fatto per passi incrementali che vedevano in un primo momento il sistema come un grande componente che espone alcune interfacce per poi rifinirlo e componentizzare le interfacce stesse riferendo la struttura finale. Nelle iterazioni successive saranno specializzati via via i componenti del sistema stesso.

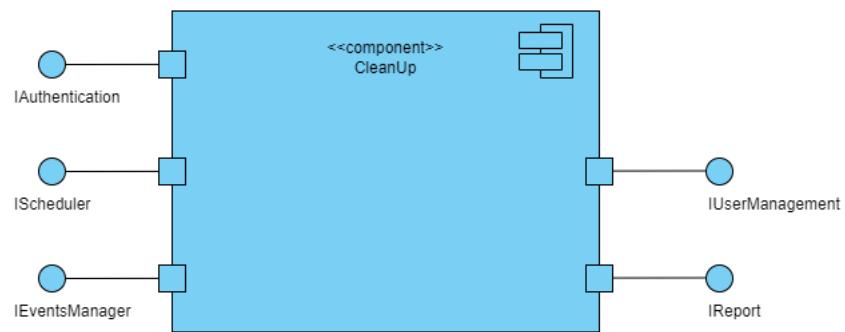


Figura 1.1: Diagramma dei componenti 1

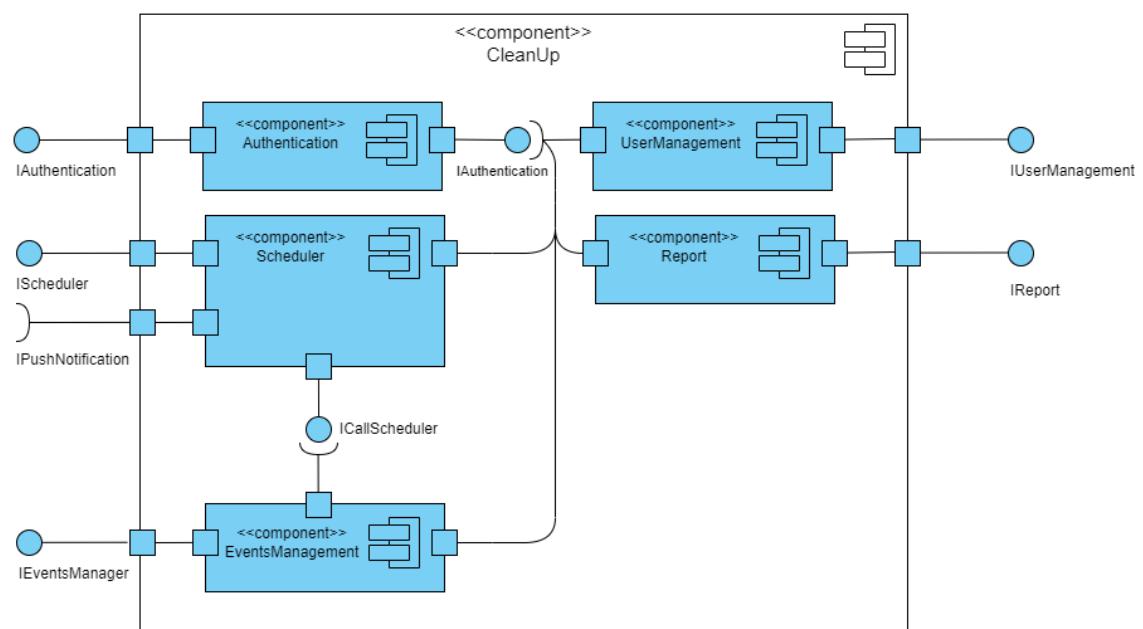


Figura 1.2: Diagramma dei componenti 2

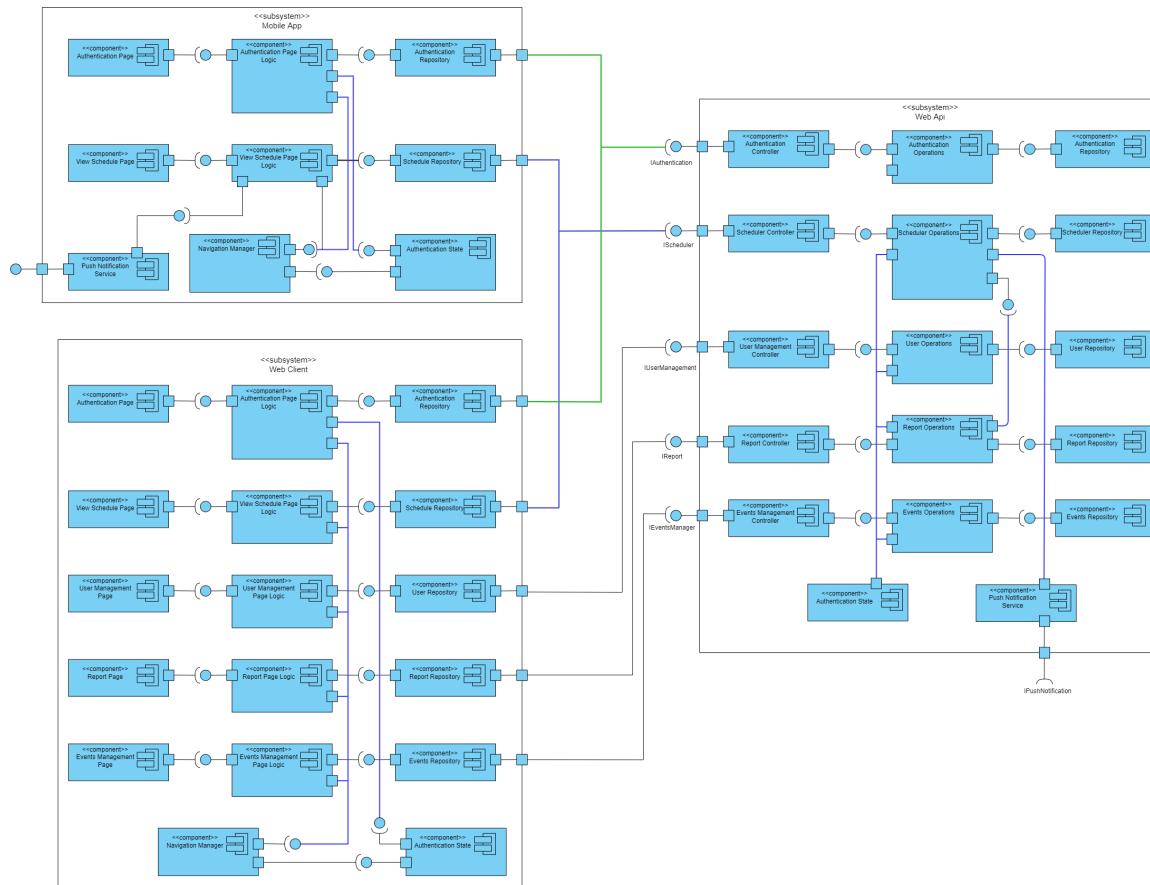


Figura 1.3: Diagramma dei componenti 3

1.3 Diagramma delle classi (interfacce)

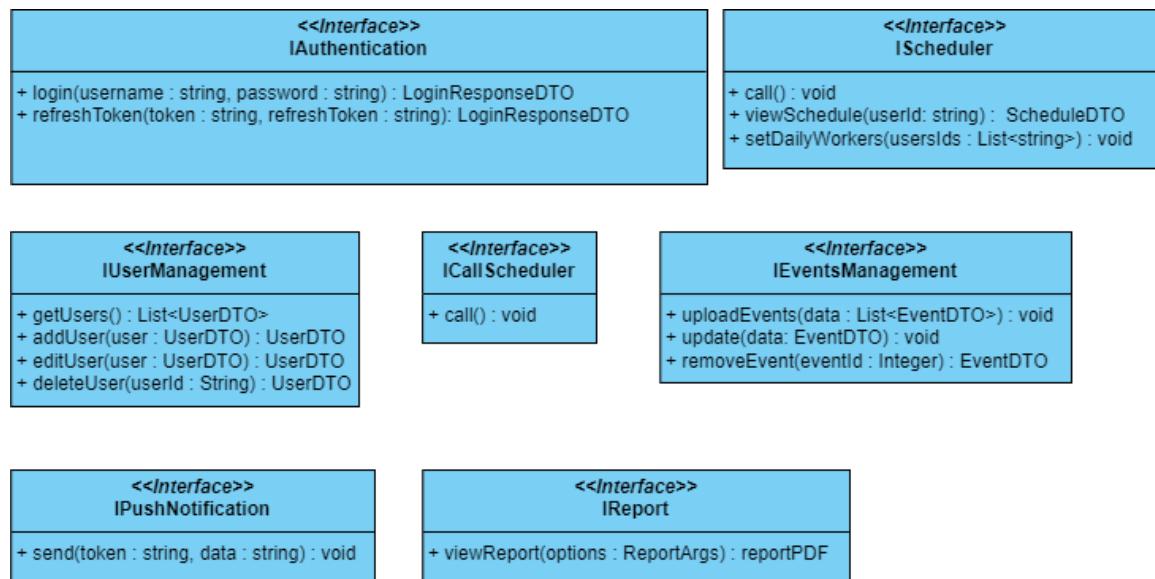


Figura 1.4: Diagramma delle classi (interfacce)

1.4 Diagramma delle classi (tipi di dati)

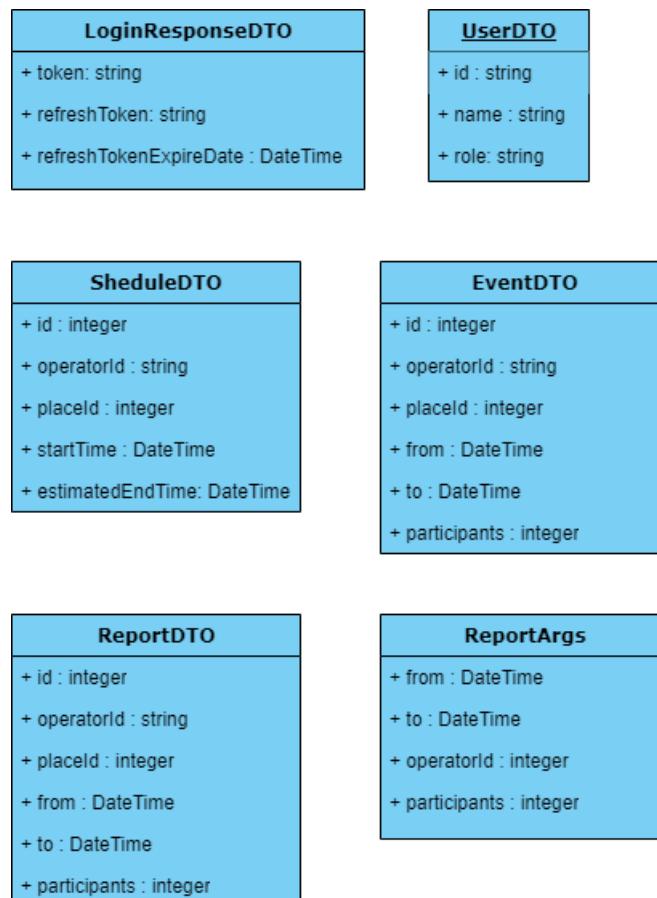


Figura 1.5: Diagramma delle classi (tipi di dati)

1.5 Diagramma di deployment

Compaiono 4 nodi:

- il server che espone le API e contiene il database
- il server Google che espone i servizi di Firebase
- il dispositivo mobile per gli operatori che accede ai servizi offerti dai server cui sopra
- Il browser lato client usato dagli amministratori per accedere al client web.

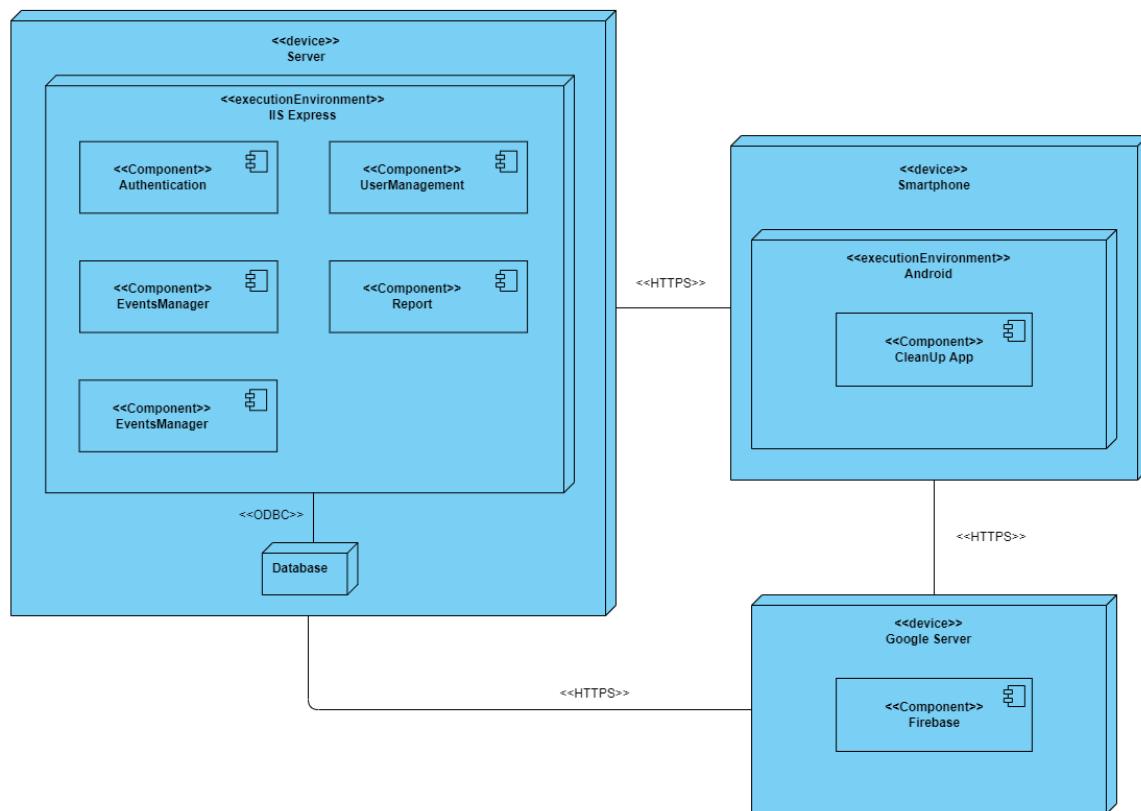


Figura 1.6: Diagramma di deployment

1.6 Diagramma dei package

La suddivisione in package si riferisce alla onion architecture, con una struttura circolare incentrata sui domain models con dipendenze verso l'interno.

I due package principali sono CleanUp e com.cleanup.app che conterranno rispettivamente la parte backend e web degli amministratori e l'applicazione android che fa da presentation layer per gli operatori del sistema. All'interno del package CleanUp si trovano altri package quali:

- CleanUp.Domain: contiene i modelli che rappresentano i dati e la logica di business
- CleanUp.Application: contiene i servizi applicativi e delle logiche relative all'applicazione e non ai dati
- CleanUp.Application.WebApi: mentre il package precedente conteneva i componenti riguardanti l'intera applicazione (come metodi helpers, modelli condivisi, interfacce, ...), qui è contenuta la logica vera e propria delle sole api, tra cui tutti i commands e queries definiti secondo il pattern CQRS.
- CleanUp.Infrastructure: si occupa di accedere ai dati e ne astrae l'accesso con il meccanismo dei repository
- CleanUp.WebApi: espone le API REST
- CleanUp.WebApi.Sdk: contiene tutti i modelli e le richieste che il client utilizzerà per recuperare i dati tramite api.
- CleanUp.Client: contiene le pagine web visitabili dagli amministratori e operatori, e la relativa logica. Per mantenere web client e api separate, non vi è alcuna dipendenza.

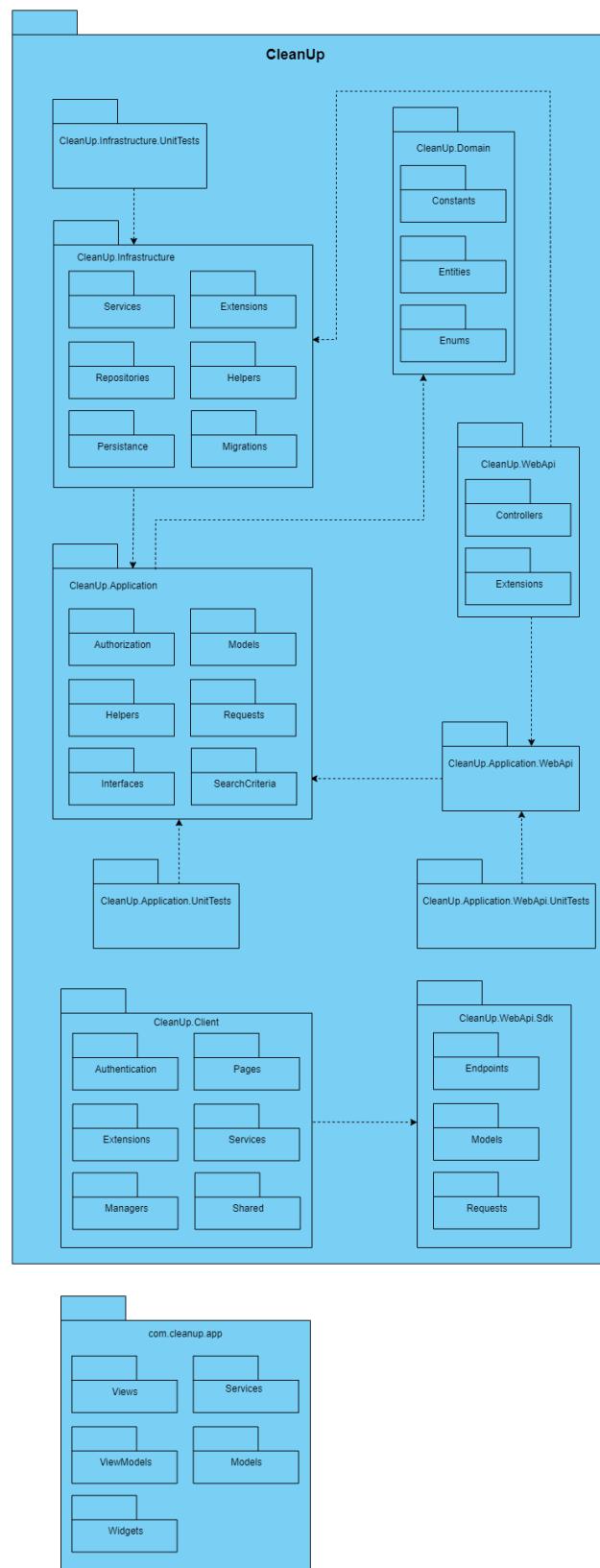


Figura 1.7: Diagramma dei package

Iterazione 2

2.1 Descrizione

In questa iterazione si sono progettati e sviluppati i casi d'uso relativi all'autenticazione (UC1, UC2), che implementano funzionalità di base necessarie per tutto il resto della progettazione. Al termine di questa fase sarà possibile autenticarsi al sistema ed accedere solo alle aree a cui si è autorizzati (per eseguire le funzionalità che saranno sviluppate nelle prossime iterazioni). L'autorizzazione verrà gestita mediante ruoli e claims.

2.2 Casi d'uso

2.2.1 UC1 - Login

Si è deciso di implementare l'autenticazione mediante Bearer token senza l'utilizzo di un Identity Server dedicato. Come per la basic authentication, anche in questo caso per garantire un certo livello di sicurezza si può utilizzare solamente il protocollo HTTPS.

Il token generato e restituito a seguito della login dell'utente dovrà essere inviato nel campo Authentication della richiesta http per autenticare ed autorizzare l'utente ad eseguire le operazioni richieste.

UC1	Login
<i>Descrizione:</i>	Autenticazione di un utente al sistema
<i>Attori:</i>	Amministratore, operatore

Precondizioni: L'utente è in possesso delle credenziali di un account creato da un amministratore

Flusso eventi:

1. L'utente inserisce le proprie credenziali
2. Il sistema controlla i dati inseriti e autentica l'utente come amministratore o come operatore

Postcondizioni: Reindirizzamento alla pagina di amministrazione o a una pagina operatore

Eccezioni: L'utente inserisce credenziali errate

2.2.2 UC1.2 - Refresh del Token

Use case aggiunto rispetto a quanto previsto dall'iterazione 1.

Al token jwt è stato assegnata una durata di validità pari a 10 minuti. Al termine di questo tempo per poter accedere alle risorse bisognerà rigenerare un nuovo token inviando al server un refresh token (anch'esso memorizzato nel client e ottenuto in fase di autenticazione). In questo modo è possibile generare periodicamente un nuovo token senza dover richiedere all'utente di autenticarsi immettendo le credenziali ogni volta.

Dopo 20 minuti di inattività anche il refresh token scade, rendendo necessario rimandare l'utente alla pagina di login per poter continuare ad accedere al servizio.

UC1.2	Refresh del Token
--------------	--------------------------

Descrizione: Refresh del Token di autenticazione

Attori: Amministratore, operatore

Precondizioni: L'utente è già autenticato

Flusso eventi:

1. Il token di autenticazione viene refreshato

Postcondizioni: Viene restituito il nuovo token da utilizzare

Eccezioni: Refresh token non valido

2.2.3 UC2 - Logout

I Token JWT sono un modo per gestire l'autenticazione di un utente in modo stateless. È uno standard che non necessita di memorizzare lo stato di autenticazione in alcun tipo di storage a backend, sia esso di sessione o a database. Un token quindi, una volta generato, può essere utilizzato teoricamente all'infinito ma in realtà ha una scadenza. A meno di implementazioni custom per rendere invalido il token quando l'utente effettua l'operazione di logout, questa consiste in una semplice cancellazione del token all'interno del browser.

UC2	Logout
<i>Descrizione:</i>	Disconnessione dal proprio account
<i>Attori:</i>	Amministratore, operatore
<i>Precondizioni:</i>	L'utente è già autenticato
<i>Flusso eventi:</i>	<ol style="list-style-type: none"> 1. L'utente richiede la disconnessione
<i>Postcondizioni:</i>	Reindirizzamento alla pagina di accesso
<i>Eccezioni:</i>	L'utente salva il token

2.3 Diagramma dei componenti

In questa iterazione sono stati sviluppati i casi d'uso riguardanti l'autenticazione e si riesce quindi a vedere che l'interfaccia per l'autenticazione ora consta di 3 metodi che eseguono le 3 operazioni necessarie per l'autenticazione al sistema. Sia l'applicazione mobile che il client web faranno uso di questa interfaccia che permetterà di ottenere, o anche rinfrescare, il token da utilizzare per accedere alle chiamate api.

Il componente per l'autenticazione è stato sviluppato secondo i pattern descritti nell'introduzione e si possono notare subito i comandi derivanti dal pattern CQRS, *Login Command* e *Refresh Login Command*, e la funzione di handling che permette l'utilizzo dei comandi da parte di un altro nuovo componente che si occupa di fornire il servizio di autenticazione in modo trasparente rispetto alla loro implementazione.

In questo modo si vede subito come una modifica futura interessi solo i comandi che potranno essere modificati senza indurre modifiche in altre parti del sistema.

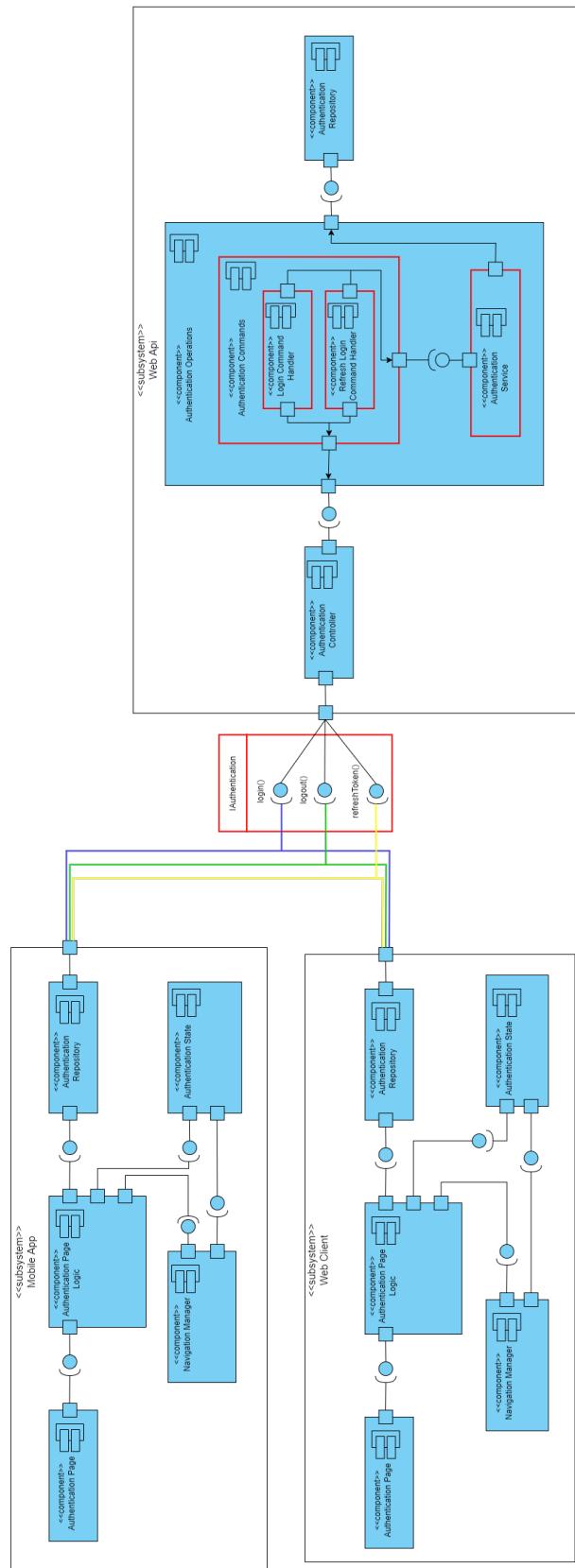


Figura 2.1: Diagramma dei componenti

2.4 Diagramma entità relazioni (ER)

L'implementazione delle funzionalità di autenticazione ha portato alla creazione di nuove tabelle all'interno del database:

- è stato introdotto un nuovo schema *auth* per raggruppante le tabelle riguardanti l'autenticazione
- questo nuovo schema contiene le tabelle principali di Utenti, Ruoli e Claims e poi tutte le tabelle per permettere le associazioni n a n tra di loro

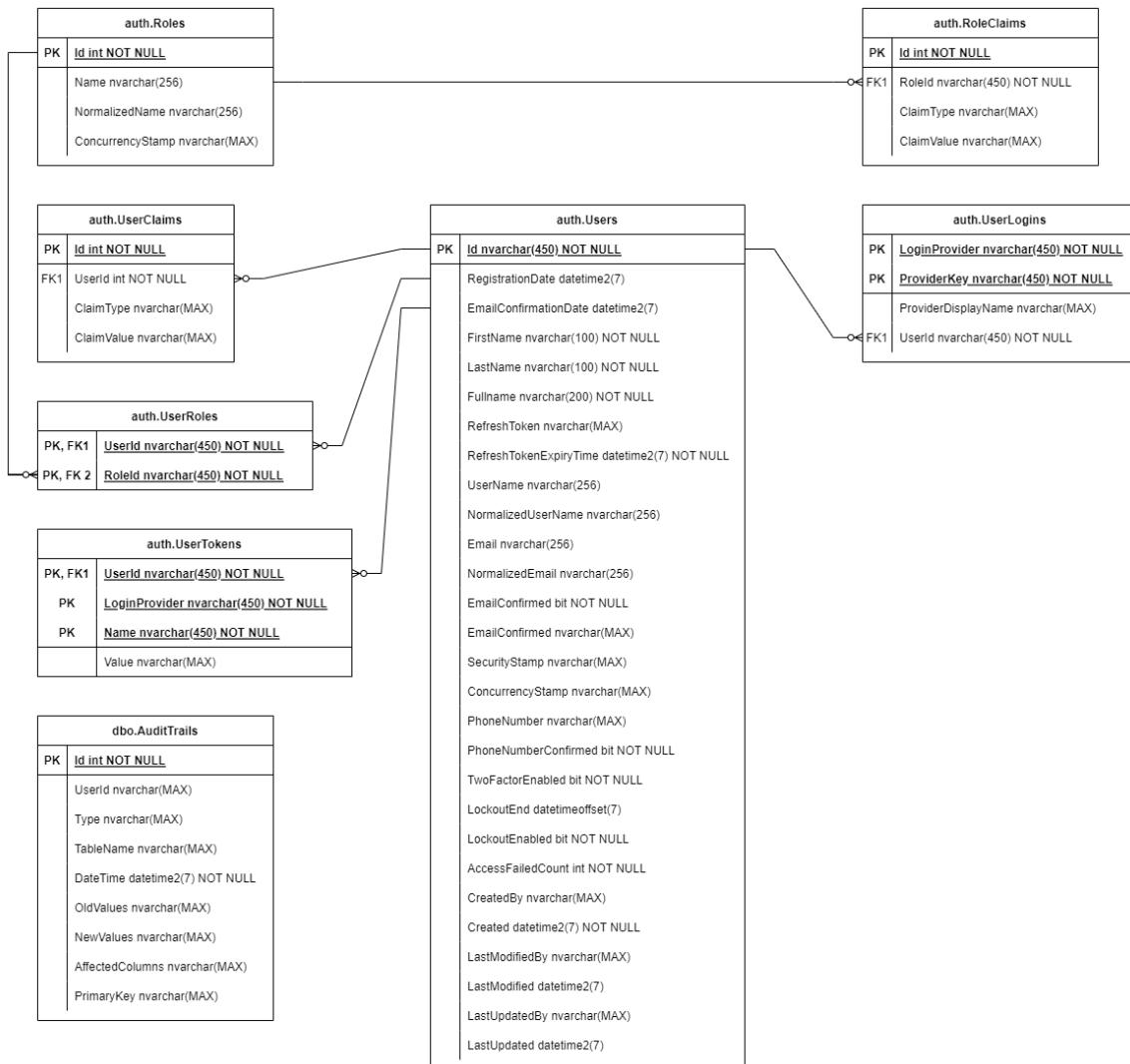


Figura 2.2: Diagramma entità relazioni

2.5 Diagramma delle classi

Per quanto concerne la classe inviata in risposta ad una richiesta di login che ha avuto successo, è stato aggiunto il campo *UserId* per restituire all'utente anche il suo identificativo.



Figura 2.3: Diagramma delle classi (tipi di dati)

2.6 Diagramma delle interfacce

Rispetto all'interfaccia definita nelle iterazioni precedenti, durante questa iterazione si è scelto di inserire una nuovo metodo accessibile per l'operazione di refresh del token JWT. Come descritto in precedenza questa introduzione è stata guidata dal fatto che i token hanno una scadenza oltre la quale l'utente deve autenticarsi nuovamente al sistema. Per rendere l'utilizzo del servizio più fruibile, senza continue richieste di accesso, è stata quindi introdotta questa funzionalità.



Figura 2.4: Diagramma delle classi (interfacce)

2.7 Diagramma dei package

La creazione dei comandi per le operazioni di autenticazione e delle tabelle per la memorizzazione dei dati ha portato all'aggiunta di nuovi package all'interno del progetto:

- è stato aggiunto un package `CleanUp.Application.WebApi.Authentication` che contiene comandi e query relative all'autenticazione

- è stato aggiunto un package `CleanUp.Application.WebApi.UnitTests.Authentication` per contenere i test del package di cui sopra
- per quanto riguarda invece l'Sdk utilizzato dal client, sono stati mappati gli oggetti di richiesta e risposta per le chiamate di autenticazione

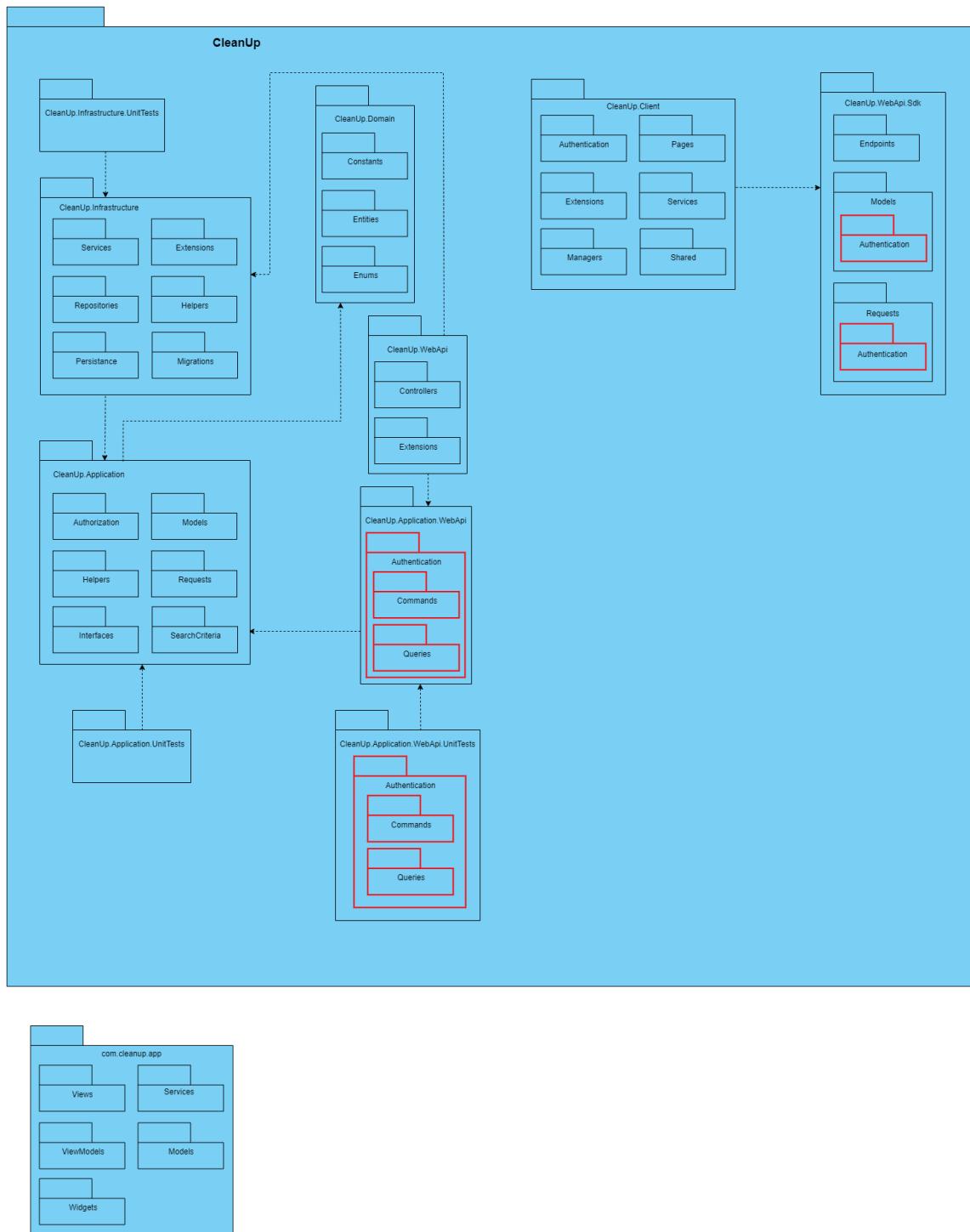


Figura 2.5: Diagramma dei package

2.8 Testing

2.8.1 Unit test

Nelle varie iterazioni è stato eseguito lo unit test solamente di quei metodi critici per la nostra applicazione. Sono comunque stati fatti tutti i test sulle Api per validare il risultato rispetto ai casi d'uso.

In questa iterazione è stata effettuato lo unit testing del metodo di Login dell'Authentication Service per verificare il corretto comportamento nei diversi scenari possibili:

1. Il primo test è il caso in cui vengono fornite credenziali valide, si verifica che venga ritornato l'oggetto corretto con tutti i campi popolati.
2. Secondo test è il caso in cui viene passato un indirizzo email esistente a database ma non verificato. Lancia correttamente eccezione.
3. Terzo test è il caso in cui viene passato un indirizzo email valido con password sbagliata. Lancia correttamente eccezione.

▲ ✓ CleanUp.Infrastructure.UnitTests.Authentication.Commands (3)	246 ms
▲ ✓ AuthenticationServiceTests (3)	246 ms
✓ Login_ShouldReturnToken_WhenCorrectCredentials	69 ms
✓ Login_ShouldThrowException_WhenEmailNotConfirmed	173 ms
✓ Login_ShouldThrowException_WhenIncorrectPassword	4 ms

Figura 2.6: Unit test Login

2.8.2 Test delle API

Ad ogni interazione sono stati effettuati test sulle api in modo da verificare che siano stati implementati e rispettati tutti i casi d'uso. Per ogni api sono stati effettuati più test per verificare:

- Corretto contenuto della risposta in funzione di dati in input come casi limite
- Schema della risposta
- Corretta gestione di errori di validazione dei dati in input
- Corretta gestione di tutti i possibili casi di errore

Ad ogni iterazione, venivano rieseguiti anche tutti i test delle iterazioni precedenti per verificare che le modifiche apportate non abbiano involontariamente intaccato funzionalità precedentemente implementate.

POST Login https://localhost:7162/v1/authentication/login / Authentication / Login	
Pass	Status code is 200
Pass	Response JSON Schema is correct
Pass	Incorrect Syntax Email
Pass	Incorrect sintax Password
Pass	Unknown Email
Pass	Email not verified
Pass	Wrong Password

Figura 2.7: Test API Login

POST RefreshToken https://localhost:7162/v1/authentication/refresh / Authentication / RefreshToken	
Pass	Status code is 200
Pass	Response JSON Schema is correct
Pass	Empty JWT Token
Pass	Empty Refresh Token
Pass	Syntactically wrong JWT Token
Pass	Wrong Refresh Token

Figura 2.8: Test API RefreshToken

2.9 Analisi statica del codice

Di seguito si riportano le metriche di qualità del codice fornite dallo strumento integrato di Visual Studio.

Gerarchia	Indice di manutenibilità	Complessità ciclomatica	Profondità dell'ereditarietà	Accoppiamenti di classi	Righe di codice ...	Righe di codice ...
src\CleanUp.Application (Debug)	95	70	4	17	191	22
src\CleanUp.Application.WebApi (Debug)	87	83	2	48	455	83
src\CleanUp.Domain (Debug)	99	29	3	9	54	1
src\CleanUp.Infrastructure (Debug)	69	82	6	142	2.755	779
src\Hangfire\CleanUp.Hangfire (Debug)	86	102	4	83	668	139
src\Hangfire\CleanUp.Hangfire.Data (Debug)	77	4	1	46	86	19
src\Sdk\CleanUp.WebApi.Sdk (Debug)	95	121	2	19	245	26
src\Web\CleanUp.Client (Debug)	88	708	3	187	9.016	775
src\Web\CleanUp.WebApi (Debug)	84	178	4	217	1.124	297
test\CleanUp.Infrastructure.UnitTests (Debug)	78	6	1	42	174	82

Figura 2.9: Metriche di qualità del codice

2.10 Documentazione API

API Test

API Testing of the CleanUp project (PAC academic year 2022-2023).

AUTHORIZATION Bearer Token

Token

Authentication

Iteration 2:

Authentication Service APIs.

POST Login

<https://localhost:7162/v1/authentication/login>

API that returns the JWT login token after entering the correct credentials.

BODY raw

```
{  
  "email": "*****",  
  "password": "*****"  
}
```

Figura 2.10: Login API

```

"response": {
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJodHRwOi8vC2NoZW1hcy54bWxzB2FwlM9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYwltcy9uYW1laWRlbnRpZmlciI6ImY0ZGE0ZGz1LTc10TMTNDU1Ny04Yjg2L
WN10GMxZTcwNzASMiisImh0dHA6Ly9zY2hlbwFzLnhbHNvYXAub33nL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL2VtYwlsYwRkcmVzcyI6InNyb256b25p0t1AZ2
1haWwuY29tIiwiHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9pZGVudG10eS9jbGfpbXvbmFtZS16IlNpbw9uZSiisImh0dHA6Ly9zY2hlbwFzLnh
tbHNVvXAut3JnL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL3N1cm5hbWUi0iJSb256b25pIiwiHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9p
ZGVudG10eS9jbGfpbXvbmUi0iILCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDgvMDgvY2xhaW1zL3JvbGUioiJBZ
G1pbm1zdHJhd9yIiwiCGVybwlzclvbi6wyJwZXJtaXNzaW9ucy5zY2h1ZHVsZXIudm1ldyIsInBlcm1pc3Npb25zLnNjaGVkdWxlci5tYW5hZ2UiLCJwZXJ
taXNzaW9ucy5yZBvvnQudm1ldyJdLC1leHAi0je2NjcyHyyNjJ9. Icy0pRU6oyCwN2DKciRdH3CL6YSenMHJJnEBp5ztPQ",
"refreshToken": "SdfeuSAadTL704WQCRic0TC8vDiuUgh+azWWQBXwzhFM=",
"refreshTokenExpiryTime": "2022-11-05T19:11:02.2811017+01:00",
"userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092"
},
" isSuccess": true,
" statusCode": "OK"
}

```

Figura 2.11: Risposta Login API

POST RefreshToken

<https://localhost:7162/v1/authentication/refresh>

API that refreshes the specified JWT token.

BODY raw

```
{
  "token": "*****",
  "refreshToken": "*****"
}
```

Figura 2.12: Refresh Token API

```

"response": {
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJodHRwOi8vC2NoZW1hcy54bWxzB2FwlM9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYwltcy9uYW1laWRlbnRpZmlciI6ImY0ZGE0ZGz1LTc10TMTNDU1Ny04Yjg2L
WN10GMxZTcwNzASMiisImh0dHA6Ly9zY2hlbwFzLnhbHNvYXAub33nL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL2VtYwlsYwRkcmVzcyI6InNyb256b25p0t1AZ2
1haWwuY29tIiwiHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9pZGVudG10eS9jbGfpbXvbmFtZS16IlNpbw9uZSiisImh0dHA6Ly9zY2hlbwFzLnh
tbHNVvXAut3JnL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL3N1cm5hbWUi0iJSb256b25pIiwiHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9p
ZGVudG10eS9jbGfpbXvbmUi0iILCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDgvMDgvY2xhaW1zL3JvbGUioiJBZ
G1pbm1zdHJhd9yIiwiCGVybwlzclvbi6wyJwZXJtaXNzaW9ucy5zY2h1ZHVsZXIudm1ldyIsInBlcm1pc3Npb25zLnNjaGVkdWxlci5tYW5hZ2UiLCJwZXJ
taXNzaW9ucy5yZBvvnQudm1ldyJdLC1leHAi0je2NjcyMzYMTZ9. PQTWhswkzyn94L_6w_ojsA89UzhywZvp94s0Ty-k084",
"refreshToken": "hT5MEQm7wlCHiLWAti3hY1SO4p0cmxhxD3FhANKk7o=",
"refreshTokenExpiryTime": "2022-11-05T19:11:02.2811017",
"userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092"
},
" isSuccess": true,
" statusCode": "OK"
}

```

Figura 2.13: Risposta Refresh Token API

Iterazione 3

3.1 Descrizione

In questa iterazione si sono progettati e sviluppati i casi d'uso relativi alla gestione degli account operatori (UC4.1, UC4.2, UC4.3, UC4.4, UC4.5, UC4.6).

3.2 Casi d'uso

3.2.1 UC4.1.1 - Visualizzazione account autenticato

Aggiunto rispetto a quanto previsto in iterazione 1.

UC4.1.1	Visualizzazione account autenticato
<i>Descrizione:</i>	Visualizzazione dell'account autenticato
<i>Attori:</i>	Amministratore, operatore
<i>Precondizioni:</i>	L'utente è autenticato

3.2.2 UC4.1.2 - Visualizzazione di tutti gli account

UC4.1.2	Visualizzazione di tutti gli account
<i>Descrizione:</i>	Visualizzazione degli account di amministrazione e operatori
<i>Attori:</i>	Amministratore
<i>Precondizioni:</i>	L'utente è autenticato come amministratore

3.2.3 UC4.2 - Aggiunta account operatore

UC4.2	Aggiunta account operatore
<i>Descrizione:</i>	Aggiunta di un account relativo a un operatore
<i>Attori:</i>	Amministratore
<i>Precondizioni:</i>	L'utente è autenticato come amministratore
<i>Flusso eventi:</i>	<ol style="list-style-type: none"> 1. L'amministratore inserisce i dati dell'operatore 2. L'amministratore inserisce una email e password relativa a quell'account 3. Il sistema salva tale account
<i>Postcondizioni:</i>	Reindirizzamento alla pagina di visualizzazione degli utenti
<i>Eccezioni:</i>	Esiste già un account con quelle credenziali

3.2.4 UC4.3 - Aggiunta account amministratore

UC4.3	Aggiunta account amministratore
<i>Descrizione:</i>	Aggiunta di un account relativo a un amministratore
<i>Attori:</i>	Amministratore
<i>Precondizioni:</i>	L'utente è autenticato come amministratore
<i>Flusso eventi:</i>	<ol style="list-style-type: none"> 1. L'amministratore inserisce i dati del nuovo utente 2. L'amministratore inserisce una email e password relativa a quell'account 3. Il sistema salva tale account
<i>Postcondizioni:</i>	Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

3.2.5 UC4.4 - Modifica account operatore

UC4.4	Modifica account operatore
--------------	-----------------------------------

Descrizione: Modifica di un account relativo ad un operatore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore inserisce i dati del nuovo utente
2. L'amministratore inserisce una email e password relativa a quell'account
3. Il sistema salva tale account

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

3.2.6 UC4.5 - Modifica account amministratore

UC4.5	Modifica account amministratore
--------------	--

Descrizione: Modifica di un account relativo ad un amministratore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore inserisce i dati del nuovo utente
2. L'amministratore inserisce una email e password relativa a quell'account
3. Il sistema salva tale account

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

3.2.7 UC4.6 - Eliminazione account

UC4.6	Eliminazione account
--------------	-----------------------------

Descrizione: Eliminazione di un account relativo ad un operatore o ad un amministratore

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. L'amministratore seleziona l'account da eliminare e lo rimuove

Postcondizioni: Reindirizzamento alla pagina di visualizzazione degli utenti

Eccezioni: Esiste già un account con quelle credenziali

3.3 Diagramma dei componenti

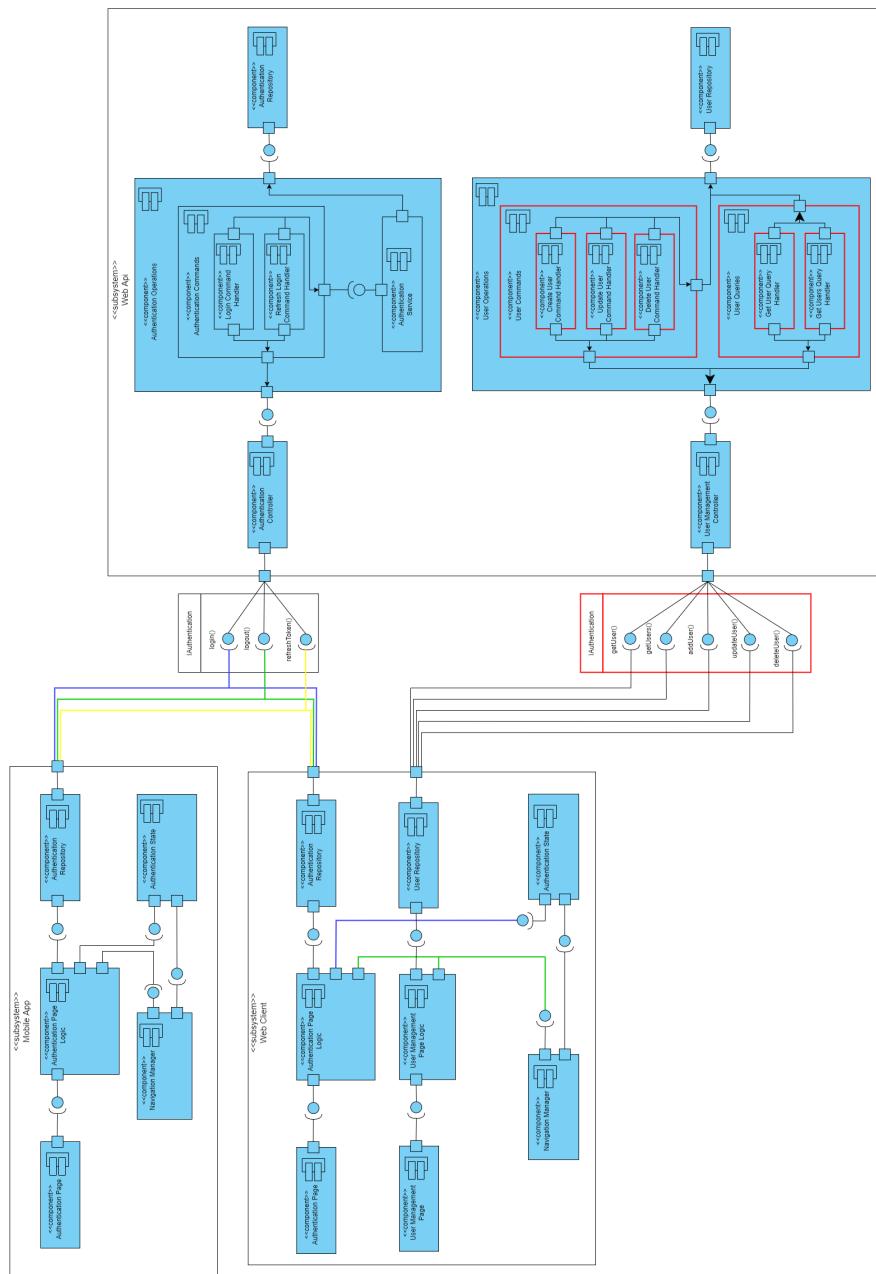


Figura 3.1: Diagramma dei componenti

3.4 Diagramma entità relazioni (ER)

Rispetto alla precedente iterazione non vi è stata apportata alcuna modifica.

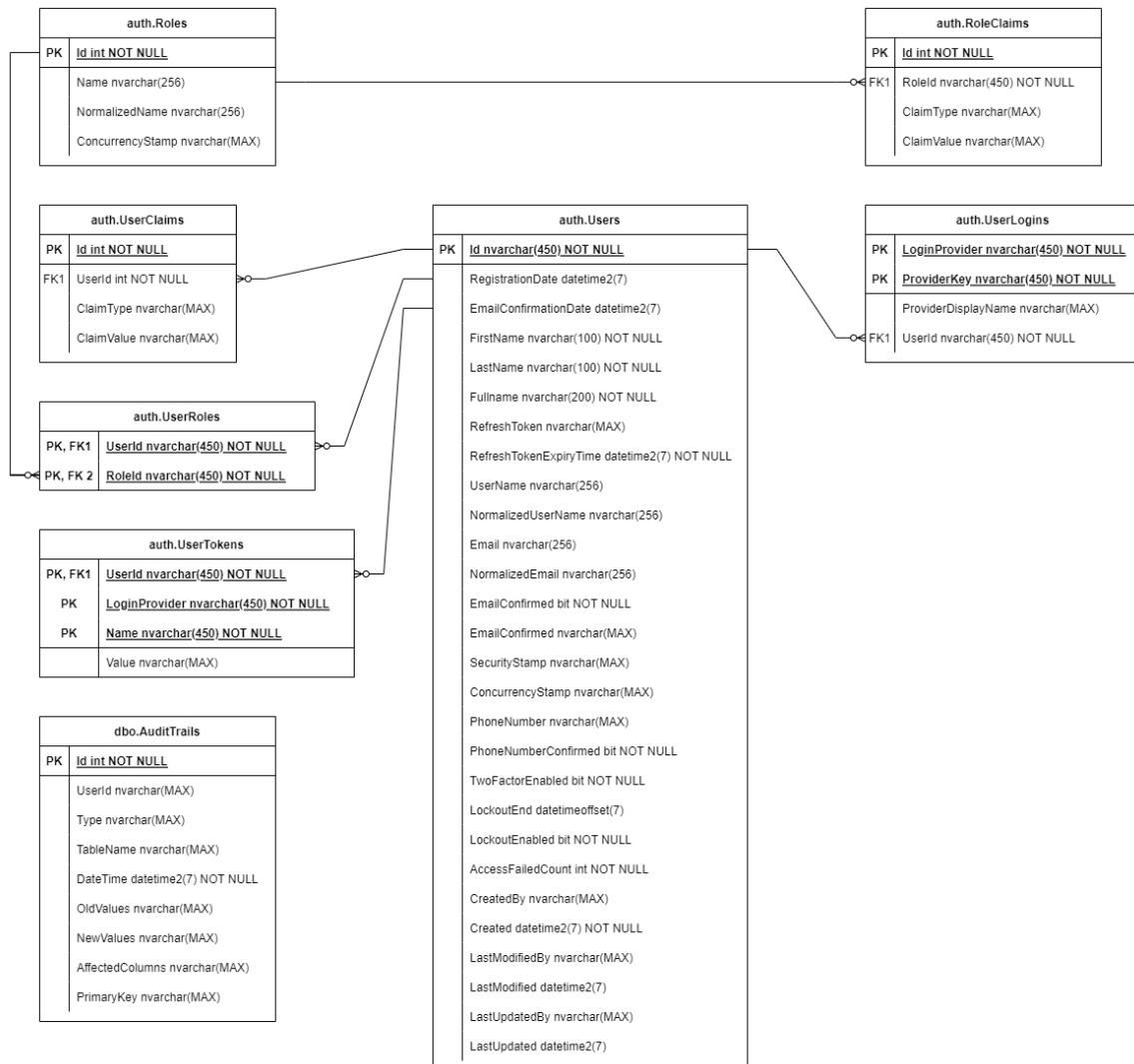


Figura 3.2: Diagramma entità relazioni

3.5 Diagramma delle classi

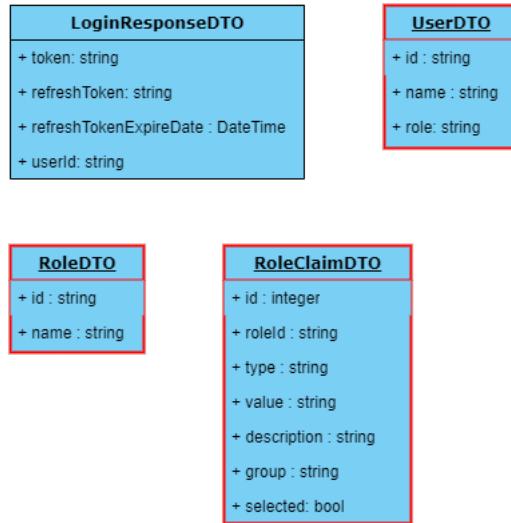


Figura 3.3: Diagramma delle classi (tipi di dati)

3.6 Diagramma delle interfacce

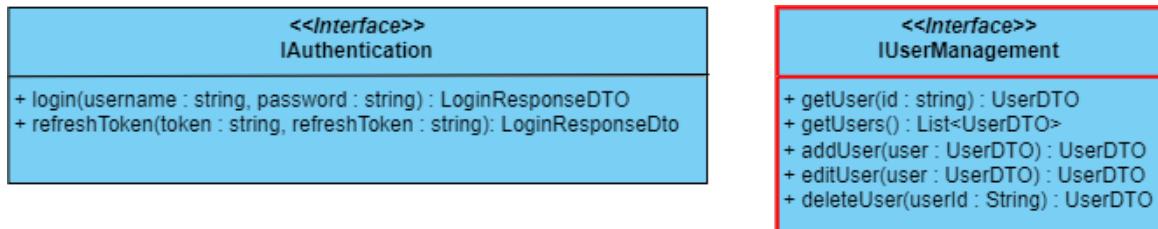


Figura 3.4: Diagramma delle classi (interfacce)

3.7 Diagramma dei package

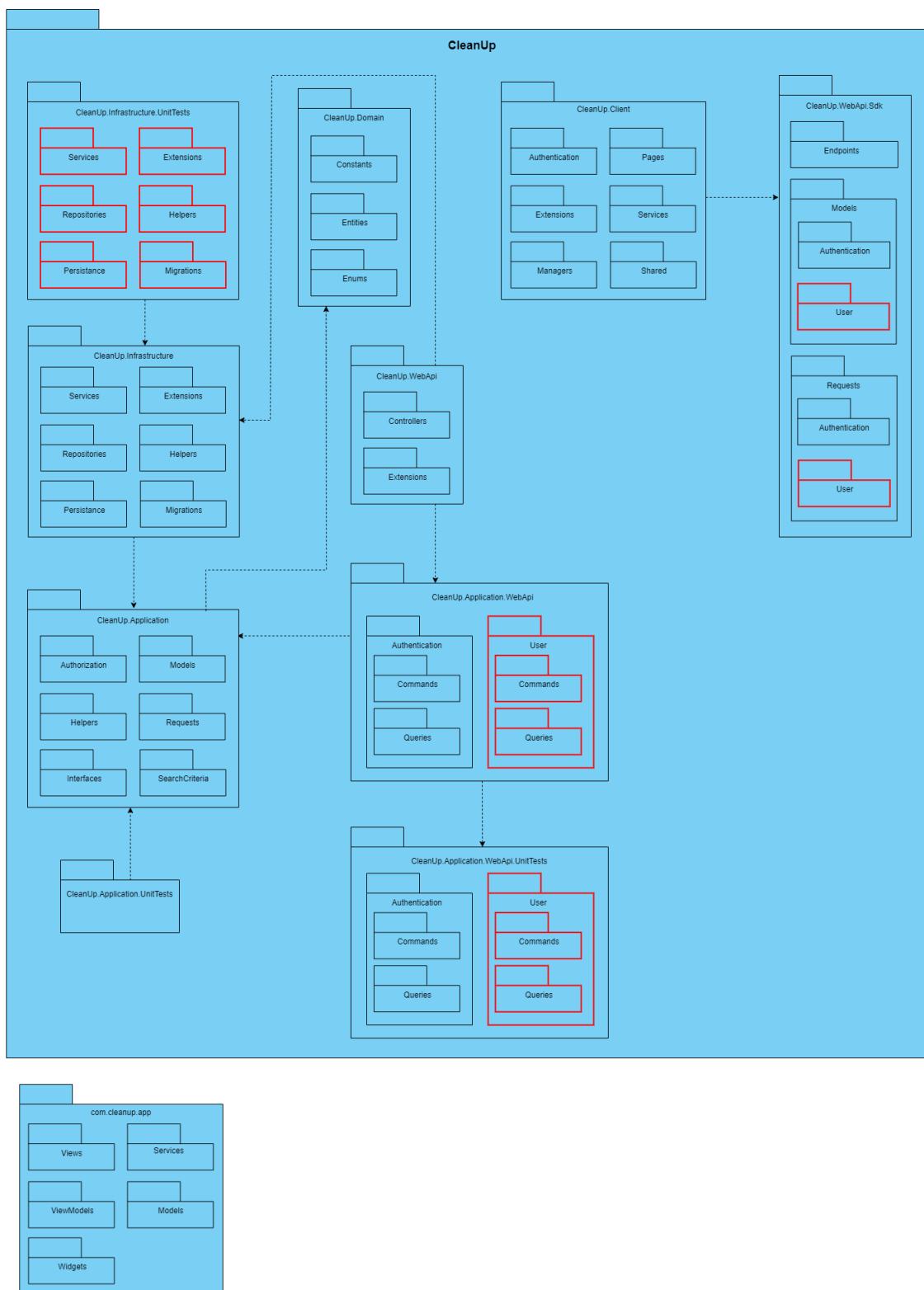


Figura 3.5: Diagramma dei package

3.8 Testing

3.8.1 Test delle API

```
GET GetUser https://localhost:7162/v1/user/f4da4dfe-7593-4557-8b86-cb8c1e707092 / Users / GetUser

Pass Status code is 200
Pass Response JSON Schema is correct
Pass Empty ID
Pass Not existing ID
```

Figura 3.6: Test API Get User

```
GET GetEvents https://localhost:7162/v1/event?fromDate=2022-10-24T22:50:14.785Z&toDate=2022-11-24T22:50:14.785Z&ClassroomId=A001%20|Edificio%20A%20-%20Bamme | / Events / GetEvents

Pass Status code is 200
Pass Response JSON Schema is correct
```

Figura 3.7: Test API Get Users

```
POST CreateUser https://localhost:7162/v1/user / Users / CreateUser

Pass Status code is 200
Pass Response JSON Schema is correct
Pass Insert of duplicate email
Pass Not specified Email
Pass Not specified Email
```

Figura 3.8: Test API Create User

PUT UpdateUser https://localhost:7162/v1/user/5f4dd598-2740-4597-b297-8154dbd197ca / Users / UpdateUser	
Pass	Status code is 200
Pass	Response JSON Schema is correct
Pass	Empty ID
Pass	Invalid ID

Figura 3.9: Test API Update Users

DELETE DeleteUser https://localhost:7162/v1/user/5f4dd598-2740-4597-b297-8154dbd197ca / Users / DeleteUser	
Pass	Status code is 200
Pass	Response JSON Schema is correct
Pass	Empty ID
Pass	Not existing ID

Figura 3.10: Test API Delete Users

3.9 Analisi statica del codice

Di seguito si riportano le metriche di qualità del codice fornite dallo strumento integrato di Visual Studio includendo le modifiche dell'iterazione 3

Gerarchia ▾	Indice di manutenibilità	Complessità ciclomatica	Profondità dell'ereditarietà	Accoppiamenti di classi	Righe di codice ...	Righe di codice ...
▷ src\CleanUp.Application (Debug)	95	82	4	19	206	23
▷ src\CleanUp.Application.WebApi (Debug)	88	103	2	54	664	116
▷ src\CleanUp.Domain (Debug)	99	29	3	9	54	1
▷ src\CleanUp.Infrastructure (Debug)	69	91	6	144	2.815	798
▷ src\Hangfire\CleanUp.Hangfire (Debug)	86	102	4	83	668	139
▷ src\Hangfire\CleanUp.Hangfire.Data (Debug)	77	4	1	46	86	19
▷ src\Sdk\CleanUp.WebApi.Sdk (Debug)	93	126	2	25	285	39
▷ src\WebCleanUp.Client (Debug)	87	791	3	192	9.765	912
▷ src\WebCleanUp.WebApi (Debug)	84	182	4	222	1.182	310
▷ test\CleanUp.Infrastructure.UnitTests (Debug)	78	6	1	42	174	82

Figura 3.11: Metriche di qualità del codice

3.10 Documentazione API

Users

Iteration 3:

User management APIs

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection [API Test](#)

GET GetUser

<https://localhost:7162/v1/user/f4da4dfe-7593-4557-8b86-cb8c1e707092>

API that returns the data of the user specified by his id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

Figura 3.12: Get User API

```

  "response": {
    "id": "f4da4dfe-7593-4557-8b86-cb8c1e707092",
    "firstName": "Simone",
    "lastName": "Ronzoni",
    "fullName": "Ronzoni Simone",
    "email": "sronzoni99@gmail.com",
    "emailConfirmed": true,
    "userName": "sronzoni99@gmail.com",
    "refreshToken": "hT5MEQm7wlCHlWAti3hY1S04p0cmxhxVd3FhANkK7o=",
    "refreshTokenExpiryTime": "2022-11-05T19:11:02.2811017"
  },
  "isSuccess": true,
  "statusCode": "OK"

```

Figura 3.13: Risposta Get User API

GET GetUsers 🔒

https://localhost:7162/v1/user

API that returns the data of all the users.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

Figura 3.14: Get Users API

```
{
  "response": [
    {
      "id": "277b0b13-b689-4dc2-95eb-56596f79c4f1",
      "firstName": "Xhorxho",
      "lastName": "Papallazi",
      "fullName": "Papallazi Xhorxho",
      "email": "xhorxho.papallazi@gmail.com",
      "emailConfirmed": false,
      "userName": "xhorxho.papallazi@gmail.com",
      "phoneNumber": "1238965493",
      "emailConfirmationDate": "2022-10-25T00:57:42.6518645",
      "refreshTokenExpiryTime": "0001-01-01T00:00:00"
    }
  ]
}
```

Figura 3.15: Risposta Get Users API

POST CreateUser 🔒

https://localhost:7162/v1/user

API that creates a new user.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY raw

```
{
  "name": "Filippo",
  "surname": "Bordogna",
  "phoneNumber": "2458975643",
  "email": "bordognapippo99@gmail.com",
  "emailConfirmed": true,
  "password": "Password2",
  "phoneNumberConfirmed": true
}
```

Figura 3.16: Create User API

```

{
  "response": {
    "id": "790a17d9-079c-4fa8-8676-afebaf04ce9",
    "firstName": "Filippo",
    "lastName": "Bordogna",
    "fullName": "Bordogna Filippo",
    "email": "bordognapippo99@gmail.com",
    "emailConfirmed": true,
    "userName": "bordognapippo99@gmail.com",
    "phoneNumber": "2458975643",
    "emailConfirmationDate": "2022-10-29T19:21:31.3850782+02:00",
    "refreshTokenExpiryTime": "0001-01-01T00:00:00"
  },
  "isSuccess": true,
  "statusCode": "OK"
}

```

Figura 3.17: Risposta Create User API

PUT UpdateUser

<https://localhost:7162/v1/user/ebe194c5-b6c5-413e-8e49-3fb7158cd186>

API that updates the data of the user specified by his id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY raw

```
{
  "id": "ebe194c5-b6c5-413e-8e49-3fb7158cd186",
  "name": "Pippo",
  "surname": "Bordo",
  "phoneNumber": "4569873026"
}
```

Figura 3.18: Update Users API

```

{
  "response": {
    "id": "790a17d9-079c-4fa8-8676-afebaf04ce9",
    "firstName": "Filippo Tommaso",
    "lastName": "Bordogna",
    "fullName": "Bordogna Filippo Tommaso",
    "email": "bordognapippo99@gmail.com",
    "emailConfirmed": true,
    "userName": "bordognapippo99@gmail.com",
    "phoneNumber": "4569873026",
    "emailConfirmationDate": "2022-10-29T19:21:31.3850782",
    "refreshTokenExpiryTime": "0001-01-01T00:00:00"
  },
  "isSuccess": true,
  "statusCode": "OK"
}

```

Figura 3.19: Risposta Update User API

DEL DeleteUser 🔒

<https://localhost:7162/v1/user/eeb194c5-b6c5-413e-8e49-3fb7158cd186>

API that deletes the user specified by his id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

Figura 3.20: Delete Users API

```
response": {
    "id": "f9a6a191-baca-4eff-9021-619f746b398d",
    "firstName": "Filippo Tommaso",
    "lastName": "Bordogna",
    "fullName": "Bordogna Filippo Tommaso",
    "email": "bordognapippo99@gmail.com",
    "emailConfirmed": true,
    "userName": "bordognapippo99@gmail.com",
    "phoneNumber": "4569873026",
    "emailConfirmationDate": "2022-10-29T19:27:27.6427445",
    "refreshTokenExpiryTime": "0001-01-01T00:00:00"
},
"isSuccess": true,
"statusCode": "OK"
```

Figura 3.21: Risposta Delete User API

Iterazione 4

4.1 Descrizione

In questa iterazione si sono progettati e sviluppati i casi d'uso relativi alla gestione eventi (UC6.1, UC6.2).

4.2 Casi d'uso

4.2.1 UC6.1 - Importazione eventi

UC6.1	Importazione eventi
<i>Descrizione:</i>	Importazione degli eventi che si svolgono in una giornata
<i>Attori:</i>	Amministratore
<i>Precondizioni:</i>	L'utente è autenticato come amministratore
<i>Flusso eventi:</i>	<ol style="list-style-type: none">1. L'amministratore carica l'elenco degli eventi2. Il sistema computa gli elenchi di interventi di pulizia3. Il sistema avvisa mediante notifica push gli operatori che sono stati assegnati ad una lista di pulizia
<i>Postcondizioni:</i>	Visualizzazione dell'elenco di interventi di pulizia
<i>Eccezioni:</i>	L'amministratore aggiunge eventi con data passata

4.2.2 UC6.2 - Modifica eventi

UC6.2	Modifica eventi
<i>Descrizione:</i>	Modifica evento
<i>Attori:</i>	Amministratore
<i>Precondizioni:</i>	L'utente è autenticato come amministratore. Ci sono già degli eventi per quel giorno.
<i>Flusso eventi:</i>	<ol style="list-style-type: none">1. L'amministratore carica l'elenco degli eventi aggiornati2. Il sistema elabora le modifiche agli elenchi di interventi di pulizia3. Il sistema avvisa mediante notifica push solo gli operatori interessati dalle modifiche
<i>Postcondizioni:</i>	Visualizzazione elenco interventi pulizia aggiornati
<i>Eccezioni:</i>	L'amministratore va a modificare eventi in corso: in questo caso non si può fare nulla

4.3 Diagramma dei componenti

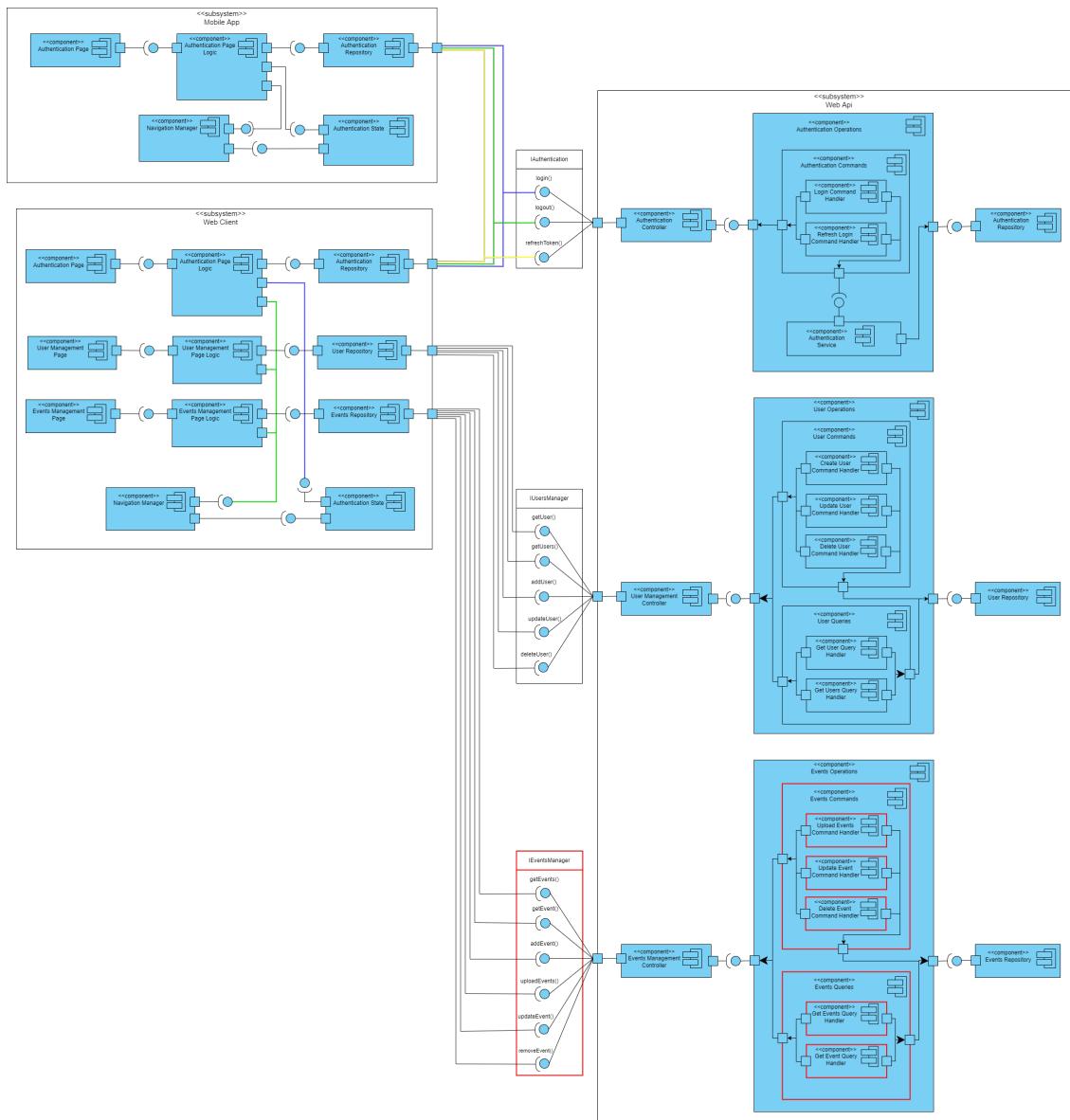


Figura 4.1: Diagramma dei componenti

4.4 Diagramma entità relazioni (ER)

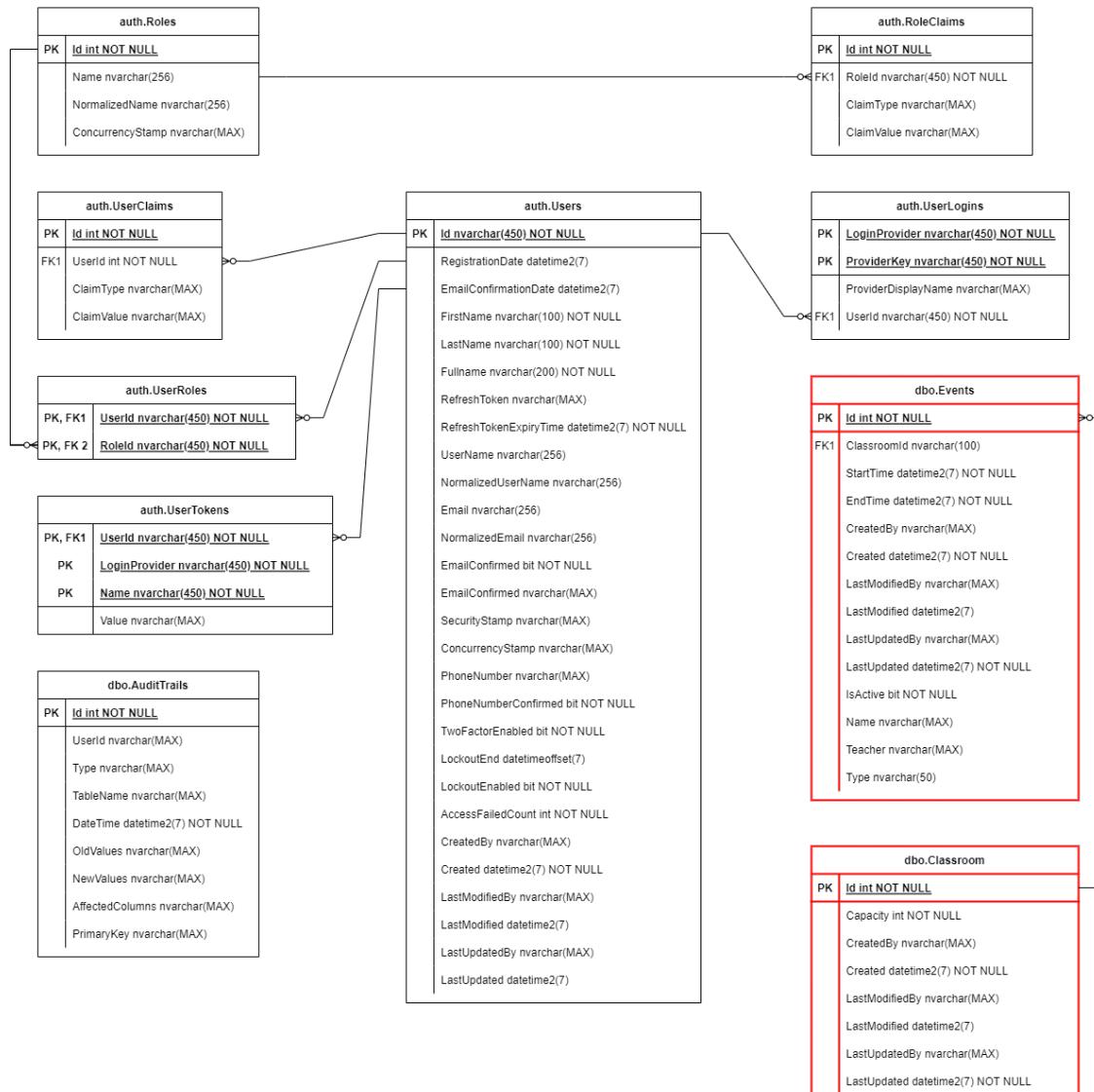


Figura 4.2: Diagramma entità relazioni

4.5 Diagramma delle classi

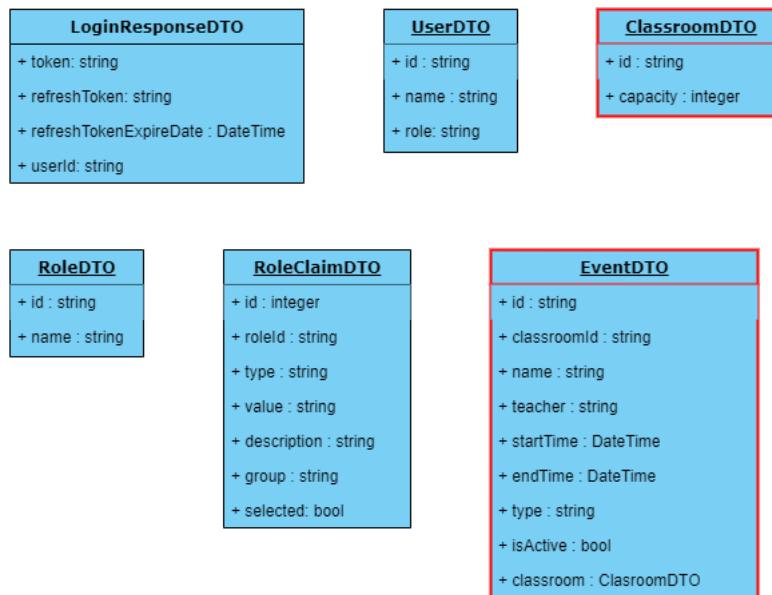


Figura 4.3: Diagramma delle classi (tipi di dati)

4.6 Diagramma delle interfacce

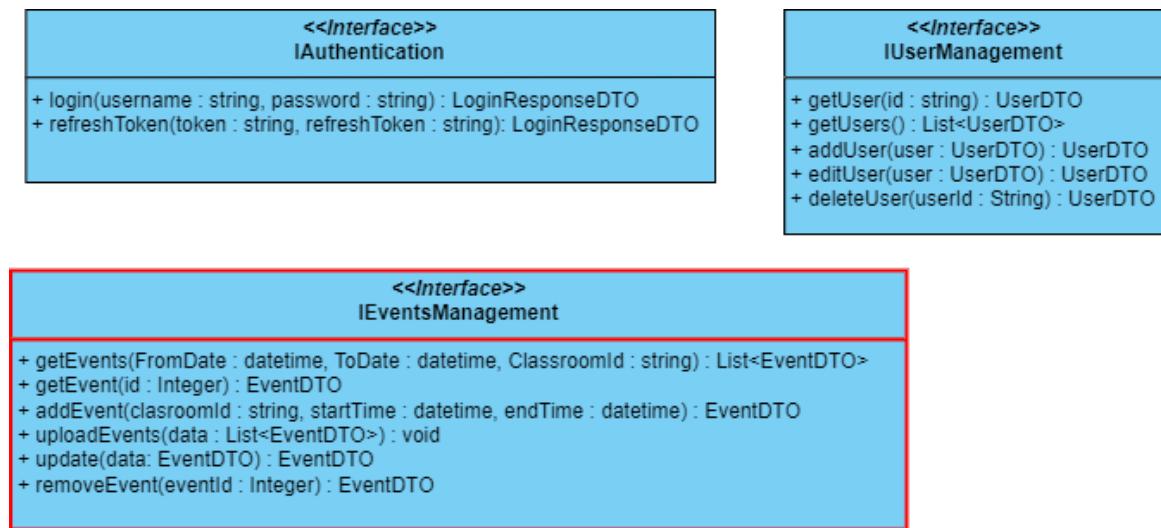


Figura 4.4: Diagramma delle classi (interfacce)

4.7 Diagramma dei package

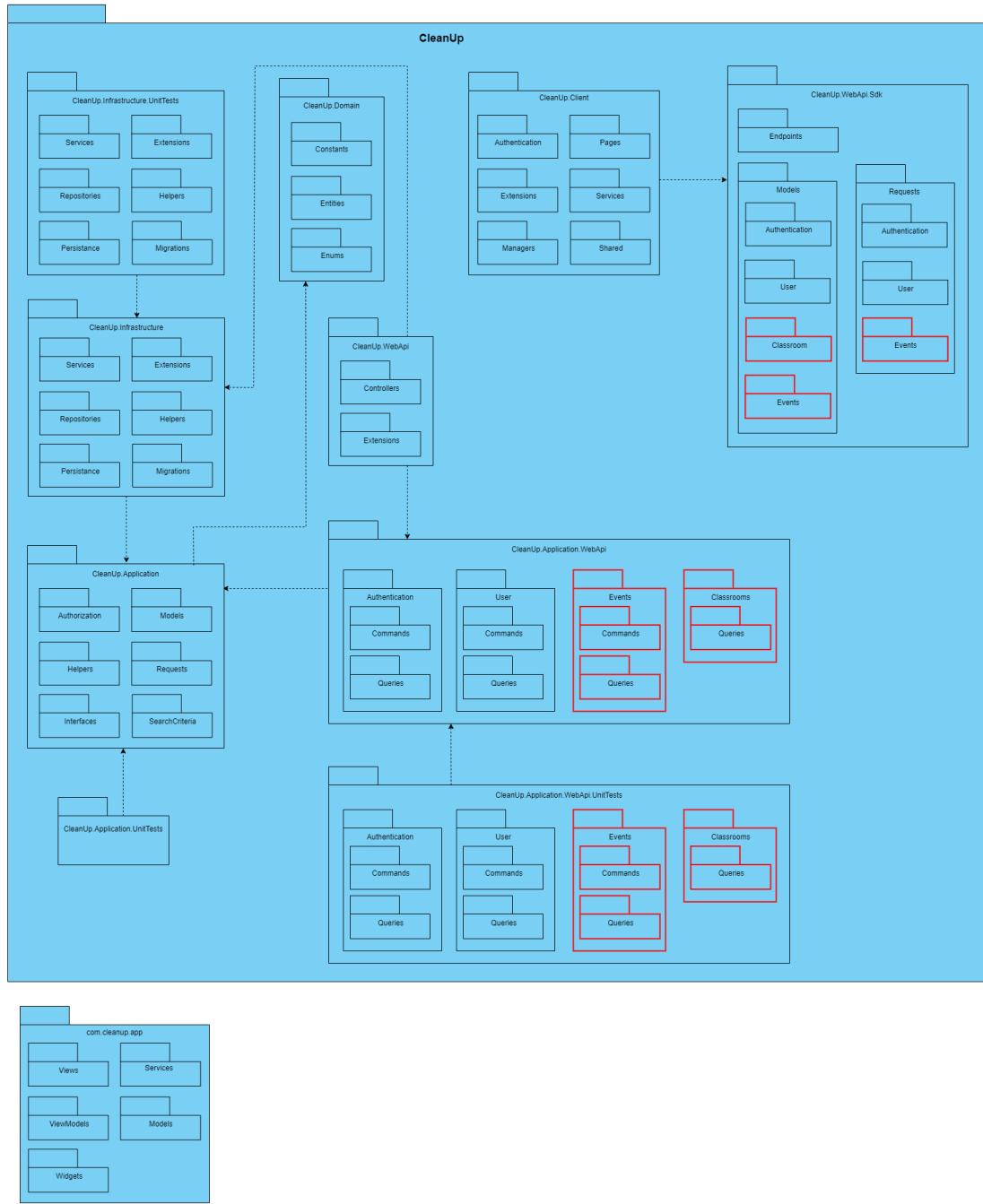


Figura 4.5: Diagramma dei package

4.8 Testing

4.8.1 Unit test

In questa iterazione la funzionalità critica riguarda l'upload massivo degli eventi, importandoli da file excel. Tramite questi unit test si è verificato il corretto comportamento.

CleanUp.Application.WebApi.UnitTests.Events (4)		567 ms
DeleteEventCommandHandlerTests (1)	Pass	102 ms
DeleteCustomerCommandHandler_WhenEventExits	Pass	102 ms
UploadEventCommandHandlerTests (3)	Pass	465 ms
UploadCustomerCommandHandler_LoadCorrectData_WhenBigFileCorrect...	Pass	136 ms
UploadCustomerCommandHandler_LoadCorrectData_WhenSmallFileCorr...	Pass	329 ms
UploadCustomerCommandHandler_ThrowsException_WhenWrongFileFor...	Pass	< 1 ms

Figura 4.6: Unit test Upload Events

4.8.2 Test delle API

GET GetEvents https://localhost:7162/v1/event?FromDate=2022-10-24T22:50:14.785Z&ToDate=2022-11-24T22:50:14.785Z&ClassroomId=A001%20[Edificio%20A%20-%20Dalmine]		/ Events / GetEvents
Pass	Status code is 200	
Pass	Response JSON Schema is correct	

Figura 4.7: Test API Get Events

GET GetEvent https://localhost:7162/v1/event/42		/ Events / GetEvent
Pass	Status code is 200	
Pass	Response JSON Schema is correct	

Figura 4.8: Test API Get Event

POST CreateEvent <https://localhost:7162/v1/event> / Events / CreateEvent

- Pass Status code is 200
- Pass Response JSON Schema is correct

Figura 4.9: Test API Create Event

PUT UpdateEvent <https://localhost:7162/v1/event/462> / Events / UpdateEvent

- Pass Status code is 200
- Pass Response JSON Schema is correct
- Pass Empty classroomId
- Pass Empty event name
- Pass startTime > endTime
- Pass Invalid ID

Figura 4.10: Test API Update Event

DELETE DeleteEvent <https://localhost:7162/v1/event/462> / Events / DeleteEvent

- Pass Status code is 200
- Pass Response JSON Schema is correct
- Pass Invalid ID

Figura 4.11: Test API Delete Event

POST UploadEvents <https://localhost:7162/v1/event/upload> / Events / UploadEvents

- Pass Status code is 200
- Pass Response JSON Schema is correct

Figura 4.12: Test API Upload Events

GET GetClassrooms <https://localhost:7162/v1/classroom> / Classrooms / GetClassrooms

- Pass Status code is 200
- Pass Response JSON Schema is correct

Figura 4.13: Test API Get Classrooms

4.9 Analisi statica del codice

Di seguito si riportano le metriche di qualità del codice fornite dallo strumento integrato di Visual Studio includendo le modifiche dell'iterazione 4

Gerarchia	Indice di manutenibilità	Complessità ciclomatica	Profondità dell'ereditarietà	Accoppiamenti di classi	Righe di codice ...	Righe di codice ...
▷ CleanUp.Infrastructure.UnitTests (Debug)	78	6	1	42	174	82
▷ src\CleanUp.Application (Debug)	94	93	4	22	250	36
▷ src\CleanUp.Application.WebApi (Debug)	88	197	2	84	1.089	253
▷ src\CleanUp.Domain (Debug)	99	37	3	9	58	1
▷ src\CleanUp.Infrastructure (Debug)	65	97	6	147	3.866	1.121
▷ src\Hangfire\CleanUp.Hangfire (Debug)	86	102	4	83	668	139
▷ src\Hangfire\CleanUp.Hangfire.Data (Debug)	77	4	1	46	86	19
▷ src\Sdk\CleanUp.WebApi.Sdk (Debug)	94	169	2	26	337	48
▷ src\Web\CleanUp.Client (Debug)	85	1.066	3	211	13.704	1.539
▷ src\Web\CleanUp.WebApi (Debug)	83	186	4	232	1.224	322
▷ test\CleanUp.Application.WebApi.IntegrationTests (Debug)	79	9	1	42	208	58

Figura 4.14: Metriche di qualità del codice

4.10 Documentazione API

Events

Iteration 4:

Event management APIs.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection [API Test](#)

GET GetEvents 🔒

`https://localhost:7162/v1/event?FromDate=2022-10-24T22:50:14.785Z&ToDate=2022-11-24T22:50:14.785Z&ClassroomId=A001 [Edificio A - Dalmine]`

API that returns the data of all the events carried out in the specified classroom and time frame.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

PARAMS

FromDate 2022-10-24T22:50:14.785Z

ToDate 2022-11-24T22:50:14.785Z

ClassroomId A001 [Edificio A - Dalmine]

Figura 4.15: Get Events API

```

"response": [
    {
        "id": 14,
        "classroomId": "A001 [Edificio A - Dalmine]",
        "name": "CHIMICA",
        "teacher": "Isabella Natali Sora",
        "startTime": "2022-11-23T08:30:00",
        "endTime": "2022-11-23T10:30:00",
        "type": "Lezione",
        "isActive": true
    },
    {
        "id": 24,
        "classroomId": "A001 [Edificio A - Dalmine]",
        "name": "MACCHINE A FLUIDO",
        "teacher": "Giovanna Barigozzi, Nicoletta Franchina, Giovanni Brumana",
        "startTime": "2022-11-23T10:30:00",
        "endTime": "2022-11-23T13:30:00",
        "type": "Lezione",
        "isActive": true
    }
]

```

Figura 4.16: Risposta Get Events API

GET GetEvent

<https://localhost:7162/v1/event/42>

API that returns the data of the event specified by its id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

Figura 4.17: Get Event API

```

{
  "response": {
    "id": 42,
    "classroomId": "A001 [Edificio A - Dalmine]",
    "name": "ANALISI MATEMATICA I",
    "teacher": "Luca Brandolini",
    "startTime": "2022-11-23T13:30:00",
    "endTime": "2022-11-23T15:30:00",
    "type": "Lezione",
    "isActive": true,
    "classroom": {
      "id": "A001 [Edificio A - Dalmine]",
      "capacity": 0
    }
  },
  "isSuccess": true,
  "statusCode": "OK"
}

```

Figura 4.18: Risposta Get Event API

POST CreateEvent

<https://localhost:7162/v1/event>

API that creates a new event.

If the classroom is not in the database it will be created.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY raw

```
{
  "classroomId": "A001 [Edificio A - Dalmine]",
  "startTime": "1999-10-10T23:09:30.000Z",
  "endTime": "1999-10-10T23:11:30.000Z"
}
```

Figura 4.19: Create Event API

```

{
  "response": {
    "id": 463,
    "classroomId": "A001 [Edificio A - Dalmine]",
    "startTime": "1999-10-10T23:09:30Z",
    "endTime": "1999-10-10T23:11:30Z",
    "isActive": false
  },
  "isSuccess": true,
  "statusCode": "OK"
}

```

Figura 4.20: Risposta Create Event API

PUT UpdateEvent 

<https://localhost:7162/v1/event/452>

API that updates the event specified by its id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY raw

```
{
  "classroomId": "A001 [Edificio A - Dalmine]",
  "name": "STATISTICA E MODELLI STOCASTICI",
  "teacher": "Filippo Bordogna",
  "startTime": "1999-10-10T14:00:00.000Z",
  "endTime": "1999-10-10T16:30:00.000Z",
  "isActive": true
}
```

Figura 4.21: Update Event API

```

  "response": {
    "id": 463,
    "classroomId": "A001 [Edificio A - Dalmine]",
    "name": "STATISTICA E MODELLI STOCASTICI",
    "teacher": "Filippo Bordogna",
    "startTime": "1999-10-10T14:00:00Z",
    "endTime": "1999-10-10T16:30:00Z",
    "isActive": true
  },
  "isSuccess": true,
  "statusCode": "OK"

```

Figura 4.22: Risposta Update Event API

DEL DeleteEvent 

<https://localhost:7162/v1/event/452>

API that deletes the event specified by its id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

Figura 4.23: Delete Event API

```
        "response": {
            "id": 463,
            "classroomId": "A001 [Edificio A - Dalmine]",
            "name": "STATISTICA E MODELLI STOCASTICI",
            "teacher": "Filippo Bordogna",
            "startTime": "1999-10-10T14:00:00",
            "endTime": "1999-10-10T16:30:00",
            "isActive": true
        },
        "isSuccess": true,
        "statusCode": "OK"
    }
```

Figura 4.24: Risposta Delete Event API

POST UploadEvents

<https://localhost:7162/v1/event/upload>

API that uploads an excel file containing the events.
If the event is already in the database it will be updated otherwise it will be created.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY formdata

files

Figura 4.25: Upload Events API

```
        "isSuccess": true,
        "statusCode": "OK"
    }
```

Figura 4.26: Risposta Upload Events API

Classrooms

Iteration 4:

Classroom management API.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection [API Test](#)

GET GetClassrooms 🔒

<https://localhost:7162/v1/classroom>

API that returns the data of all the classrooms.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

Figura 4.27: Get Classrooms API

```
{
  "response": [
    {
      "id": "A001 [Edificio A - Dalmine]",
      "capacity": 0
    },
    {
      "id": "A002 [Edificio A - Dalmine]",
      "capacity": 0
    },
    {
      "id": "A003 [Edificio A - Dalmine]",
      "capacity": 0
    }
  ]
}
```

Figura 4.28: Risposta Get Classrooms API

Iterazione 5

5.1 Descrizione

In questa iterazione si sono progettati e sviluppati i casi d'uso relativi allo scheduler (UC5, UC7, UC8).

5.2 Casi d'uso

5.2.1 UC5 - Assegnazione giornaliera operatori

UC5	Assegnazione giornaliera operatori
<i>Descrizione:</i>	Caricamento dell'orario di lavoro degli operatori
<i>Attori:</i>	Amministratore
<i>Precondizioni:</i>	L'utente è autenticato come amministratore
<i>Flusso eventi:</i>	<ol style="list-style-type: none">1. L'utente specifica l'orario nel quale gli operatori sono disponibili in un dato giorno2. Il sistema effettua una verifica dei dati inseriti dall'utente<ol style="list-style-type: none">a. Se i dati sono corretti, il sistema salva lo stato degli operatorib. Se i dati contengono errori, essi vengono mostrati all'utente
<i>Postcondizioni:</i>	-

Eccezioni: L'utente inserisce un piano di disponibilità errato

5.2.2 UC7 - Schedulazione interventi pulizia

UC7	Schedulazione interventi pulizia
------------	---

Descrizione: Richiamare lo scheduler per forzare l'aggiornamento della schedule

Attori: Amministratore

Precondizioni: L'utente è autenticato come amministratore

Flusso eventi:

1. Viene richiamato il componente che implementa l'algoritmo di scheduling

Postcondizioni: -

Eccezioni: L'algoritmo non ha eventi da rischedulare

5.2.3 UC8 - Visualizzazione elenco attività pulizia

UC8	Visualizzazione elenco attività pulizia
------------	--

Descrizione: Visualizzazione dell'elenco degli interventi di pulizia di un certo operatore

Attori: Amministratore, operatore

Precondizioni: L'utente è autenticato come operatore o amministratore

Flusso eventi:

1. Viene caricato e mostrato l'elenco degli interventi di pulizia di un operatore specificato

a. Se l'account è di un amministratore può richiedere qualsiasi utente

b. Se l'account è di un operatore può accedere solo al suo

Postcondizioni: -

Eccezioni: L'utente che richiede non ha interventi di pulizia da effettuare.

Un operatore cerca di richiedere l'elenco attività non sue.

5.3 Scheduler

Il problema della schedulazione è un problema ricorrente in vari aspetti del mondo dell'informatica e in generale nella vita quotidiana. Questo progetto ha come obiettivo la generazione di una schedule, per quanto possibile ottimale, degli interventi di pulizia all'interno della nostra università, potendo essere riutilizzabile anche in altri contesti che condividono requisiti simili.

Si è iniziato il processo di progettazione andando ad eseguire una ricerca approfondita su problemi simili e tecniche utilizzate per risolverli; sono stati trovati due problemi in letteratura che rassomigliavano il nostro caso:

- Problema della schedulazione di task per un sistema multiprocessore: risolubile tramite strategie greedy con alcuni accorgimenti;
- Problema della generazione di un orario scolastico: risolubile tramite tecniche di ottimizzazione o algoritmi genetici.

Si è scartata la possibilità di studiare una soluzione basandosi sul secondo tipo visto che risolve problemi con ben più vincoli di quelli che si hanno e conduce ad una soluzione troppo complicata per il caso studiato. Si farà, da ora, riferimento al primo problema trovato e la soluzione si baserà su un adattamento di questi problemi al caso specifico studiato dal progetto.

L'algoritmo in questione dovrebbe permette di creare una schedule che rispetti i vincoli:

1. per quanto riguarda gli operatori delle pulizie dovranno essere rispetti i giorni di lavoro e soprattutto gli orari nei giorni di lavoro e quindi:

- Giorni di lavoro
- Orario di lavoro nei giorni di lavoro

2. per quanto riguarda gli interventi di pulizia, questi avvengono a seguito della

terminazione di una lezione in un'aula, devono essere svolti entro l'inizio della prossima lezione nella stessa aula con una certa durata e quindi:

- Orario di inizio
- Massimo orario di fine
- Durata

I vincoli di orario di inizio e orario di fine degli interventi di pulizia determinano la fascia oraria di disponibilità in cui può essere effettuato. L'urgenza di un intervento rispetto ad un altro è data dal rapporto tra il tempo necessario per la pulizia (durata) e il tempo in cui l'aula è libera prima dell'inizio dell'evento successivo (disponibilità). La possibilità di poter spostare l'orario di inizio intervento all'interno dell'orario di disponibilità potrebbe far pensare al bisogno di un algoritmo che ricerchi l'orario migliore all'interno della fascia oraria di disponibilità per far spazio ad altri interventi più urgenti e con vincoli più stringenti, ma questo porta ad una possibile valutazione di tutte le possibili permutazioni di tutti gli interventi giornalieri; tenendo poi conto del fatto che sono disponibili più di un solo operatore la situazione si complica ancora. Se vedessimo il problema sotto un'altra luce si potrebbe però pensare di utilizzare una strategia greedy che permetta di evitare il peso computazionale di una soluzione ad un problema apparentemente NP completo!

Rilassando il problema, rinunciando alla presenza di più processori e vedendo il sistema come un sistema monoprocesso (in cui il processore è il nostro operatore delle pulizie) con un set di task da assegnargli (ovvero gli interventi di pulizia), potremmo risolvere banalmente il problema osservando solo le deadline dei task. Di ogni intervento sono noti infatti l'orario entro cui va terminato e la sua durata, sappiamo che un intervento non può essere interrotto prima della sua durata e vogliamo quindi trovare una schedule che riesca ad eseguire più interventi possibile.

Riaggiungendo poi la possibilità di avere a disposizione più operatori si proverà a schedulare su uno di essi e se non possibile allora si potrà provare sugli altri (per bilanciare il carico di ogni operatore gli operatori saranno ordinati in base al numero di interventi già assegnatigli e si inizierà a provare a schedulare dal primo operatore con meno interventi assegnati).

5.3.1 Algoritmo

L'algoritmo per il calcolo dell'assegnazione giornaliera degli interventi agli operatori riceverà la lista di operatori attivi in quella giornata e la lista di interventi da eseguire nella stessa. Ogni intervento avrà 4 campi importanti quali:

1. *AvailableFrom*: che rappresenta l'orario nel quale l'intervento può iniziare
2. *AvailableTo*: che rappresenta invece l'orario oltre il quale non si può più eseguire l'intervento e che quindi assume il significato di *Deadline*
3. *CleaningDuration*: la durata dell'intervento di pulizia
4. *OperationStart*: non inizializzato, sarà valorizzato all'orario di inizio dell'intervento quando schedulato. Questo valore rispetterà le seguenti condizioni:

$$\textit{AvailableFrom} \leq \textit{OperationStart}$$

&

$$(\textit{OperationStart} + \textit{CleaningDuration}) \leq \textit{AvailableTo}$$

Al termine dell'esecuzione restituirà:

1. La lista degli interventi che non è stato possibile assegnare ad alcun operatore
2. La programmazione giornaliera di ogni operatore

Algoritmo 1 generatore della programmazione giornaliera

Require: operators, unorderedTasks

```

1: tasks  $\leftarrow$  unorderedTasks sorted by AvailableTo & AvailableTo – AvailableFrom
   – CleaningDuration
2: unscheduled  $\leftarrow$  new empty List // Vettore degli interventi che non possono essere
   schedulati
3: scheduledWithOp  $\leftarrow$  new empty Dictionary // La chiave sarà l'id dell'operatore
   mentre il valore un vettore ordinato temporalmente di interventi programmati
4: for index = 1, 2, . . . , minLength(operators, tasks) do
5:   Create new entry in scheduledWithOp with key $\leftarrow$ (i-th operator) and value $\leftarrow$ (i-th task) // Si inizia cercando di assegnare almeno un intervento ad ogni operatore
   disponibile, se gli interventi terminano allora non è necessario continuare
6: end for
7: for task in remaining tasks do
8:   scheduled  $\leftarrow$  false // Variabile che servirà a terminare la ricerca
9:   tryScheduleOrder  $\leftarrow$  keys of scheduledWithOp sorted by number of tasks already
   assigned // Si cercherà di assegnare prima agli operatori a cui sono stati
   assegnati meno interventi
10:  for index = 1, 2, . . . , length(tryScheduleOrder) do
11:    operatorSchedule  $\leftarrow$  value of key tryScheduleOrder[index] in scheduledWithOp // Variabile per referenziare il vettore degli interventi assegnati all'operatore
12:    for scheduledTask in operatorSchedule do
13:      if scheduledTask.OperationStart  $\geq$  task.AvailableTo then // Caso in
         cui la deadline viene mancata e bisogna provare con il prossimo operatore
14:        break
15:      end if
16:      if scheduledTask is first of operatorSchedule & task.AvailableFrom +
         task.CleaningDuration  $\leq$  scheduledSlot.OperationStart then // Caso in cui si può
         inserire in cima alla lista
17:        task.OperationStart  $\leftarrow$  task.AvailableFrom
18:        Schedule task with current operator
19:        scheduled  $\leftarrow$  true

```

```

20:      end if
21:       $end \leftarrow scheduledTask.OperationStart + scheduledTask.CleaningDuration$ 
22:       $maxStart \leftarrow \max(end, slot.AvailableFrom)$ 
23:      if scheduledTask is last of operatorSchedule & scheduledTask.OperationStart
+ scheduledTask.CleaningDuration + task.CleaningDuration  $\leq task.AvailableTo$ 
then // Caso in cui si può inserire alla fine della lista
24:          task.OperationStart  $\leftarrow maxStart$ 
25:          Schedule task with current operator
26:          scheduled  $\leftarrow true$ 
27:      end if
28:      if scheduledTask is not last of operatorSchedule & there is enough time
between current ScheduledTask and next ScheduledTask then // Caso in cui si
può inserire fra due interventi già schedulati
29:          task.OperationStart  $\leftarrow maxStart$ 
30:          Schedule task with current operator
31:          scheduled  $\leftarrow true$ 
32:      end if
33:  end for
34:  if scheduled == false then
35:      unscheduled  $\leftarrow add task$ 
36:  end if
37: end for
38: end for

```

5.3.2 Analisi della complessità

Analizzando l'algoritmo descritto nel capitolo precedente, definiamo:

- t il numero di tasks da programmare
- n il numero di operatori disponibili

Per studiare la complessità della soluzione possiamo procedere per step incrementalni:

1. per ordinare tutti i task troviamo un'istruzione di ordinamento la cui complessità è nota essere $\mathcal{O}(t \log t)$
2. un secondo ciclo inizializza la schedule di ogni operatore con un intervento; se $t < n$ si avrebbero solo t iterazioni e in tal caso il programma terminerebbe visto che tutti i task sono stati schedulati. Nel caso generale in cui $t \gg n$ saranno eseguite n iterazioni e quindi $\mathcal{O}(n)$
3. a questo punto, per ogni task non ancora assegnato nel ciclo precedente ($t - n \approx t$, $t \gg n$), bisognerà assegnare il task se possibile. In particolare si avranno:
 - per bilanciare l'assegnazione dei task giornalieri agli operatori l'ordine seguito per decidere l'operatore a cui assegnare l'intervento inizia da quello che ha meno interventi assegnatigli. Sono quindi riordinati gli operatori in base al numero di interventi che gli sono stati assegnati con una complessità nota $\mathcal{O}(n \log n)$
 - a questo punto si prova ad assegnare l'intervento agli operatori, seguendo l'ordine dettato dal punto precedente, e o si assegna al primo operatore a cui si può assegnare l'evento oppure, una volta controllati tutti gli operatori, si aggiunge l'intervento alla lista degli interventi non assegnabili! Studiamo quindi i due casi:
 - intervento viene assegnato: si è notato durante i test che la probabilità di collisione fra interventi decresce con l'aumento del numero di operatori disponibili e supponendo valido l'assunto per cui il numero di operatori è sufficiente per coprire tutti gli interventi da eseguire, possiamo considerare che ogni intervento processato sia assegnato direttamente al primo operatore testato, con un'alta probabilità. Il ciclo sarà quindi eseguito una sola volta e all'interno del ciclo si dovranno scandire, nel

caso peggiore, tutti gli eventi assegnati nei cicli precedenti. Si tratta di una serie aritmetica che porta ad una complessità di $\mathcal{O}(t^2)$. In queste condizioni quindi si crea un bilanciamento fra gli interventi assegnati agli operatori ed è come se si dovessero assegnare solo $\frac{t}{n}$ interventi ma essendo $t \gg n$ la complessità può essere considerata invariata.

- intervento non viene assegnato a nessun operatore: in tal caso si devono scandire tutti gli operatori che però si suppone siano comunque bilanciati per le osservazioni del punto precedente. Si arriva ad una complessità $\mathcal{O}(n * t^2)$

Il costo totale in media, in termini di tempo, dell'algoritmo risulta quindi essere $\mathcal{O}(t^2)$, dove si ricorda ancora che t è il numero degli interventi da programmare.

5.4 Diagramma dei componenti

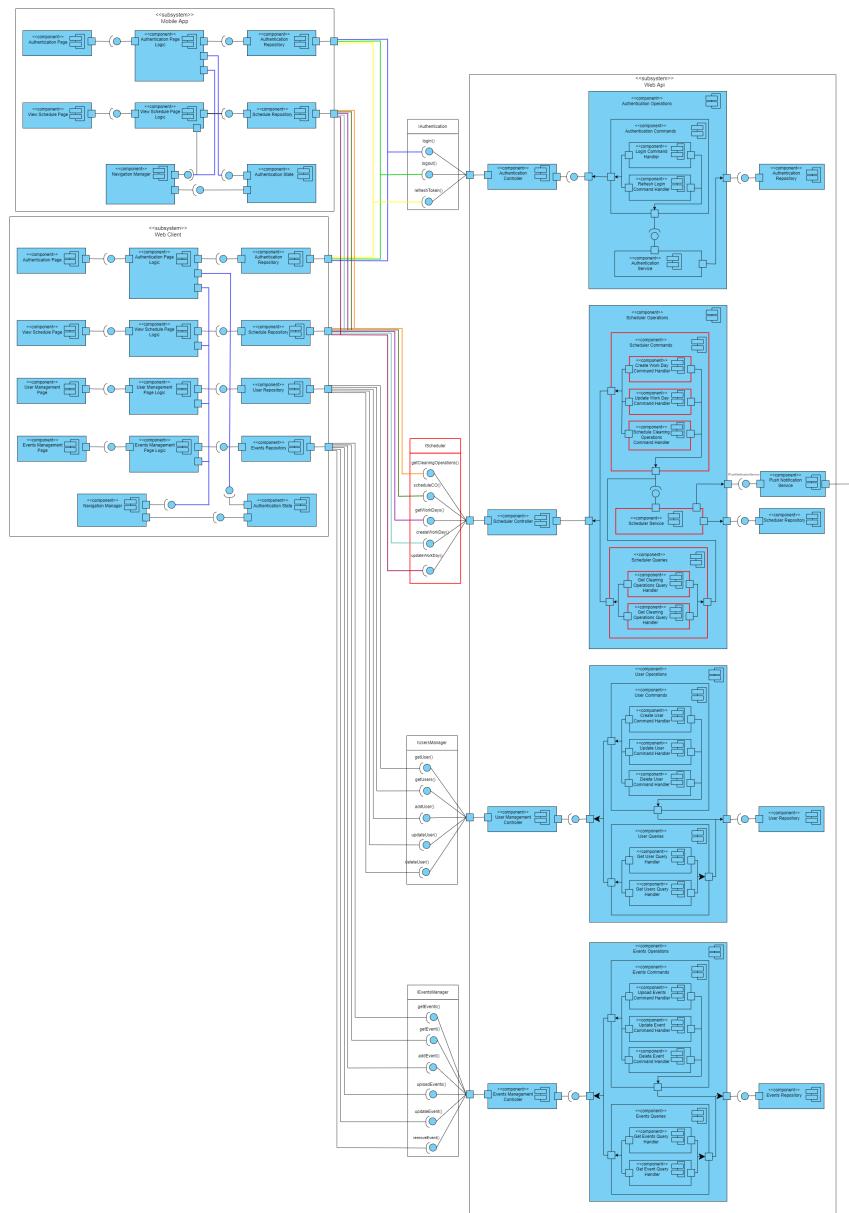


Figura 5.1: Diagramma dei componenti

5.5 Diagramma entità relazioni (ER)

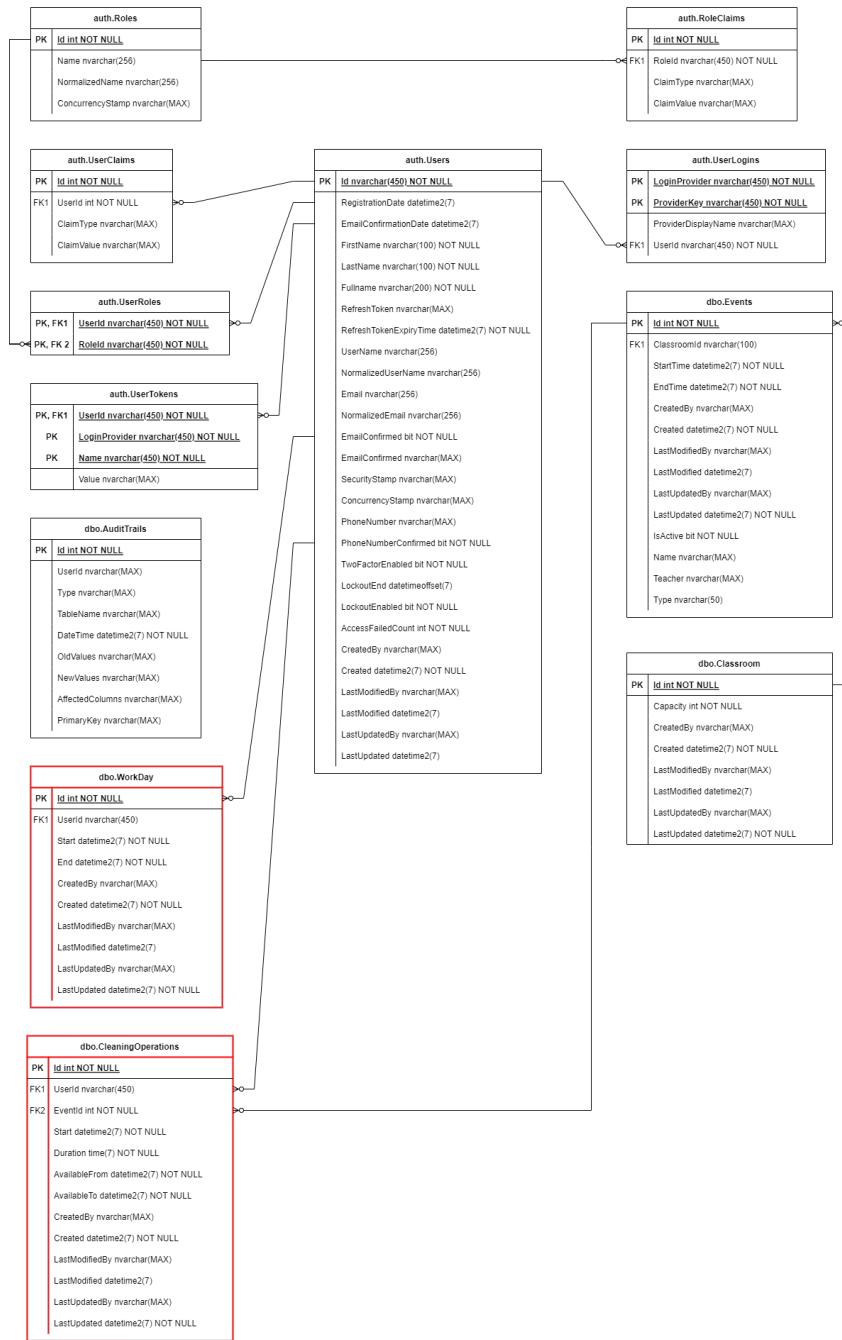


Figura 5.2: Diagramma entità relazioni

5.6 Diagramma delle classi



Figura 5.3: Diagramma delle classi (tipi di dati)

5.7 Diagramma delle interfacce

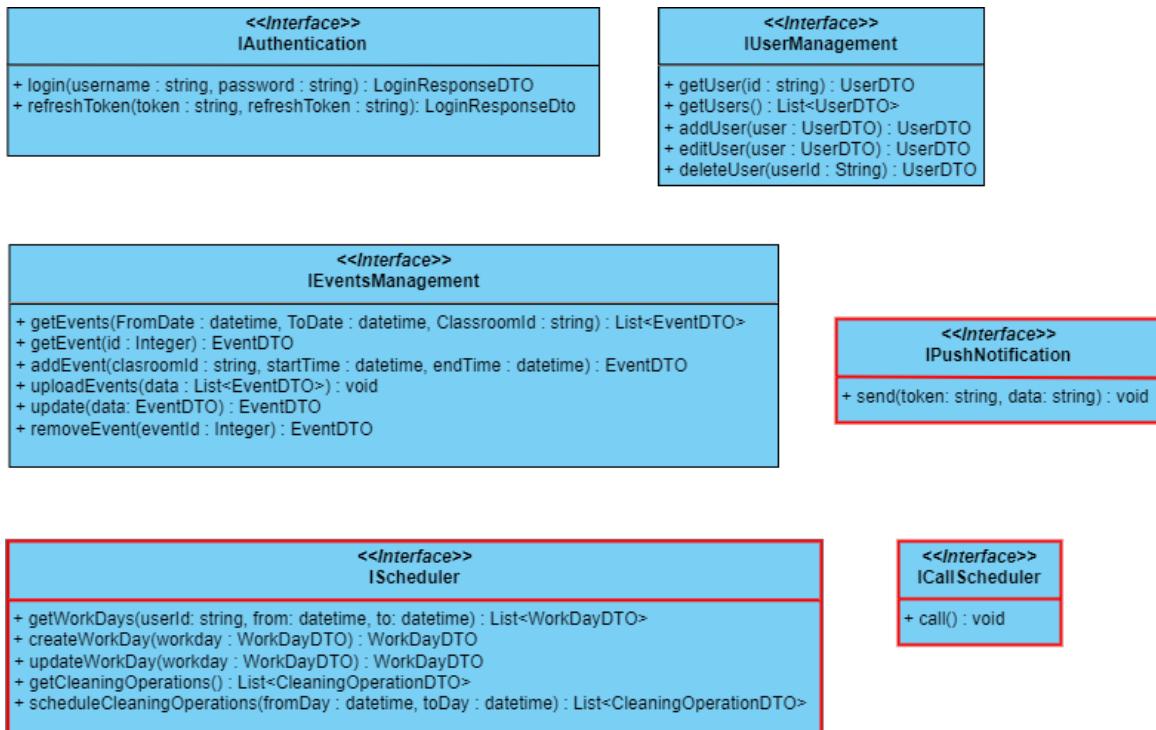


Figura 5.4: Diagramma delle classi (interfacce)

5.8 Diagramma dei package

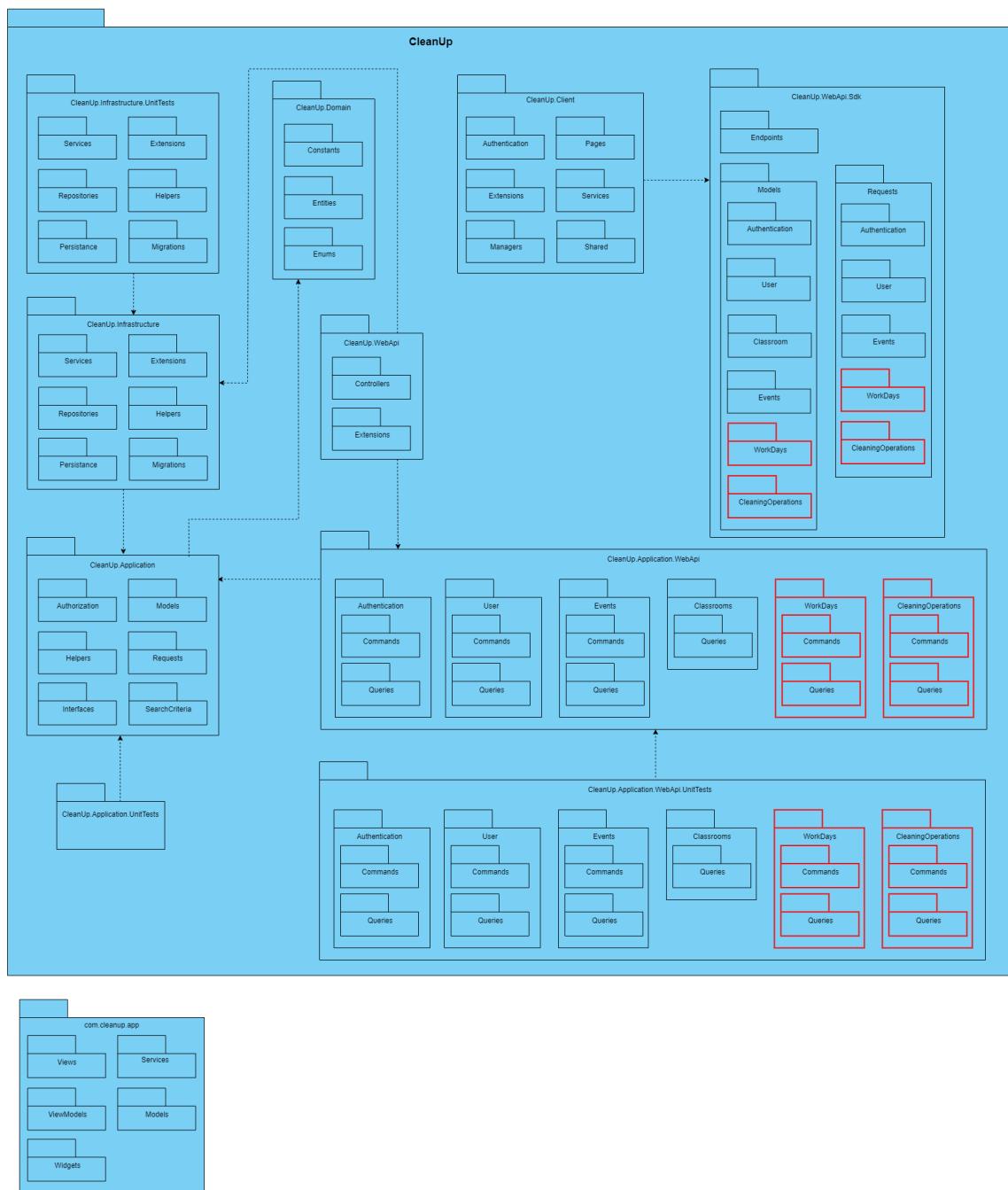


Figura 5.5: Diagramma dei package

5.9 Testing

5.9.1 Unit test

Per verificare il corretto funzionamento dell'algoritmo di scheduling, sono stati effettuati degli unit test del metodo Scheduler dello Scheduler Service.

▲ ✓ SortBasedSchedulerServiceTests (3)	179 ms
✓ Schedule_ShouldReturnCleaningOperations_WhenConflicts	< 1 ms
✓ Schedule_ShouldReturnCleaningOperations_WhenOperatorsAvailable	104 ms
✓ Schedule_ShouldReturnException_WhenOperatorsNotAvailable	75 ms

Figura 5.6: Unit test

5.9.2 Test delle API

GET GetSchedule https://localhost:7162/v1/scheduler?FromDate=2022-10-24T22:50:14.785Z&ToDate=2022-11-24T22:50:14.785Z	/ Scheduler / GetSchedule
Pass Status code is 200	
Pass Response JSON Schema is correct	

Figura 5.7: Test API Get Schedule

POST ComputeSchedule https://localhost:7162/v1/scheduler/schedule?Date=2022-10-24T22:50:14.785Z	/ Scheduler / ComputeSchedule
Pass Status code is 200	
Pass Response JSON Schema is correct	
Pass Empty ID	

Figura 5.8: Test API Compute Schedule

GET getWorkDays https://localhost:7162/v1/scheduler/work-day/f6da4dfe-7593-4557-8b86-cb8c1e707092?from=2022-10-24T22:50:14.785Z&to=2022-11-24T22:50:14.785Z	/ Scheduler / getWorkDays
Pass Status code is 200	
Pass Response JSON Schema is correct	

Figura 5.9: Test API Get Work Days

POST CreateWorkDay <https://localhost:7162/v1/scheduler/work-day> / Scheduler / CreateWorkDay

- Pass Status code is 200
- Pass Response JSON Schema is correct
- Pass Not existing ID

Figura 5.10: Test API Create Work Day

PUT UpdateWorkDay <https://localhost:7162/v1/scheduler/work-day/17> / Scheduler / UpdateWorkDay

- Pass Status code is 200
- Pass Response JSON Schema is correct
- Pass Not existing ID

Figura 5.11: Test API Update Work Day

5.10 Analisi statica del codice

Di seguito si riportano le metriche di qualità del codice fornite dallo strumento integrato di Visual Studio includendo le modifiche dell'iterazione 5

Gerarchia ▾	Indice di manutenib... Complessità ciclomatica Profondità dell'ereditarietà Accoppiamenti di classi ... Rigue di codice ... Rigue di codice ...
▷ src\CleanUp.Application (Debug)	94 105 4 24 289 49
▷ src\CleanUp.Application.WebApi (Debug)	89 259 2 95 1.364 299
▷ src\CleanUp.Domain (Debug)	98 69 4 15 103 2
▷ src\CleanUp.Infrastructure (Debug)	68 175 6 161 5.241 1.551
▷ src\Hangfire\CleanUp.Hangfire (Debug)	89 102 4 77 590 92
▷ src\Hangfire\CleanUp.Hangfire.Data (Debug)	77 4 1 46 86 19
▷ src\Sdk\CleanUp.WebApi.Sdk (Debug)	94 189 2 29 361 50
▷ src\WebCleanUp.Client (Debug)	90 1.141 3 192 9.542 674
▷ src\WebCleanUp.WebApi (Debug)	84 189 4 230 1.191 290
▷ test\CleanUp.Application.WebApi.UnitTests (Debug)	79 9 1 42 208 58
▷ test\CleanUp.Infrastructure.UnitTests (Debug)	79 10 1 51 304 90

Figura 5.12: Metriche di qualità del codice

5.11 Documentazione API

Scheduler

Iteration 5:

Scheduler call APIs.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection [API Test](#)

GET GetSchedule 🔒

`https://localhost:7162/v1/scheduler?FromDate=2022-10-24T22:50:14.785Z&ToDate=2022-11-24T22:50:14.785Z`

API that returns the schedule of the events that take place in the specified time frame.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

PARAMS

FromDate 2022-10-24T22:50:14.785Z

ToDate 2022-11-24T22:50:14.785Z

Figura 5.13: Get Schedule API

```

{
  "response": [
    {
      "id": 1,
      "eventId": 1,
      "userId": "277b0b13-b689-4dc2-95eb-56596f79c4f1",
      "start": "2022-11-23T10:30:00",
      "duration": "00:10:00",
      "availableFrom": "2022-11-23T10:30:00",
      "availableTo": "2022-11-23T11:00:00",
      "event": {
        "id": 1,
        "classroomId": "A203 [Edificio A - Dalmine]",
        "name": "Fisiologia generale exe A (stud. A-L)",
        "teacher": "Chiara Emma CAMPIGLIO",
        "startTime": "2022-11-23T08:30:00",
        "endTime": "2022-11-23T10:30:00",
        "type": "Lezione",
        "isActive": true
      },
      "user": {
        "id": "277b0b13-b689-4dc2-95eb-56596f79c4f1",
        "firstName": "Xhorxho",
        "lastName": "Papallazi",
        "fullName": "Papallazi Xhorxho",
        "email": "xhorxho.papallazi@gmail.com",
        "emailConfirmed": false,
        "userName": "xhorxho.papallazi@gmail.com",
        "phoneNumber": "1238965493",
        "emailConfirmationDate": "2022-10-25T00:57:42.6518645",
        "refreshTokenExpiryTime": "0001-01-01T00:00:00"
      }
    }
  ]
}

```

Figura 5.14: Risposta Get Schedule API

POST ComputeSchedule

<https://localhost:7162/v1/scheduler/schedule?Date=2022-10-24T22:50:14.785Z>

API that calls the scheduler in order to compute the schedule for the specified day.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

PARAMS

Date	2022-10-24T22:50:14.785Z
------	--------------------------

Figura 5.15: Compute Schedule API

```

{
  "isSuccess": true,
  "statusCode": "OK"
}

```

Figura 5.16: Risposta Compute Schedule API

GET getWorkDays 🔒

```
https://localhost:7162/v1/scheduler/work-day/f4da4dfe-7593-4557-8b86-cb8c1e707092?From=2022-10-24T22:50:14.785Z&To=2022-11-24T22:50:14.785Z
```

API that returns the user's working days in the time frame according to the specified data.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

PARAMS

From 2022-10-24T22:50:14.785Z

To 2022-11-24T22:50:14.785Z

Figura 5.17: Get Work Days API



```

{
  "response": [
    {
      "id": 1,
      "userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092",
      "start": "2022-11-01T08:00:00",
      "end": "2022-11-01T12:00:00"
    },
    {
      "id": 2,
      "userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092",
      "start": "2022-11-02T08:00:00",
      "end": "2022-11-02T12:00:00"
    }
  ]
}

```

Figura 5.18: Risposta Get Work Days API

POST CreateWorkDay 🔒

```
https://localhost:7162/v1/scheduler/work-day
```

API that creates a new working day for the user specified by his id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY raw

```
{
  "userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092",
  "start": "2022-10-01T08:00:00.000Z",
  "end": "2022-10-01T12:00:00.000Z"
}
```

Figura 5.19: Create Work Day API

```

  "response": {
    "id": 18,
    "userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092",
    "start": "2022-11-23T08:00:00Z",
    "end": "2022-11-23T12:00:00Z"
  },
  "isSuccess": true,
  "statusCode": "OK"

```

Figura 5.20: Risposta Create Work Day API

PUT UpdateWorkDay

<https://localhost:7162/v1/scheduler/work-day/2>

API that update the working day specified by its id.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection [API Test](#)

BODY raw

```
{
  "start": "2022-10-01T08:00:00.000Z",
  "end": "2022-10-01T12:00:00.000Z"
}
```

Figura 5.21: Update Work Day API

```

  "response": {
    "id": 18,
    "userId": "f4da4dfe-7593-4557-8b86-cb8c1e707092",
    "start": "2022-11-24T08:00:00Z",
    "end": "2022-11-24T12:00:00Z"
  },
  "isSuccess": true,
  "statusCode": "OK"

```

Figura 5.22: Risposta Update Work Day API

Iterazione 6

6.1 Descrizione

In questa iterazione si sono progettati e sviluppati i casi d'uso relativi al Report (UC3).

6.2 Casi d'uso

6.2.1 UC3 - Visualizzazione report

UC3	Visualizzazione report
<i>Descrizione:</i>	Visualizzazione dei report giornalieri delle attività
<i>Attori:</i>	Amministratore, operatore
<i>Precondizioni:</i>	L'utente è già autenticato
<i>Flusso eventi:</i>	<ol style="list-style-type: none">1. L'amministratore richiede il report di un giorno2. Generare il report relativo
<i>Postcondizioni:</i>	Report generato
<i>Eccezioni:</i>	In quel giorno non sono presenti eventi o attività

6.3 Diagramma dei componenti

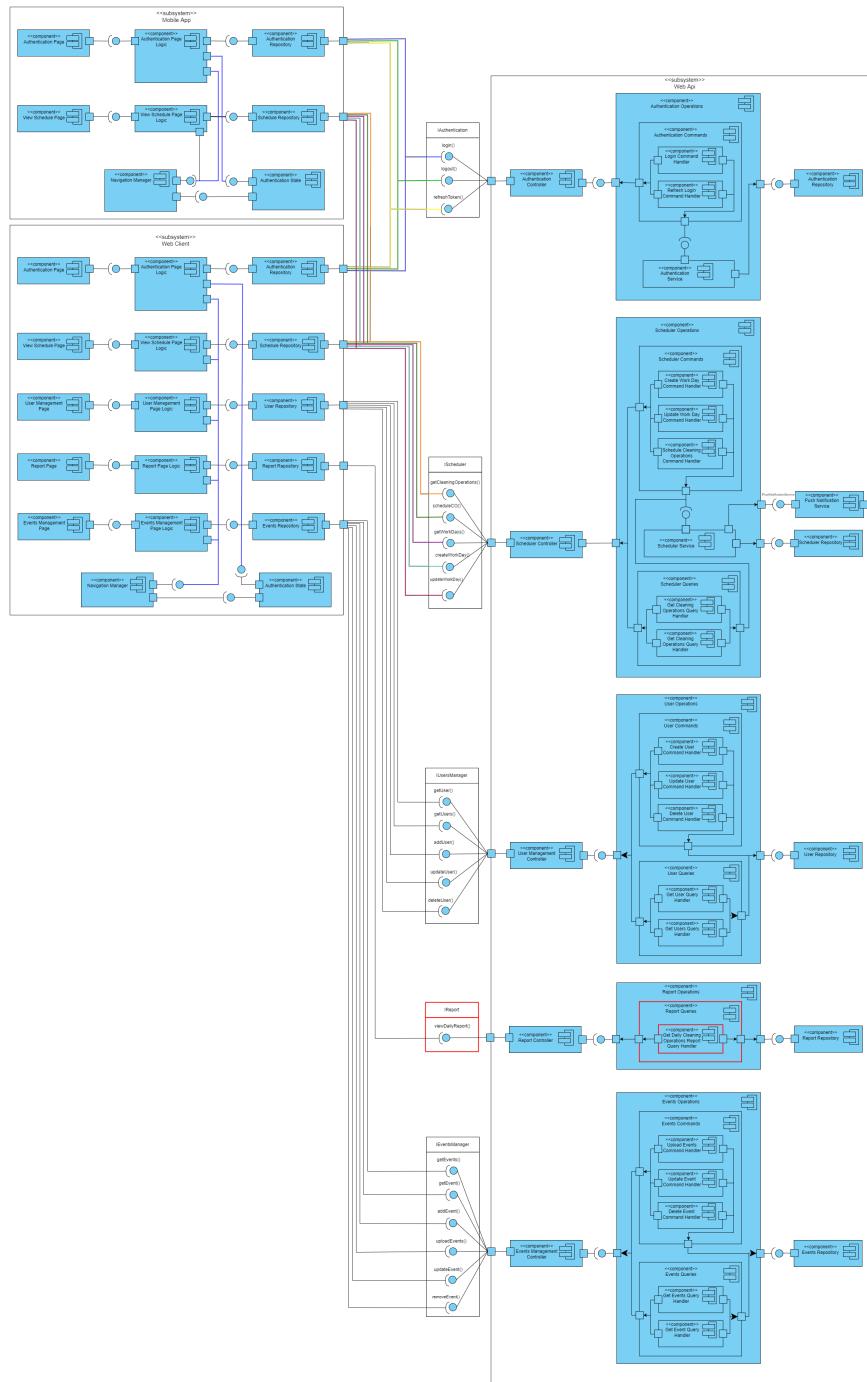


Figura 6.1: Diagramma dei componenti

6.4 Diagramma entità relazioni (ER)

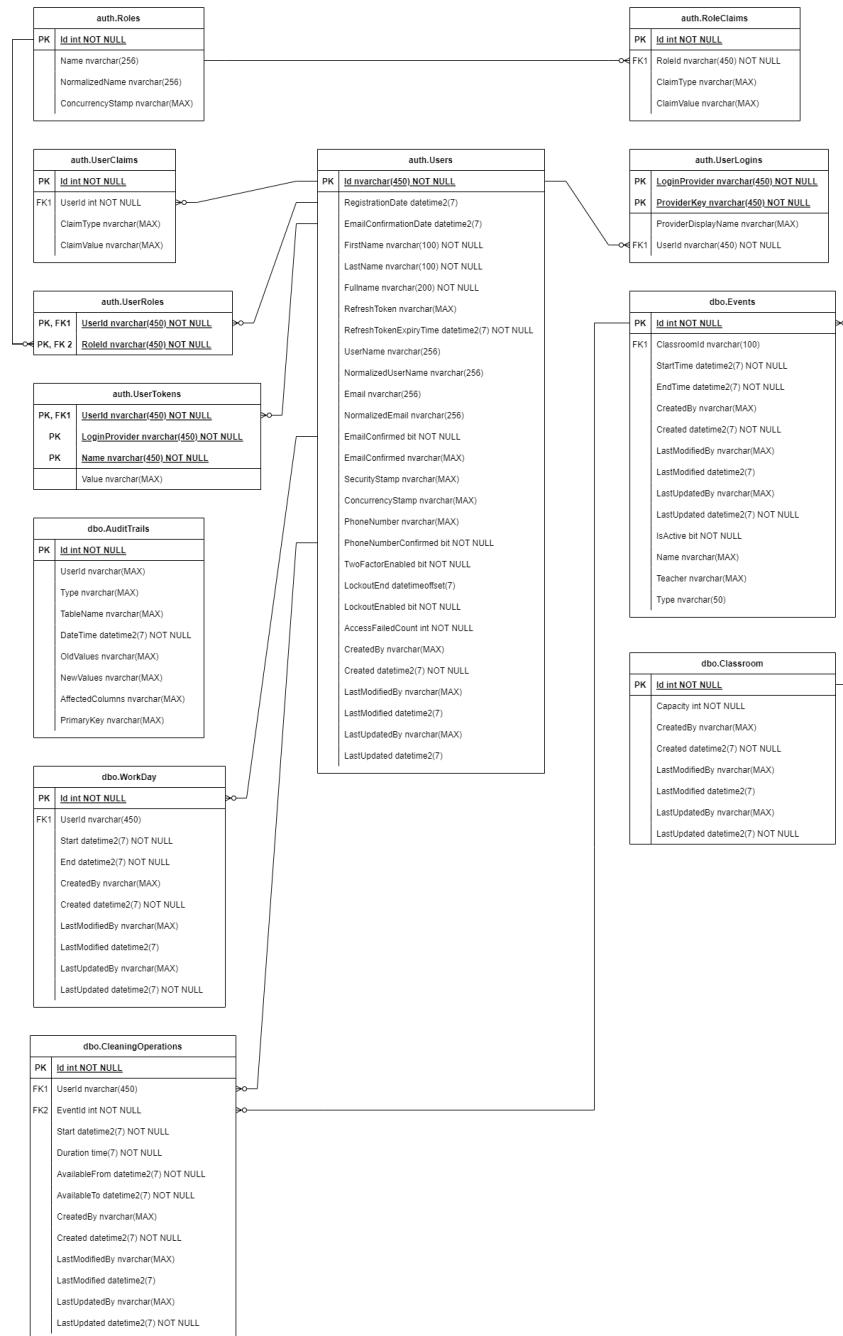


Figura 6.2: Diagramma entità relazioni

6.5 Diagramma delle classi

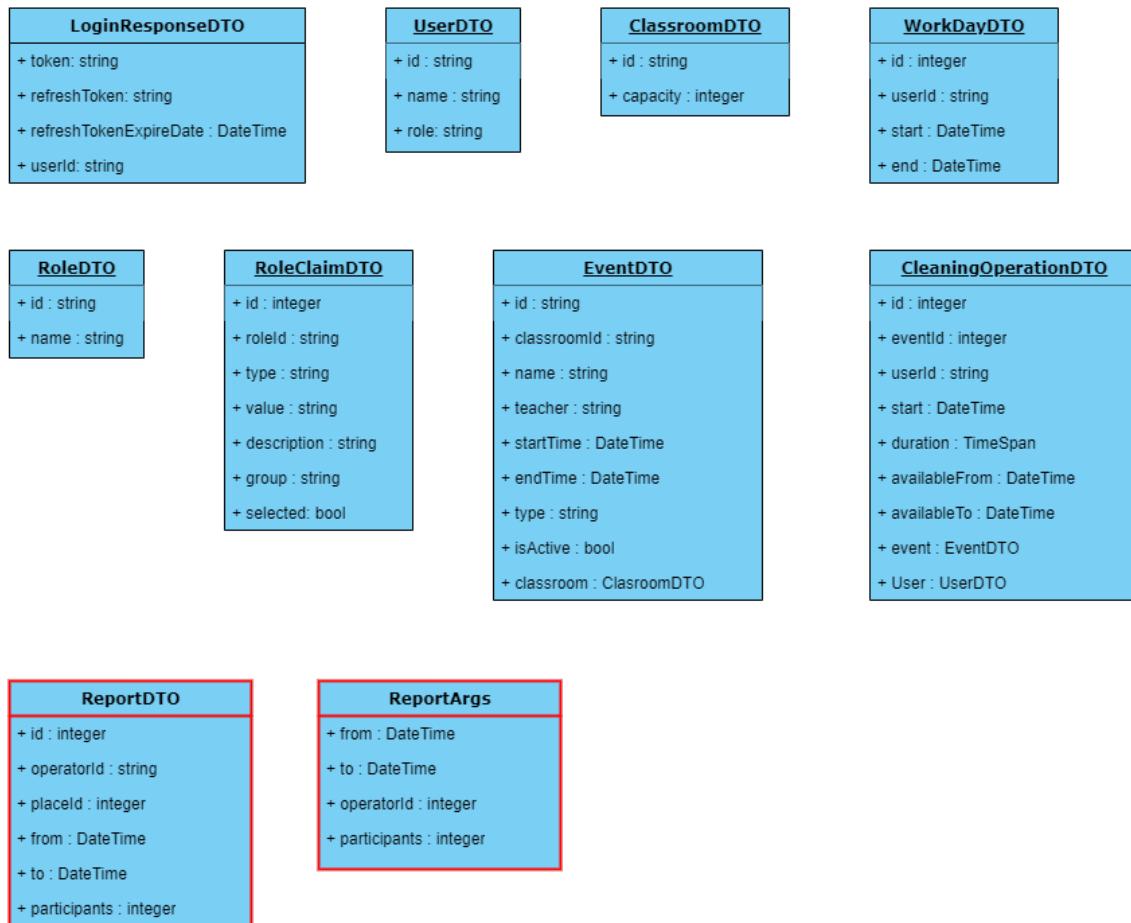


Figura 6.3: Diagramma delle classi (tipi di dati)

6.6 Diagramma delle interfacce

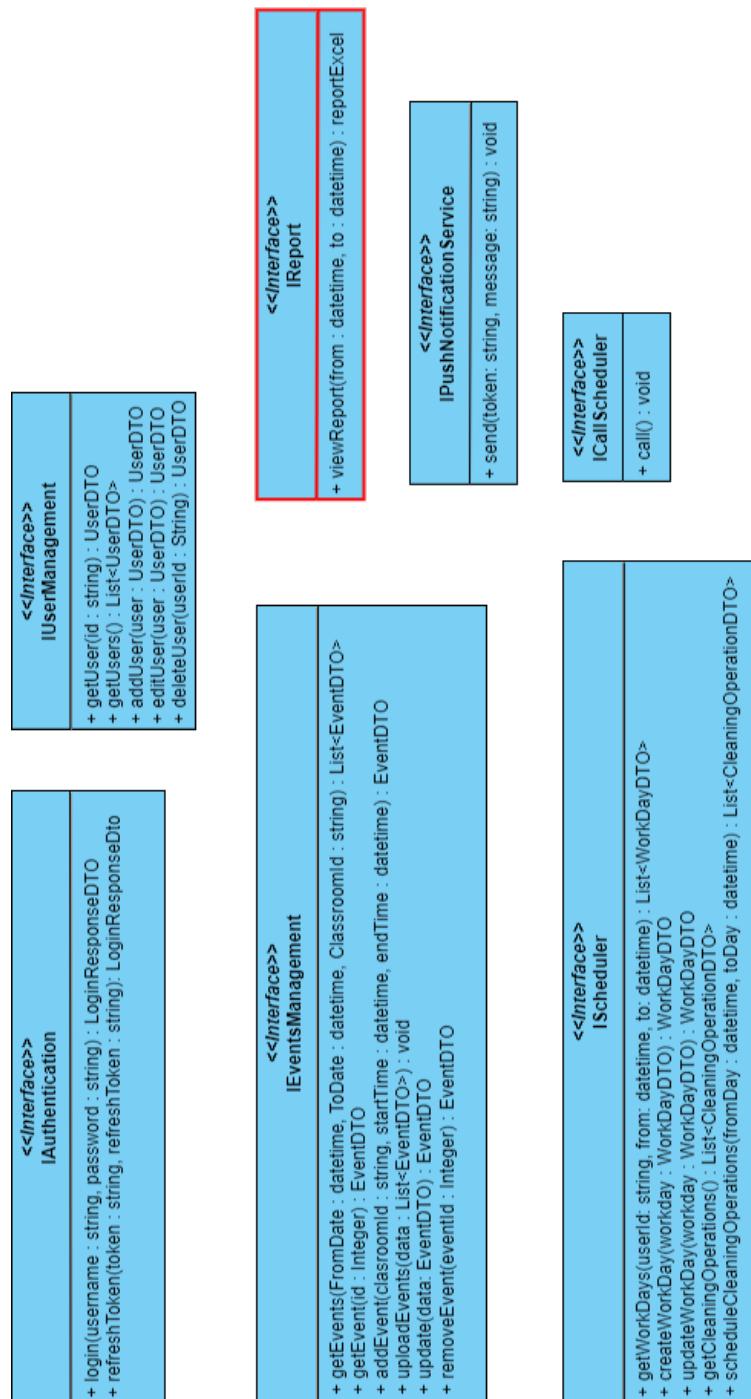


Figura 6.4: Diagramma delle classi (interfacce)

6.7 Diagramma dei package

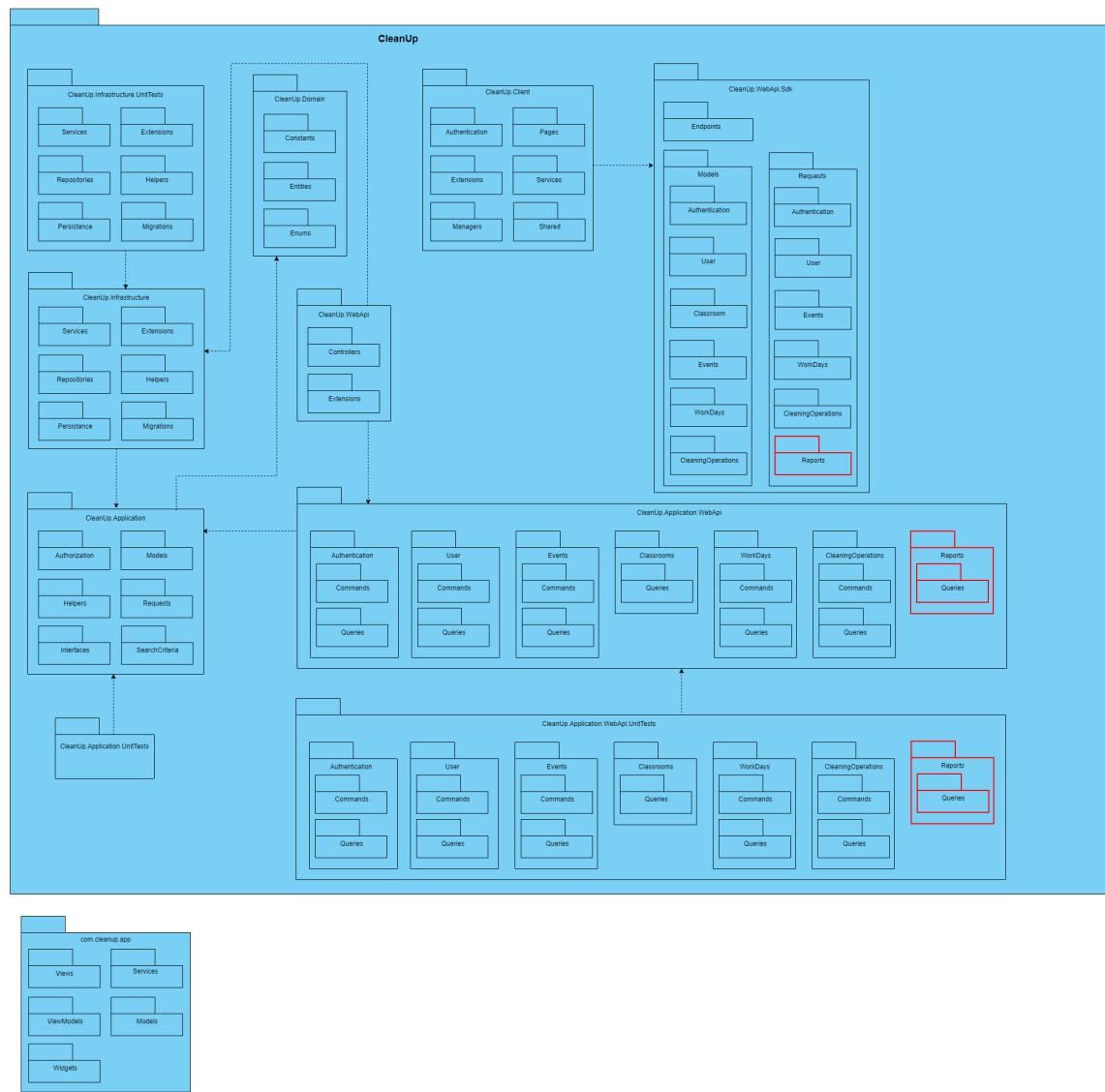


Figura 6.5: Diagramma dei package

6.8 Testing

6.8.1 Test delle API

GET getReport https://localhost:7162/v1/report/daily-cleaning-operations?fromDate=2022-10-24T22:50:14.785Z&toDate=2022-11-24T22:50:14.785Z		/ Report / getReport
Pass	Status code is 200	
Pass	Body matches string	

Figura 6.6: Test API Get Report

6.9 Analisi statica del codice

Di seguito si riportano le metriche di qualità del codice fornite dallo strumento integrato di Visual Studio includendo le modifiche dell'iterazione 6.

Gerarchia	Indice di manutenibilità	Complessità ciclomatica	Profondità dell'ereditarietà	Accoppiamenti di classi	Righe di codice ...	Righe di codice ...
src\CleanUp.Application (Debug)	93	116	4	25	338	63
src\CleanUp.Application.WebApi (Debug)	88	279	2	104	1.535	345
src\CleanUp.Domain (Debug)	98	69	4	15	103	2
src\CleanUp.Infrastructure (Debug)	69	180	6	162	5.322	1.557
src\Hangfire\CleanUp.Hangfire (Debug)	86	102	4	83	668	139
src\Hangfire\CleanUp.Hangfire.Data (Debug)	77	4	1	46	86	19
src\Sdk\CleanUp.WebApi.Sdk (Debug)	94	227	2	29	409	57
src\Web\CleanUp.Client (Debug)	84	1.215	3	232	15.051	1.789
src\Web\CleanUp.WebApi (Debug)	82	191	4	243	1.302	345
test\CleanUp.Application.WebApi.UnitTests (Debug)	79	9	1	42	208	58
test\CleanUp.Infrastructure.UnitTests (Debug)	75	11	1	55	365	108

Figura 6.7: Metriche di qualità del codice

6.10 Documentazione API

Report

Iteration 6:

Report download API.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection [API Test](#)

GET `getReport` 

<https://localhost:7162/v1/report/daily-cleaning-operations?fromDate=2022-10-24T22:50:14.785Z&toDate=2022-11-24T22:50:14.785Z>

API that returns an excel file containing the report of the scheduled interventions in the specified time frame.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection API Test

PARAMS

```
fromDate      2022-10-24T22:50:14.785Z  
toDate        2022-11-24T22:50:14.785Z
```

Figura 6.8: Get Report API

Figura 6.9: Risposta Get Report API (file .xlsx)

Manuale utente

Per quanto riguarda l'installazione delle api e del client web pur considerando le tecnologie utilizzate e nonostante .NET sia un framework di proprietà di Microsoft, il progetto è multipiattaforma, consentendo l'esecuzione su Windows, OS X e su diverse distribuzioni Linux.

Per quanto riguarda l'applicazione per smartphone, il framework utilizzato permette la portabilità sia su Android che iOS.

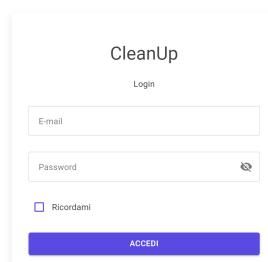


Figura 7.1: Schermata login

Name	Surname	User name	Email	Phone number	Role	Confirmation status	Actions
Pippo	Verdi	pippoverdi@gmail.com	pippoverdi@gmail.com		Operator	<input checked="" type="checkbox"/>	AZIONI
Mario	Rossi	mariorossi@gmail.com	mariorossi@gmail.com		Operator	<input checked="" type="checkbox"/>	AZIONI
Pinco	Pallino	pincopallino@gmail.com	pincopallino@gmail.com		Operator	<input checked="" type="checkbox"/>	AZIONI
Paolo	Bianchi	paoletobianchi@gmail.com	paoletobianchi@gmail.com		Operator	<input checked="" type="checkbox"/>	AZIONI
Simone	Ronconi	sronzoni99@gmail.com	sronzoni99@gmail.com		Administrator	<input checked="" type="checkbox"/>	AZIONI

Figura 7.2: Schermata utente

Name	Room	Teacher	Start date	End date	Type	Status	Actions
QUALITY IMPROVEMENT IN HEALTHCARE	Lab. Galvani - piano1 [Laboratori - Dalmine]	Mariangela Quarto	26/09/2022 13:30:00	26/09/2022 15:30:00	<input checked="" type="checkbox"/>	AZIONI	
Virtual and Physical Prototyping	B103 [Edificio B - Dalmine]	Daniele Regazzoni, Daniele LANDI	26/09/2022 14:00:00	26/09/2022 16:00:00	<input checked="" type="checkbox"/>	AZIONI	
TERMOFLUIDODINAMICA	B104 [Edificio B - Dalmine]	Gianpietro Cossali, Lorenzo Alessio Botti	26/09/2022 14:00:00	26/09/2022 16:00:00	<input checked="" type="checkbox"/>	AZIONI	
DISPOSITIVI MEDICALI E DIAGNOSTICI	A101 [Edificio A - Dalmine]	Andrea Remuzzi, Chiara Emma CAMPIGLIO	26/09/2022 14:00:00	26/09/2022 16:00:00	<input checked="" type="checkbox"/>	AZIONI	
STATICA E FONDAMENTI DI SCIENZA DELLE COSTRUZIONI	B-106 [Edificio B - Dalmine]	Aram Comeggia	26/09/2022 14:00:00	26/09/2022 17:00:00	<input checked="" type="checkbox"/>	AZIONI	
BIOMATERIALI	A203 [Edificio A - Dalmine]	Marina Cabrini, Denny Coffetti, Alessandro CARROZZA	26/09/2022 14:00:00	26/09/2022 17:00:00	<input checked="" type="checkbox"/>	AZIONI	
ANALISI MATEMATICA II	A204 [Edificio A - Dalmine]	Giacomo Gigante	26/09/2022 14:00:00	26/09/2022 17:00:00	<input checked="" type="checkbox"/>	AZIONI	
COSTRUZIONI IN ACCIAIO	B005 [Edificio B - Dalmine]	Andrea Belleri	26/09/2022 14:00:00	26/09/2022 17:00:00	<input checked="" type="checkbox"/>	AZIONI	
MODELLI E ALGORITMI DI OTTIMIZZAZIONE	A104 Lab [Edificio A - Dalmine]	Maria Teresa Vespucci	26/09/2022 14:00:00	26/09/2022 17:00:00	<input checked="" type="checkbox"/>	AZIONI	
MACCHINE A FLUIDO	A002 [Edificio A - Dalmine]	Giovanna Barigozzi, Nicoletta Franchina, Giovanni Brumana	26/09/2022 14:00:00	26/09/2022 17:00:00	<input type="checkbox"/>	AZIONI	

Figura 7.3: Schermata eventi

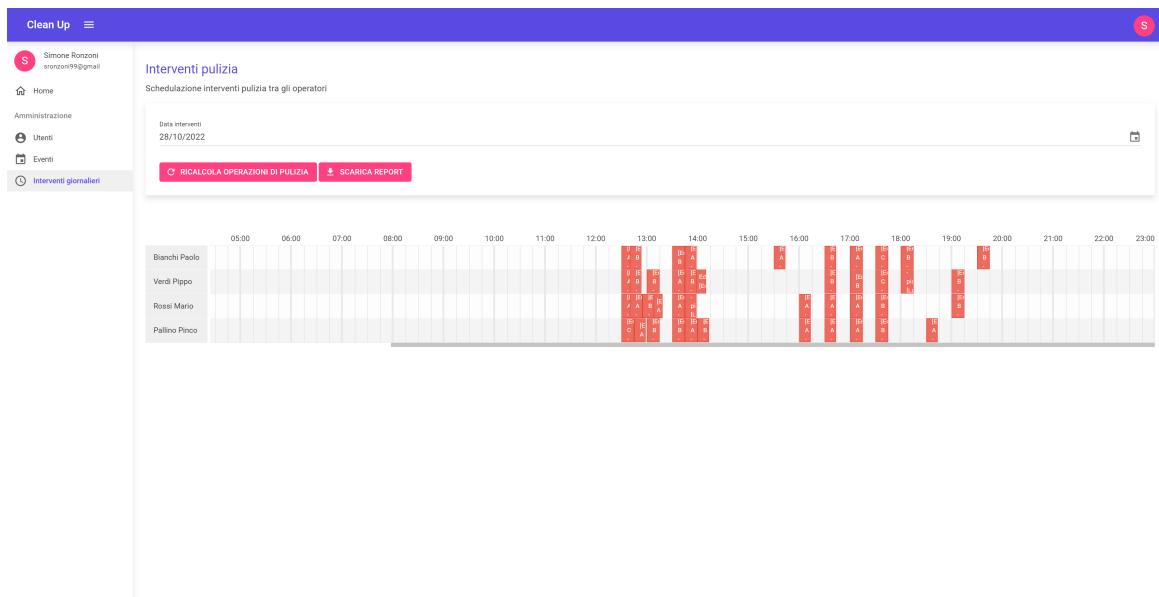


Figura 7.4: Schermata interventi

Conclusioni

8.1 Sviluppi futuri

Non tutto il progetto è stato implementato.

Le web api e il web client sono ultimanti e correttamente funzionanti, mentre la parte relativa all'applicazione mobile è da sviluppare interamente.

Per quanto riguarda l'algoritmo di scheduling, per ora non vengono gestiti gli orari di lavoro degli operatori ma si considera un operatore disponibile per tutta le giornata.