

Documentazione API

Sistema di Gestione Spazi di Coworking

Tettamanti Andrea

Mascetti Luca

Musetti Gregorio

Vernavà Lorenzo

14 settembre 2025

Sommario

Questa documentazione descrive in dettaglio le API REST del sistema CoWorkSpace, una piattaforma per la gestione di spazi di coworking. Il documento include le specifiche degli endpoint, gli schemi dei dati, i metodi di autenticazione e gli esempi di utilizzo. L'API segue i principi REST e utilizza JSON per lo scambio dei dati, implementando un sistema completo di autenticazione JWT e autorizzazione basata sui ruoli.

Indice

1	Panoramica del Sistema API	7
1.1	Architettura Generale	7
1.2	Architettura REST	7
1.3	URL Base e Versioning	8
1.4	Formato delle Risposte	8
1.4.1	Risposta di Successo	8
1.4.2	Risposta di Errore	8
1.5	Codici di Stato HTTP	8
1.6	Rate Limiting	9
1.7	CORS e Sicurezza	10
1.7.1	Sviluppo	10
1.7.2	Produzione	10
1.8	Middleware di Sicurezza	10
1.9	Gestione degli Errori	10
1.10	Gestione Transazioni e Rollback	11
1.10.1	Operazioni Transazionali	11
1.10.2	Pattern di Implementazione	11
1.10.3	Strategie di Rollback	11
1.10.4	Esempio Pratico: Pagamento con Aggiornamento Prenotazione . . .	12
1.11	Paginazione	13
1.12	Content-Type e Accept Headers	13
1.13	Documentazione Interattiva	13
1.14	Health Check e Monitoring	13
2	Autenticazione e Autorizzazione	14
2.1	Sistema di Autenticazione JWT	14
2.1.1	Processo di Login	14
2.1.2	Struttura del JWT Token	15
2.2	Utilizzo del Token	15
2.3	Sistema di Autorizzazione Basato sui Ruoli	16
2.3.1	Ruoli Disponibili	16
2.3.2	Matrice dei Permessi	16
2.4	Middleware di Autenticazione	16
2.4.1	authMiddleware.protect	16
2.4.2	authMiddleware.authorize	17
2.5	Gestione delle Password	17
2.5.1	Hashing delle Password	17
2.5.2	Reset Password	17
2.6	Rate Limiting per Sicurezza	18
2.7	Gestione Token FCM	18
2.8	Logout e Invalidazione Token	18
2.9	Sicurezza delle Sessioni	18
2.10	Validazione Input	19
2.11	Gestione Errori di Autenticazione	19

3	Endpoints API	19
3.1	Panoramica degli Endpoints	19
3.2	Endpoint Users	20
3.2.1	POST /api/users/register	20
3.2.2	POST /api/users/login	20
3.2.3	GET /api/users/profile	21
3.2.4	GET /api/users/dashboard	21
3.2.5	PUT /api/users/profile	21
3.2.6	PUT /api/users/change-password	21
3.2.7	POST /api/users/request-password-reset	21
3.2.8	POST /api/users/logout	21
3.2.9	POST /api/users/fcm-token	21
3.2.10	GET /api/users/check-email	21
3.3	Endpoint Locations	21
3.3.1	GET /api/locations	21
3.3.2	GET /api/locations/:id	22
3.3.3	GET /api/locations/:id/spaces	22
3.3.4	POST /api/locations	22
3.3.5	PUT /api/locations/:id	22
3.3.6	DELETE /api/locations/:id	22
3.4	Endpoint Spaces	22
3.4.1	GET /api/spaces	22
3.4.2	GET /api/spaces/:id	23
3.4.3	GET /api/spaces/:id/availability	23
3.4.4	POST /api/spaces	23
3.4.5	PUT /api/spaces/:id	23
3.4.6	DELETE /api/spaces/:id	23
3.5	Endpoint Space Types	23
3.5.1	GET /api/space-types	23
3.5.2	GET /api/space-types/:id	23
3.5.3	POST /api/space-types	23
3.5.4	PUT /api/space-types/:id	24
3.5.5	DELETE /api/space-types/:id	24
3.6	Endpoint Bookings	24
3.6.1	POST /api/bookings/check-availability	24
3.6.2	POST /api/bookings	24
3.6.3	GET /api/bookings	25
3.6.4	GET /api/bookings/:id	25
3.6.5	PUT /api/bookings/:id	25
3.6.6	DELETE /api/bookings/:id	25
3.6.7	PUT /api/bookings/:id/confirm	25
3.6.8	PUT /api/bookings/:id/complete	25
3.7	Endpoint Availability	25
3.7.1	GET /api/availability/space/:spaceId	25
3.7.2	POST /api/availability	26
3.7.3	PUT /api/availability/:id	26
3.7.4	DELETE /api/availability/:id	26
3.8	Endpoint Payments	26

3.8.1	GET /api/payments	26
3.8.2	GET /api/payments/:id	26
3.8.3	POST /api/payments	26
3.8.4	PUT /api/payments/:id/refund	26
3.9	Endpoint Notifications	27
3.9.1	GET /api/notifications	27
3.9.2	GET /api/notifications/:id	27
3.9.3	PUT /api/notifications/:id/read	27
3.9.4	POST /api/notifications/test	27
3.10	Endpoint Manager	27
3.10.1	GET /api/manager/dashboard	27
3.10.2	GET /api/manager/location	27
3.10.3	GET /api/manager/bookings	27
3.10.4	GET /api/manager/spaces	27
3.10.5	PUT /api/manager/spaces/:id	27
3.11	Endpoint Admin	28
3.11.1	GET /api/admin/dashboard	28
3.11.2	GET /api/admin/users	28
3.11.3	GET /api/admin/users/pending-managers	28
3.11.4	PUT /api/admin/users/:id/promote-manager	28
3.11.5	PUT /api/admin/users/:id/demote	28
3.11.6	GET /api/admin/bookings	28
3.11.7	GET /api/admin/stats	28
3.11.8	POST /api/admin/locations	28
3.11.9	PUT /api/admin/locations/:id/assign-manager	28
3.12	Endpoint di Sistema	29
3.12.1	GET /api/health	29
3.12.2	GET /api/	29
3.12.3	GET /api-docs	29
3.13	Convenzioni degli Endpoint	29
4	Schemi Dati e Validazioni	30
4.1	Panoramica degli Schemi	30
4.2	Schema User	30
4.3	Schema Location	31
4.4	Schema Space Type	31
4.5	Schema Space	32
4.6	Schema Booking	33
4.7	Schema Payment	34
4.8	Schema Notification	34
4.9	Schema Availability	36
4.10	Validazioni Globali del Sistema	36
4.10.1	Validazioni Date	36
4.10.2	Validazioni Email	36
4.10.3	Validazioni Password	37
4.10.4	Validazioni Numeriche	37
4.11	Gestione Errori di Validazione	37

5	Esempi di Utilizzo e Casi d'Uso	38
5.1	Panoramica dei Casi d'Uso	38
5.2	Caso d'Uso 1: Registrazione e Login Utente	38
5.2.1	Passo 1: Verifica Email Disponibilità	38
5.2.2	Passo 2: Registrazione Nuovo Utente	38
5.2.3	Passo 3: Login Utente	39
5.3	Caso d'Uso 2: Ricerca e Prenotazione Spazio	40
5.3.1	Passo 1: Ricerca Spazi con Filtri	40
5.3.2	Passo 2: Verifica Disponibilità Specifica	41
5.3.3	Passo 3: Creazione Prenotazione	41
5.4	Caso d'Uso 3: Gestione Pagamento	42
5.4.1	Passo 1: Creazione Pagamento	42
5.5	Caso d'Uso 4: Dashboard Utente	43
5.6	Caso d'Uso 5: Workflow Manager	44
5.6.1	Passo 1: Dashboard Manager	44
5.6.2	Passo 2: Visualizza Prenotazioni Location	45
5.6.3	Passo 3: Conferma Prenotazione	45
5.7	Caso d'Uso 6: Gestione Notifiche	45
5.7.1	Salvataggio Token FCM	45
5.7.2	Recupero Notifiche	45
5.8	Caso d'Uso 7: Workflow Admin	45
5.8.1	Dashboard Amministrativa	45
5.8.2	Promozione Manager	46
5.9	Gestione Errori Comuni	46
5.9.1	Errore Autenticazione	46
5.9.2	Errore Disponibilità	46
5.10	Best Practices per Integrazione	47
5.10.1	Autenticazione	47
5.10.2	Chiamate API	47
5.10.3	Gestione Errori	47
5.11	Esempi SDK/Client	47
5.11.1	JavaScript/TypeScript Client	47
5.12	Testing API	48
5.12.1	Test con curl	48
5.12.2	Test con Postman	48

Panoramica del Sistema API

Architettura Generale

L'architettura generale dell'applicazione segue il seguente schema:

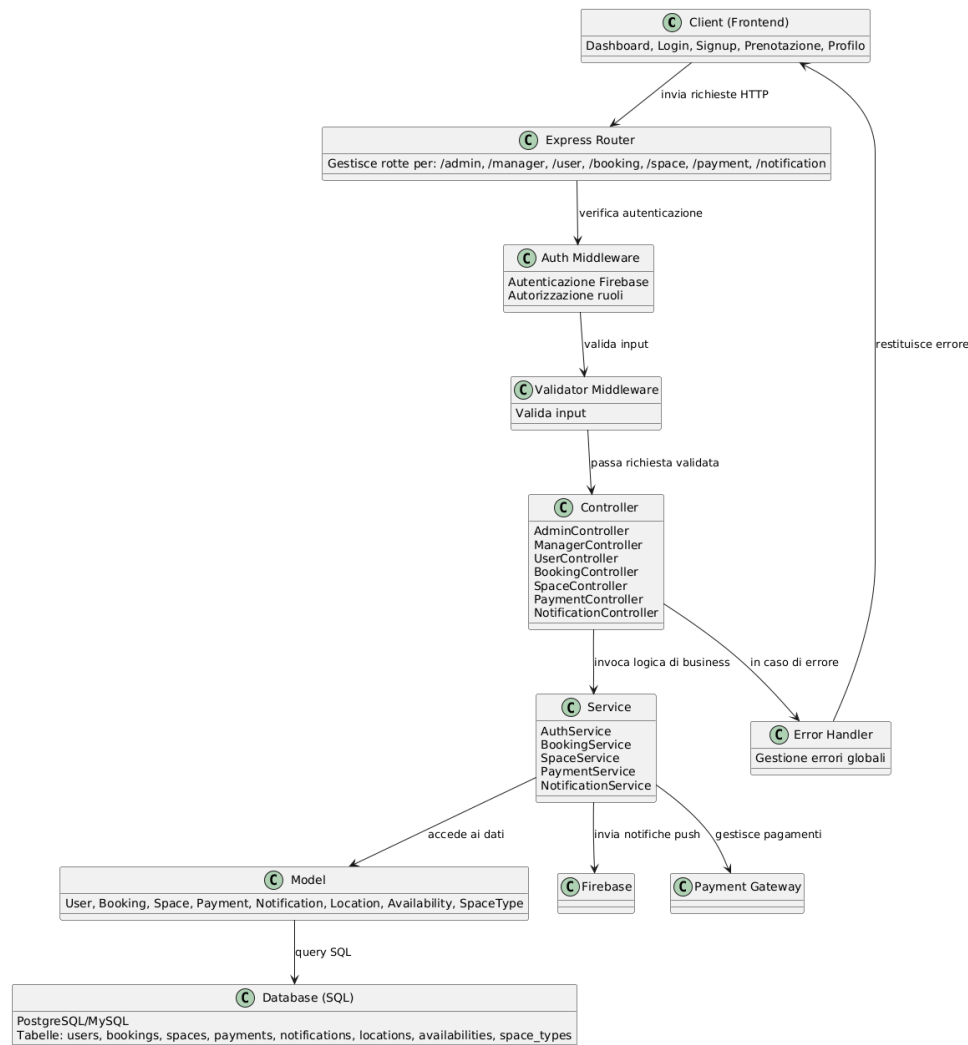


Figura 1: Architettura Generale del Sistema CoWorkSpace

Architettura REST

L'API CoWorkSpace implementa un'architettura REST (Representational State Transfer) che segue i principi di design moderni per sistemi distribuiti. Le caratteristiche principali includono:

- **Stateless:** Ogni richiesta è indipendente e contiene tutte le informazioni necessarie
- **Resource-oriented:** Gli endpoint sono organizzati attorno alle risorse del sistema
- **HTTP Methods:** Utilizzo appropriato dei verbi HTTP (GET, POST, PUT, DELETE)
- **JSON Format:** Tutti i dati sono scambiati in formato JSON

- **Consistent Response:** Struttura uniforme delle risposte API

URL Base e Versioning

```
1 # Development
2 https://localhost:3000/api
3
4 # Production
5 https://api.coworkspace.com/api
```

Listing 1: URL Base dell'API

L'API utilizza un approccio basato su path per l'organizzazione degli endpoint senza versioning esplicito nell'URL, seguendo il principio di evoluzione backward-compatible.

Formato delle Risposte

Tutte le risposte dell'API seguono una struttura uniforme per garantire consistenza e facilità di parsing:

1.4.1 Risposta di Successo

```
1 {
2   "success": true,
3   "message": "Operazione completata con successo",
4   "data": {
5     // Contenuto specifico della risposta
6   }
7 }
```

Listing 2: Struttura Risposta di Successo

1.4.2 Risposta di Errore

```
1 {
2   "success": false,
3   "message": "Descrizione dell'errore per l'utente",
4   "error": "Dettagli tecnici dell'errore",
5   "errors": [
6     {
7       "field": "email",
8       "message": "Email non valida"
9     }
10  ]
11 }
```

Listing 3: Struttura Risposta di Errore

Codici di Stato HTTP

L'API utilizza i codici di stato HTTP standard per comunicare l'esito delle operazioni:

Codice	Significato	Utilizzo nell'API
2xx - Successo		
200	OK	Operazione completata con successo
201	Created	Risorsa creata (registrazione, prenotazione)
202	Accepted	Richiesta accettata ma in elaborazione
204	No Content	Operazione completata senza contenuto
4xx - Errori Client		
400	Bad Request	Dati di input non validi
401	Unauthorized	Mancanza di autenticazione
403	Forbidden	Mancanza di autorizzazione
404	Not Found	Risorsa non trovata
409	Conflict	Conflitto con stato corrente (email già esistente)
422	Unprocessable Entity	Validazione business logic fallita
429	Too Many Requests	Rate limiting superato
5xx - Errori Server		
500	Internal Server Error	Errore interno del server
503	Service Unavailable	Servizio temporaneamente non disponibile

Tabella 1: Codici di Stato HTTP Utilizzati

Rate Limiting

Il sistema implementa rate limiting per proteggere l'API da abusi e garantire qualità del servizio:

Endpoint	Limite	Finestra Temporale
Authentication	5 tentativi	15 minuti
Password Reset	5 tentativi	15 minuti

Tabella 2: Configurazione Rate Limiting

Quando il limite viene superato, l'API restituisce un codice 429 con header informativi:

```

1 HTTP/1.1 429 Too Many Requests
2 X-RateLimit-Limit: 5
3 X-RateLimit-Remaining: 0
4 X-RateLimit-Reset: 1640995200
5 Retry-After: 900
6
7 {
8   "success": false,
9   "message": "Troppi tentativi. Riprova tra 15 minuti."
10 }
```

Listing 4: Response Headers Rate Limiting

CORS e Sicurezza

L'API implementa politiche CORS (Cross-Origin Resource Sharing) differenziate per ambiente:

1.7.1 Sviluppo

- Origini permesse: `localhost:3000`, `127.0.0.1:5500`
- Metodi: GET, POST, PUT, DELETE, PATCH
- Headers: Content-Type, Authorization
- Credentials: Supportate

1.7.2 Produzione

- Origini: Solo quelle specificate in `FRONTEND_URL`
- Politiche più restrittive per sicurezza
- Helmet.js per header di sicurezza

Middleware di Sicurezza

Il sistema utilizza diversi middleware per garantire la sicurezza:

Middleware	Funzione
Helmet	Content Security Policy, prevenzione attacchi XSS
CORS	Controllo accessi cross-origin
Rate Limiting	Prevenzione attacchi DoS e brute force
Input Validation	Validazione e sanitizzazione input utente
JWT Authentication	Verifica token di autenticazione
Role-based Authorization	Controllo permessi basato sui ruoli

Tabella 3: Middleware di Sicurezza Implementati

Gestione degli Errori

Il sistema implementa una gestione degli errori centralizzata con logging strutturato:

```
1 {  
2   "success": false,  
3   "message": "Dati non validi",  
4   "errors": [  
5     {  
6       "type": "field",  
7       "field": "email",  
8       "message": "Email deve essere un indirizzo valido",  
9       "value": "email-non-valida"  
10    },  
11    {  
12      "type": "field",
```

```
13     "field": "password",
14     "message": "Password deve contenere almeno 8 caratteri",
15     "value": "***"
16   }
17 ]
18 }
```

Listing 5: Esempio Gestione Errore di Validazione

Gestione Transazioni e Rollback

Il sistema implementa transazioni database per garantire l'integrità dei dati nelle operazioni complesse:

1.10.1 Operazioni Transazionali

Le seguenti operazioni utilizzano transazioni atomiche:

- **Elaborazione Pagamenti:** Creazione pagamento + aggiornamento stato prenotazione
- **Gestione Disponibilità:** Aggiornamento multiplo slot temporali
- **Operazioni SpaceType:** Modifica tipi spazio con validazioni incrociate
- **Prenotazioni Complesse:** Creazione prenotazione + blocco disponibilità + notifiche

1.10.2 Pattern di Implementazione

```
1 // Utility function per transazioni
2 const transaction = async (callback) => {
3   const client = await pool.connect();
4   try {
5     await client.query('BEGIN');
6     const result = await callback(client);
7     await client.query('COMMIT');
8     return result;
9   } catch (error) {
10    await client.query('ROLLBACK');
11    throw error;
12   } finally {
13     client.release();
14   }
15 };
```

Listing 6: Esempio Transazione Automatica

1.10.3 Strategie di Rollback

- **Rollback Automatico:** In caso di errore, tutte le operazioni della transazione vengono annullate
- **Gestione Connessioni:** Rilascio automatico delle connessioni database

- **Error Propagation:** Gli errori vengono propagati al livello superiore per logging
- **Stato Consistente:** Il database rimane sempre in uno stato coerente

1.10.4 Esempio Pratico: Pagamento con Aggiornamento Prenotazione

```
1 try {  
2   await client.query('BEGIN');  
3  
4   // 1. Crea il pagamento  
5   const payment = await Payment.create({  
6     booking_id, amount, payment_method,  
7     status: 'completed', transaction_id  
8   });  
9  
10  // 2. Aggiorna stato prenotazione  
11  await Booking.update(booking_id, { status: 'confirmed' });  
12  
13  await client.query('COMMIT');  
14  return payment;  
15 } catch (error) {  
16   await client.query('ROLLBACK');  
17   throw error;  
18 }
```

Listing 7: Transazione Pagamento Completa

Paginazione

Per gli endpoint che restituiscono liste di dati, l'API supporta paginazione basata su offset:

```
1 GET /api/bookings?page=2&limit=10&sort=created_at&order=desc
```

Listing 8: Parametri di Paginazione

```
1 {
2   "success": true,
3   "data": {
4     "items": [...],
5     "pagination": {
6       "page": 2,
7       "limit": 10,
8       "total": 150,
9       "totalPages": 15,
10      "hasNext": true,
11      "hasPrev": true
12    }
13  }
14 }
```

Listing 9: Risposta con Metadati di Paginazione

Content-Type e Accept Headers

L'API supporta esclusivamente JSON per input e output:

```
1 Content-Type: application/json
2 Accept: application/json
```

Listing 10: Headers Richiesti

Documentazione Interattiva

L'API include documentazione Swagger/OpenAPI accessibile durante lo sviluppo:

- **URL Sviluppo:** `http://localhost:3000/api-docs`
- **Formato:** OpenAPI 3.0.0
- **Caratteristiche:** Test interattivi, schemi completi, esempi

Health Check e Monitoring

Il sistema include endpoint per monitoraggio dello stato:

```
1 GET /api/health
```

Listing 11: Health Check Endpoint

```
1 {  
2   "success": true,  
3   "data": {  
4     "status": "ok",  
5     "timestamp": "2024-01-15T10:30:00Z",  
6     "version": "1.0.0",  
7     "database": "connected",  
8     "uptime": 3600  
9   }  
10 }
```

Listing 12: Risposta Health Check

Autenticazione e Autorizzazione

Sistema di Autenticazione JWT

L'API utilizza JSON Web Tokens (JWT) per l'autenticazione degli utenti. Il sistema implementa un approccio stateless che garantisce scalabilità e sicurezza.

2.1.1 Processo di Login

1. L'utente invia credenziali (email e password)
2. Il server verifica le credenziali contro il database
3. Se valide, genera un JWT token con payload utente
4. Il token viene restituito al client per le richieste successive

```
1 POST /api/users/login  
2 Content-Type: application/json  
3  
4 {  
5   "email": "mario.rossi@email.com",  
6   "password": "password123"  
7 }
```

Listing 13: Richiesta di Login

```
1 {
2   "success": true,
3   "message": "Login effettuato con successo",
4   "data": {
5     "user": {
6       "user_id": 1,
7       "name": "Mario",
8       "surname": "Rossi",
9       "email": "mario.rossi@email.com",
10      "role": "user",
11      "created_at": "2024-01-15T10:30:00Z"
12    },
13    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
14  }
15 }
```

Listing 14: Risposta Login Successful

2.1.2 Struttura del JWT Token

Il JWT token contiene le seguenti informazioni nel payload:

```
1 {
2   "user_id": 1,
3   "email": "mario.rossi@email.com",
4   "role": "user",
5   "iat": 1640995200, // Issued at
6   "exp": 1641081600 // Expiration (24 ore)
7 }
```

Listing 15: Payload JWT Token

Utilizzo del Token

Per accedere agli endpoint protetti, il client deve includere il token nell'header Authorization:

```
1 GET /api/users/profile
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
```

Listing 16: Header Authorization

Sistema di Autorizzazione Basato sui Ruoli

Il sistema implementa tre livelli di autorizzazione basati sui ruoli utente:

2.3.1 Ruoli Disponibili

Ruolo	Descrizione
user	Utente standard che può visualizzare spazi disponibili, creare prenotazioni e gestire il proprio profilo
manager	Manager di location che ha tutti i permessi utente plus gestione degli spazi nelle proprie location, visualizzazione prenotazioni della propria location
admin	Amministratore sistema con accesso completo: gestione utenti, promozione manager, gestione globale del sistema

Tabella 4: Ruoli Utente del Sistema

2.3.2 Matrice dei Permessi

Operazione	Pubblico	User	Manager	Admin
Visualizzare spazi pubblici	✓	✓	✓	✓
Registrazione/Login	✓	✓	✓	✓
Gestire proprio profilo	–	✓	✓	✓
Creare prenotazioni	–	✓	✓	✓
Visualizzare proprie prenotazioni	–	✓	✓	✓
Gestire spazi propria location	–	–	✓	✓
Visualizzare prenotazioni location	–	–	✓	✓
Gestire disponibilità	–	–	✓	✓
Gestire tutti gli utenti	–	–	–	✓
Promuovere manager	–	–	–	✓
Gestire tutte le location	–	–	–	✓
Accesso dashboard admin	–	–	–	✓

Tabella 5: Matrice Permessi per Ruolo

Middleware di Autenticazione

Il sistema utilizza middleware personalizzati per proteggere gli endpoint:

2.4.1 authMiddleware.protect

Verifica la presenza e validità del JWT token:

```

1 // Verifica presenza header Authorization
2 // Estrae il token dal formato "Bearer <token>"
3 // Verifica validità del token JWT
4 // Decodifica payload e ottiene informazioni utente
5 // Aggiunge req.user per controller successivi

```

Listing 17: Funzionamento authMiddleware.protect

2.4.2 authMiddleware.authorize

Controlla i permessi basati sui ruoli:

```
1 // Solo manager e admin possono accedere
2 router.get('/spaces',
3   authMiddleware.protect,
4   authMiddleware.authorize(['manager', 'admin']),
5   spaceController.getSpaces
6 );
7
8 // Solo admin può accedere
9 router.get('/admin/users',
10  authMiddleware.protect,
11  authMiddleware.authorize(['admin']),
12  adminController.getAllUsers
13 );
```

Listing 18: Esempi Utilizzo Authorization

Gestione delle Password

Il sistema implementa pratiche di sicurezza avanzate per la gestione delle password:

2.5.1 Hashing delle Password

- Utilizza bcrypt con salt factor 12
- Le password non vengono mai memorizzate in plain text
- Hash diversi anche per password identiche grazie al salt

2.5.2 Reset Password

Il sistema supporta due modalità di reset password:

1. **Password dimenticata:** Processo pubblico via email
2. **Cambio password:** Dal profilo utente autenticato

```
1 POST /api/users/request-password-reset
2 Content-Type: application/json
3
4 {
5   "email": "mario.rossi@email.com"
6 }
```

Listing 19: Richiesta Reset Password

Rate Limiting per Sicurezza

Protezione specifica per endpoint di autenticazione:

Endpoint	Limite	Protezione
/users/login	5 richieste/15min per IP	Attacchi brute force
/users/request-password-reset	5 richieste/15min per IP	Spam di reset
/users/register	Rate limiting generale	Registrazioni massive

Tabella 6: Rate Limiting Endpoints Autenticazione

Gestione Token FCM

Il sistema supporta notifiche push tramite Firebase Cloud Messaging:

```

1 POST /api/users/fcm-token
2 Authorization: Bearer <jwt-token>
3 Content-Type: application/json
4
5 {
6   "fcm_token": "fGH1jK2L3m4N5o6P7q8R9s0T..."
7 }
```

Listing 20: Salvataggio Token FCM

Logout e Invalidazione Token

Benché JWT sia stateless, il sistema supporta logout lato client:

```

1 POST /api/users/logout
2 Authorization: Bearer <jwt-token>
```

Listing 21: Endpoint Logout

```

1 {
2   "success": true,
3   "message": "Logout effettuato con successo",
4   "data": {
5     "message": "Token invalidato lato client"
6   }
7 }
```

Listing 22: Risposta Logout

Sicurezza delle Sessioni

- **Durata Token:** 24 ore (configurabile)
- **Rinnovo:** Automatic refresh non implementato per sicurezza
- **Scope:** Un token per tutte le operazioni autorizzate
- **Revoca:** Solo tramite scadenza naturale

Validazione Input

Tutti gli endpoint implementano validazione rigorosa degli input:

```
1 // Email: formato email valido, lunghezza massima 255
2 // Password: minimo 6 caratteri, massimo 255
3 // Rate limiting: 5 tentativi per IP ogni 15 minuti
4 // Sanitizzazione: rimozione caratteri pericolosi
```

Listing 23: Esempio Validazioni Login

Gestione Errori di Autenticazione

Il sistema fornisce messaggi di errore specifici ma sicuri:

Scenario	Codice	Messaggio
Token mancante	401	"Token di accesso richiesto"
Token scaduto	401	"Token scaduto, effettua nuovamente il login"
Token non valido	401	"Token non valido"
Permessi insufficienti	403	"Non hai i permessi per questa operazione"
Credenziali errate	401	"Credenziali non valide"
Account sospeso	401	"Account in attesa di approvazione"

Tabella 7: Messaggi di Errore Autenticazione

Endpoints API

Panoramica degli Endpoints

L'API CoWorkSpace è organizzata in 10 categorie principali di endpoint, ciascuna dedicata a specifiche funzionalità del sistema:

Categoria	Descrizione	Base Path
Users	Gestione utenti e autenticazione	/api/users
Locations	Gestione sedi coworking	/api/locations
Spaces	Gestione spazi prenotabili	/api/spaces
Space Types	Gestione tipologie di spazio	/api/space-types
Bookings	Gestione prenotazioni	/api/bookings
Availability	Gestione disponibilità spazi	/api/availability
Payments	Gestione pagamenti	/api/payments
Notifications	Gestione notifiche	/api/notifications
Manager	Endpoint specifici per manager	/api/manager
Admin	Endpoint amministrativi	/api/admin

Tabella 8: Categorie di Endpoint API

Endpoint Users

Gestione degli utenti, autenticazione e profili.

3.2.1 POST /api/users/register

Descrizione: Registrazione nuovo utente

Autenticazione: Pubblica

Parametri Body:

- **email** (*string*): Email utente (formato email valido)
- **password** (*string*): Password (minimo 8 caratteri)
- **name** (*string*): Nome utente (massimo 100 caratteri)
- **surname** (*string*): Cognome utente (massimo 100 caratteri)
- **requestManagerRole** (*boolean*): Richiesta ruolo manager (opzionale)

Risposte:

- **201**: Utente registrato con successo (può effettuare login)
- **202**: Utente registrato, in attesa approvazione manager
- **400**: Dati non validi
- **409**: Email già esistente

```
1 POST /api/users/register
2
3 {
4   "email": "mario.rossi@email.com",
5   "password": "Password123!",
6   "name": "Mario",
7   "surname": "Rossi",
8   "requestManagerRole": false
9 }
```

Listing 24: Esempio Registrazione Utente

3.2.2 POST /api/users/login

Descrizione: Login utente

Autenticazione: Pubblica + Rate Limiting

Parametri Body:

- **email** (*string*): Email registrata
- **password** (*string*): Password utente

Risposte:

- **200**: Login effettuato con successo
- **401**: Credenziali non valide
- **429**: Troppi tentativi di login

3.2.3 GET /api/users/profile

Descrizione: Ottieni profilo utente corrente

Autenticazione: JWT Required

3.2.4 GET /api/users/dashboard

Descrizione: Dashboard con statistiche e prenotazioni

Autenticazione: JWT Required

3.2.5 PUT /api/users/profile

Descrizione: Aggiorna profilo utente

Autenticazione: JWT Required

3.2.6 PUT /api/users/change-password

Descrizione: Cambia password utente

Autenticazione: JWT Required

3.2.7 POST /api/users/request-password-reset

Descrizione: Richiesta reset password

Autenticazione: Pubblica + Rate Limiting

3.2.8 POST /api/users/logout

Descrizione: Logout utente

Autenticazione: JWT Required

3.2.9 POST /api/users/fcm-token

Descrizione: Salva token FCM per notifiche push

Autenticazione: JWT Required

3.2.10 GET /api/users/check-email

Descrizione: Verifica se email è registrata

Autenticazione: Pubblica

Endpoint Locations

Gestione delle sedi coworking.

3.3.1 GET /api/locations

Descrizione: Lista tutte le location

Autenticazione: Pubblica

Parametri Query (opzionali):

- **city** (*string*): Filtra per città

- **search** (*string*): Ricerca nel nome o descrizione
- **manager_id** (*integer*): Filtra per manager

3.3.2 GET /api/locations/:id

Descrizione: Dettagli specifica location

Autenticazione: Pubblica

3.3.3 GET /api/locations/:id/spaces

Descrizione: Spazi di una location

Autenticazione: Pubblica

3.3.4 POST /api/locations

Descrizione: Crea nuova location

Autenticazione: Admin Only

3.3.5 PUT /api/locations/:id

Descrizione: Aggiorna location

Autenticazione: Admin Only

3.3.6 DELETE /api/locations/:id

Descrizione: Elimina location

Autenticazione: Admin Only

Endpoint Spaces

Gestione degli spazi prenotabili.

3.4.1 GET /api/spaces

Descrizione: Lista spazi con filtri avanzati

Autenticazione: Pubblica

Parametri Query (opzionali):

- **location_id** (*integer*): Filtra per location
- **space_type_id** (*integer*): Filtra per tipo spazio
- **capacity_min** (*integer*): Capacità minima
- **capacity_max** (*integer*): Capacità massima
- **price_max** (*number*): Prezzo massimo giornaliero
- **available_date** (*date*): Data disponibilità
- **city** (*string*): Filtra per città location

- **search** (*string*): Ricerca nel nome

```
1 GET /api/spaces?location_id=1&capacity_min=4&price_max=150&available_date=2024-01-20
```

Listing 25: Esempio Ricerca Spazi con Filtri

3.4.2 GET /api/spaces/:id

Descrizione: Dettagli specifico spazio

Autenticazione: Pubblica

3.4.3 GET /api/spaces/:id/availability

Descrizione: Disponibilità spazio per periodo

Autenticazione: Pubblica

3.4.4 POST /api/spaces

Descrizione: Crea nuovo spazio

Autenticazione: Manager/Admin

3.4.5 PUT /api/spaces/:id

Descrizione: Aggiorna spazio

Autenticazione: Manager/Admin

3.4.6 DELETE /api/spaces/:id

Descrizione: Elimina spazio

Autenticazione: Manager/Admin

Endpoint Space Types

Gestione tipologie di spazio.

3.5.1 GET /api/space-types

Descrizione: Lista tutte le tipologie

Autenticazione: Pubblica

3.5.2 GET /api/space-types/:id

Descrizione: Dettagli tipologia specifica

Autenticazione: Pubblica

3.5.3 POST /api/space-types

Descrizione: Crea nuova tipologia

Autenticazione: Admin Only

3.5.4 PUT /api/space-types/:id

Descrizione: Aggiorna tipologia

Autenticazione: Admin Only

3.5.5 DELETE /api/space-types/:id

Descrizione: Elimina tipologia

Autenticazione: Admin Only

Endpoint Bookings

Gestione del sistema di prenotazioni.

3.6.1 POST /api/bookings/check-availability

Descrizione: Verifica disponibilità spazio

Autenticazione: Pubblica

Parametri Body:

- **space_id** (*integer*): ID dello spazio
- **start_date** (*date*): Data inizio prenotazione
- **end_date** (*date*): Data fine prenotazione

3.6.2 POST /api/bookings

Descrizione: Crea nuova prenotazione

Autenticazione: JWT Required

Parametri Body:

- **space_id** (*integer*): ID dello spazio da prenotare
- **start_date** (*date*): Data inizio (formato YYYY-MM-DD)
- **end_date** (*date*): Data fine (formato YYYY-MM-DD)
- **notes** (*string*): Note aggiuntive (opzionale)

```
1 POST /api/bookings
2 Authorization: Bearer <jwt-token>
3
4 {
5   "space_id": 1,
6   "start_date": "2024-01-20",
7   "end_date": "2024-01-22",
8   "notes": "Riunione team marketing"
9 }
```

Listing 26: Esempio Creazione Prenotazione

3.6.3 GET /api/bookings

Descrizione: Lista prenotazioni utente corrente

Autenticazione: JWT Required

Parametri Query (opzionali):

- **status** (*string*): Filtra per stato (confirmed, pending, cancelled, completed)
- **from_date** (*date*): Prenotazioni da questa data
- **to_date** (*date*): Prenotazioni fino a questa data
- **page** (*integer*): Numero pagina (default: 1)
- **limit** (*integer*): Elementi per pagina (default: 10)

3.6.4 GET /api/bookings/:id

Descrizione: Dettagli prenotazione specifica

Autenticazione: JWT Required

3.6.5 PUT /api/bookings/:id

Descrizione: Aggiorna prenotazione

Autenticazione: JWT Required

3.6.6 DELETE /api/bookings/:id

Descrizione: Cancella prenotazione

Autenticazione: JWT Required

3.6.7 PUT /api/bookings/:id/confirm

Descrizione: Conferma prenotazione

Autenticazione: Manager/Admin

3.6.8 PUT /api/bookings/:id/complete

Descrizione: Completa prenotazione

Autenticazione: Manager/Admin

Endpoint Availability

Gestione disponibilità degli spazi.

3.7.1 GET /api/availability/space/:spaceId

Descrizione: Disponibilità spazio per periodo

Autenticazione: Pubblica

Parametri Query:

- **from_date** (*date*): Data inizio periodo
- **to_date** (*date*): Data fine periodo

3.7.2 POST /api/availability

Descrizione: Imposta disponibilità spazio

Autenticazione: Manager/Admin

3.7.3 PUT /api/availability/:id

Descrizione: Aggiorna disponibilità

Autenticazione: Manager/Admin

3.7.4 DELETE /api/availability/:id

Descrizione: Rimuovi disponibilità

Autenticazione: Manager/Admin

Endpoint Payments

Gestione sistema di pagamenti.

3.8.1 GET /api/payments

Descrizione: Lista pagamenti utente corrente

Autenticazione: JWT Required

3.8.2 GET /api/payments/:id

Descrizione: Dettagli pagamento specifico

Autenticazione: JWT Required

3.8.3 POST /api/payments

Descrizione: Crea nuovo pagamento

Autenticazione: JWT Required

Parametri Body:

- **booking_id** (*integer*): ID prenotazione da pagare
- **payment_method** (*string*): Metodo: credit_card (solo carta di credito)
- **amount** (*number*): Importo pagamento
- **transaction_id** (*string*): ID transazione gateway (opzionale)

3.8.4 PUT /api/payments/:id/refund

Descrizione: Rimborsa pagamento

Autenticazione: Manager/Admin

Endpoint Notifications

Gestione sistema notifiche.

3.9.1 GET /api/notifications

Descrizione: Lista notifiche utente corrente

Autenticazione: JWT Required

3.9.2 GET /api/notifications/:id

Descrizione: Dettagli notifica specifica

Autenticazione: JWT Required

3.9.3 PUT /api/notifications/:id/read

Descrizione: Segna notifica come letta

Autenticazione: JWT Required

3.9.4 POST /api/notifications/test

Descrizione: Invia notifica di test

Autenticazione: Admin Only

Endpoint Manager

Funzionalità specifiche per manager.

3.10.1 GET /api/manager/dashboard

Descrizione: Dashboard manager con statistiche

Autenticazione: Manager/Admin

3.10.2 GET /api/manager/location

Descrizione: Location gestita dal manager

Autenticazione: Manager/Admin

3.10.3 GET /api/manager/bookings

Descrizione: Prenotazioni location del manager

Autenticazione: Manager/Admin

3.10.4 GET /api/manager/spaces

Descrizione: Spazi della location del manager

Autenticazione: Manager/Admin

3.10.5 PUT /api/manager/spaces/:id

Descrizione: Aggiorna spazio della propria location

Autenticazione: Manager/Admin

Endpoint Admin

Funzionalità amministrative del sistema.

3.11.1 GET /api/admin/dashboard

Descrizione: Dashboard amministrativa completa

Autenticazione: Admin Only

3.11.2 GET /api/admin/users

Descrizione: Lista tutti gli utenti

Autenticazione: Admin Only

3.11.3 GET /api/admin/users/pending-managers

Descrizione: Utenti in attesa approvazione manager

Autenticazione: Admin Only

3.11.4 PUT /api/admin/users/:id/promote-manager

Descrizione: Promuovi utente a manager

Autenticazione: Admin Only

3.11.5 PUT /api/admin/users/:id/demote

Descrizione: Rimuovi ruolo manager

Autenticazione: Admin Only

3.11.6 GET /api/admin/bookings

Descrizione: Tutte le prenotazioni sistema

Autenticazione: Admin Only

3.11.7 GET /api/admin/stats

Descrizione: Statistiche globali sistema

Autenticazione: Admin Only

3.11.8 POST /api/admin/locations

Descrizione: Crea nuova location

Autenticazione: Admin Only

3.11.9 PUT /api/admin/locations/:id/assign-manager

Descrizione: Assegna manager a location

Autenticazione: Admin Only

Endpoint di Sistema

3.12.1 GET /api/health

Descrizione: Health check sistema

Autenticazione: Pubblica

```
1 {  
2   "success": true,  
3   "data": {  
4     "status": "ok",  
5     "timestamp": "2024-01-15T10:30:00Z",  
6     "version": "1.0.0",  
7     "database": "connected",  
8     "uptime": 3600  
9   }  
10 }
```

Listing 27: Risposta Health Check

3.12.2 GET /api/

Descrizione: Informazioni generali API

Autenticazione: Pubblica

3.12.3 GET /api-docs

Descrizione: Documentazione Swagger

Autenticazione: Sviluppo

Convenzioni degli Endpoint

L'API segue convenzioni REST standard:

Metodo	Utilizzo	Esempio
GET	Lettura risorse	GET /api/spaces - Lista spazi
POST	Creazione risorse	POST /api/bookings - Nuova prenotazione
PUT	Aggiornamento completo	PUT /api/spaces/:id - Aggiorna spazio
PATCH	Aggiornamento parziale	PATCH /api/users/profile - Aggiorna profilo
DELETE	Eliminazione risorse	DELETE /api/bookings/:id - Cancella prenotazione

Tabella 9: Convenzioni Metodi HTTP

Schemi Dati e Validazioni

Panoramica degli Schemi

L'API utilizza schemi dati strutturati che corrispondono alle entità del database. Ogni schema implementa validazioni rigorose per garantire integrità e sicurezza dei dati.

Schema User

Rappresenta gli utenti del sistema con i loro ruoli e permessi.

```
1 {  
2   "user_id": 1,  
3   "name": "Mario",  
4   "surname": "Rossi",  
5   "email": "mario.rossi@email.com",  
6   "role": "user",  
7   "is_password_reset_required": false,  
8   "fcm_token": "fGH1jK2L3m4N5o6P7q8R9s0T...",  
9   "manager_request_pending": false,  
10  "manager_request_date": null,  
11  "created_at": "2024-01-15T10:30:00Z"  
12 }
```

Listing 28: Schema User Completo

Validazioni Campo User:

Campo	Tipo	Validazioni	Vincoli
name	string	Lunghezza 1-100 caratteri	Obbligatorio
surname	string	Lunghezza 1-100 caratteri	Obbligatorio
email	string	Formato email valido, max 255 caratteri	Unico, obbligatorio
password	string	Minimo 8 caratteri, hash bcrypt	Obbligatorio
role	enum	user, manager, admin	Default: user
fcm_token	string	Max 255 caratteri	Opzionale

Tabella 10: Validazioni Schema User

Schema Location

Rappresenta le sedi fisiche dei coworking.

```

1 {
2   "location_id": 1,
3   "location_name": "CoWork Milano Centro",
4   "address": "Via Brera 10, Milano",
5   "city": "Milano",
6   "description": "Moderno spazio coworking nel cuore di Milano",
7   "manager_id": 2,
8   "manager": {
9     "user_id": 2,
10    "name": "Luca",
11    "surname": "Manager",
12    "email": "luca.manager@email.com"
13  },
14  "spaces_count": 15,
15  "active_bookings": 8
16 }
```

Listing 29: Schema Location

Validazioni Campo Location:

Campo	Tipo	Validazioni	Vincoli
location_name	string	Max 255 caratteri, non vuoto	Obbligatorio
address	string	Max 255 caratteri, non vuoto	Obbligatorio
city	string	Max 100 caratteri, non vuoto	Obbligatorio
description	text	Testo libero	Opzionale
manager_id	integer	Riferimento utente con ruolo manager	Opzionale

Tabella 11: Validazioni Schema Location

Schema Space Type

Definisce i tipi di spazio disponibili.

```

1 {
2   "space_type_id": 1,
3   "type_name": "Ufficio Privato",
4   "description": "Ufficio privato per singola persona o piccoli team",
5   "spaces_count": 25
6 }
```

Listing 30: Schema Space Type

Tipi Predefiniti del Sistema:

Tipo	Descrizione
Ufficio Privato	Ufficio privato per singola persona o piccoli team
Sala Riunioni	Sala attrezzata per meeting e riunioni di lavoro
Open Space	Spazio aperto condiviso per lavoro collaborativo
Coworking Desk	Singola postazione di lavoro in ambiente condiviso
Phone Booth	Cabina telefonica insonorizzata per chiamate private
Sala Conferenze	Ampia sala per conferenze e presentazioni
Focus Room	Stanza silenziosa per lavoro concentrato
Lounge Area	Area relax informale per incontri casual
Training Room	Aula per formazione e workshop
Event Space	Spazio per eventi e networking

Tabella 12: Tipologie di Spazio Predefinite

Schema Space

Rappresenta gli spazi prenotabili all'interno delle location.

```

1 {
2   "space_id": 1,
3   "location_id": 1,
4   "space_type_id": 1,
5   "space_name": "Stanza 101",
6   "description": "Ufficio privato con vista sul cortile",
7   "capacity": 4,
8   "price_per_hour": 15.50,
9   "price_per_day": 120.00,
10  "opening_time": "09:00:00",
11  "closing_time": "18:00:00",
12  "available_days": [1, 2, 3, 4, 5],
13  "booking_advance_days": 30,
14  "status": "active",
15  "location": {
16    "location_name": "CoWork Milano Centro",
17    "city": "Milano",
18    "address": "Via Brera 10, Milano"
19  },
20  "space_type": {
21    "type_name": "Ufficio Privato",
22    "description": "Ufficio privato per singola persona o piccoli team"
23  }
24 }
```

Listing 31: Schema Space Completo

Validazioni Campo Space:

Campo	Tipo	Validazioni	Vincoli
space_name	string	Max 255 caratteri, non vuoto	Obbligatorio
capacity	integer	Maggiore di 0, massimo 100	Obbligatorio
price_per_hour	decimal	Maggiore di 0, max 2 decimali	Obbligatorio
price_per_day	decimal	Maggiore di 0, max 2 decimali	Obbligatorio
opening_time	time	Formato HH:MM:SS	Default: 09:00:00
closing_time	time	Formato HH:MM:SS, dopo opening_time	Default: 18:00:00
available_days	array	Numeri 1-7 (1=Lunedì, 7=Domenica)	Default: [1,2,3,4,5,6,7]
status	enum	active, maintenance, inactive	Default: active

Tabella 13: Validazioni Schema Space

Schema Booking

Rappresenta le prenotazioni degli spazi.

```
1 {
2   "booking_id": 1,
3   "user_id": 1,
4   "space_id": 1,
5   "start_date": "2024-01-20",
6   "end_date": "2024-01-22",
7   "total_days": 3,
8   "total_price": 360.00,
9   "status": "confirmed",
10  "payment_status": "completed",
11  "notes": "Riunione team marketing",
12  "created_at": "2024-01-15T10:30:00Z",
13  "user": {
14    "name": "Mario",
15    "surname": "Rossi",
16    "email": "mario.rossi@email.com"
17  },
18  "space": {
19    "space_name": "Stanza 101",
20    "capacity": 4,
21    "price_per_day": 120.00,
22    "location": {
23      "location_name": "CoWork Milano Centro",
24      "city": "Milano"
25    }
26  },
27  "payment": {
28    "payment_id": 1,
29    "amount": 360.00,
30    "payment_method": "credit_card",
31    "status": "completed",
32    "payment_date": "2024-01-15T10:35:00Z"
33  }
34 }
```

Listing 32: Schema Booking Completo

Stati Prenotazione:

Stato	Descrizione
pending	Prenotazione creata, in attesa di conferma/pagamento
confirmed	Prenotazione confermata e pagata
completed	Prenotazione utilizzata e completata
cancelled	Prenotazione cancellata dall'utente o sistema

Tabella 14: Stati Prenotazione

Validazioni Campo Booking:

Campo	Tipo	Validazioni	Vincoli
start_date	date	Formato YYYY-MM-DD, >= data corrente	Obbligatorio
end_date	date	Formato YYYY-MM-DD, >= start_date	Obbligatorio
total_days	integer	Calcolato automaticamente	Auto-generato
total_price	decimal	Calcolato automaticamente	Auto-generato
status	enum	pending, confirmed, cancelled, completed	Default: pending
notes	text	Max 1000 caratteri	Opzionale

Tabella 15: Validazioni Schema Booking

Schema Payment

Gestisce i pagamenti delle prenotazioni.

```

1 {
2   "payment_id": 1,
3   "booking_id": 1,
4   "amount": 360.00,
5   "payment_date": "2024-01-15T10:35:00Z",
6   "payment_method": "credit_card",
7   "status": "completed",
8   "transaction_id": "txn_1234567890",
9   "booking": {
10    "booking_id": 1,
11    "start_date": "2024-01-20",
12    "end_date": "2024-01-22",
13    "space_name": "Stanza 101"
14  }
15 }
```

Listing 33: Schema Payment

Metodi di Pagamento:

Metodo	Descrizione
credit_card	Pagamento con carta di credito

Tabella 16: Metodi di Pagamento Supportati

Stati Pagamento:

Stato	Descrizione
pending	Pagamento in attesa di elaborazione
completed	Pagamento completato con successo
failed	Pagamento fallito
refunded	Pagamento rimborsato

Tabella 17: Stati Pagamento

Schema Notification

Sistema completo di notifiche multi-canale.

```

1 {
2   "notification_id": 1,
3   "user_id": 1,
4   "type": "email",
5   "channel": "booking_confirmation",
6   "recipient": "mario.rossi@email.com",
7   "subject": "Conferma prenotazione #1",
8   "content": "La tua prenotazione è stata confermata",
9   "template_name": "booking_confirmation.html",
10  "template_data": {
11    "userName": "Mario",
12    "spaceName": "Stanza 101",
13    "startDate": "2024-01-20",
14    "endDate": "2024-01-22"
15  },
16  "status": "delivered",
17  "metadata": {
18    "provider": "sendgrid",
19    "messageId": "abc123"
20  },
21  "booking_id": 1,
22  "sent_at": "2024-01-15T10:35:00Z",
23  "delivered_at": "2024-01-15T10:36:00Z",
24  "read_at": null,
25  "retry_count": 0,
26  "created_at": "2024-01-15T10:35:00Z"
27 }

```

Listing 34: Schema Notification

Tipi di Notifica:

Tipo	Descrizione
email	Notifica via email
push	Notifica push mobile
sms	Notifica SMS

Tabella 18: Tipi di Notifica

Canali di Notifica:

Canale	Descrizione
booking_confirmation	Conferma prenotazione
booking_cancellation	Cancellazione prenotazione
payment_success	Pagamento riuscito
payment_failed	Pagamento fallito
payment_refund	Rimborso pagamento
booking_reminder	Promemoria prenotazione
user_registration	Benvenuto nuovi utenti
password_reset	Reset password

Tabella 19: Canali di Notifica

Schema Availability

Gestisce la disponibilità giornaliera degli spazi.

```

1 {
2   "availability_id": 1,
3   "space_id": 1,
4   "availability_date": "2024-01-20",
5   "is_available": true,
6   "space": {
7     "space_name": "Stanza 101",
8     "capacity": 4,
9     "price_per_day": 120.00
10  }
11 }
```

Listing 35: Schema Availability

Validazioni Globali del Sistema**4.10.1 Validazioni Date**

- Tutte le date devono essere in formato ISO 8601 (YYYY-MM-DD)
- Le date di prenotazione non possono essere nel passato
- La data fine deve essere maggiore o uguale alla data inizio
- Massimo 90 giorni di anticipo per le prenotazioni

4.10.2 Validazioni Email

```
1 ^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$
```

Listing 36: Regex Validazione Email

4.10.3 Validazioni Password

- Lunghezza minima: 8 caratteri
- Deve contenere almeno una lettera maiuscola
- Deve contenere almeno una lettera minuscola
- Deve contenere almeno un numero
- Caratteri speciali consigliati ma non obbligatori

4.10.4 Validazioni Numeriche

- Prezzi: massimo 2 decimali, maggiori di 0
- Capacità: numeri interi positivi, massimo 100
- ID: numeri interi positivi

Gestione Errori di Validazione

Il sistema restituisce errori di validazione strutturati:

```
1 {
2   "success": false,
3   "message": "Dati non validi",
4   "errors": [
5     {
6       "type": "field",
7       "field": "email",
8       "message": "Email deve essere un indirizzo valido",
9       "value": "email-non-valida",
10      "location": "body"
11    },
12    {
13      "type": "field",
14      "field": "start_date",
15      "message": "Data inizio non può essere nel passato",
16      "value": "2023-01-01",
17      "location": "body"
18    },
19    {
20      "type": "business_rule",
21      "message": "Lo spazio non è disponibile nelle date selezionate",
22      "details": {
23        "space_id": 1,
24        "conflicting_dates": ["2024-01-20", "2024-01-21"]
25      }
26    }
27  ]
28 }
```

Listing 37: Esempio Errore Validazione Multipla

Esempi di Utilizzo e Casi d'Uso

Panoramica dei Casi d'Uso

Questa sezione presenta esempi pratici di utilizzo dell'API CoWorkSpace, organizzati per scenari comuni di integrazione. Gli esempi includono le chiamate API complete con headers, parametri e risposte.

Caso d'Uso 1: Registrazione e Login Utente

Scenario completo di registrazione di un nuovo utente e successivo login.

5.2.1 Passo 1: Verifica Email Disponibilità

```
1 GET /api/users/check-email?email=mario.rossi@email.com
2 Content-Type: application/json
```

Listing 38: Verifica Email Esistente

```
1 {
2   "success": true,
3   "data": {
4     "email": "mario.rossi@email.com",
5     "exists": false
6   }
7 }
```

Listing 39: Risposta Email Non Esistente

5.2.2 Passo 2: Registrazione Nuovo Utente

```
1 POST /api/users/register
2 Content-Type: application/json
3
4 {
5   "email": "mario.rossi@email.com",
6   "password": "Password123!",
7   "name": "Mario",
8   "surname": "Rossi",
9   "requestManagerRole": false
10 }
```

Listing 40: Registrazione Utente

```
1 {
2   "success": true,
3   "message": "Utente registrato con successo",
4   "data": {
5     "user": {
6       "user_id": 15,
7       "name": "Mario",
8       "surname": "Rossi",
9       "email": "mario.rossi@email.com",
10      "role": "user",
11      "created_at": "2024-01-15T10:30:00Z"
12    },
13    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
14    "canLogin": true
15  }
16 }
```

Listing 41: Risposta Registrazione Riuscita

5.2.3 Passo 3: Login Utente

```
1 POST /api/users/login
2 Content-Type: application/json
3
4 {
5   "email": "mario.rossi@email.com",
6   "password": "Password123!"
7 }
```

Listing 42: Login Utente

Caso d'Uso 2: Ricerca e Prenotazione Spazio

Workflow completo di ricerca spazi disponibili e creazione prenotazione.

5.3.1 Passo 1: Ricerca Spazi con Filtri

```
1 GET
  /api/spaces?city=Milano&capacity_min=4&price_max=150&available_date=2024-01-20
2 Content-Type: application/json
```

Listing 43: Ricerca Spazi Disponibili

```
1 {
2   "success": true,
3   "data": {
4     "spaces": [
5       {
6         "space_id": 1,
7         "space_name": "Stanza 101",
8         "capacity": 6,
9         "price_per_day": 120.00,
10        "price_per_hour": 15.50,
11        "description": "Ufficio privato con vista sul cortile",
12        "location": {
13          "location_id": 1,
14          "location_name": "CoWork Milano Centro",
15          "city": "Milano",
16          "address": "Via Brera 10, Milano"
17        },
18        "space_type": {
19          "type_name": "Ufficio Privato"
20        },
21        "availability": {
22          "is_available": true,
23          "next_available_date": "2024-01-20"
24        }
25      }
26    ],
27    "total": 1,
28    "filters_applied": {
29      "city": "Milano",
30      "capacity_min": 4,
31      "price_max": 150,
32      "available_date": "2024-01-20"
33    }
34  }
35 }
```

Listing 44: Risposta Spazi Disponibili

5.3.2 Passo 2: Verifica Disponibilità Specifica

```
1 POST /api/bookings/check-availability
2 Content-Type: application/json
3
4 {
5   "space_id": 1,
6   "start_date": "2024-01-20",
7   "end_date": "2024-01-22"
8 }
```

Listing 45: Verifica Disponibilità Periodo

```
1 {
2   "success": true,
3   "data": {
4     "available": true,
5     "space_id": 1,
6     "period": {
7       "start_date": "2024-01-20",
8       "end_date": "2024-01-22",
9       "total_days": 3
10    },
11    "pricing": {
12      "price_per_day": 120.00,
13      "total_price": 360.00,
14      "currency": "EUR"
15    },
16    "space_details": {
17      "space_name": "Stanza 101",
18      "capacity": 6,
19      "location_name": "CoWork Milano Centro"
20    }
21  }
22 }
```

Listing 46: Risposta Disponibilità Confermata

5.3.3 Passo 3: Creazione Prenotazione

```
1 POST /api/bookings
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
4
5 {
6   "space_id": 1,
7   "start_date": "2024-01-20",
8   "end_date": "2024-01-22",
9   "notes": "Riunione team marketing - progetto Q1"
10 }
```

Listing 47: Creazione Prenotazione

```
1 {
2   "success": true,
3   "message": "Prenotazione creata con successo",
4   "data": {
5     "booking": {
6       "booking_id": 25,
7       "user_id": 15,
8       "space_id": 1,
9       "start_date": "2024-01-20",
10      "end_date": "2024-01-22",
11      "total_days": 3,
12      "total_price": 360.00,
13      "status": "pending",
14      "payment_status": "pending",
15      "notes": "Riunione team marketing - progetto Q1",
16      "created_at": "2024-01-15T11:00:00Z"
17    },
18    "next_steps": {
19      "payment_required": true,
20      "payment_deadline": "2024-01-16T11:00:00Z"
21    }
22  }
23 }
```

Listing 48: Risposta Prenotazione Creata

Caso d'Uso 3: Gestione Pagamento

Workflow di pagamento per confermare una prenotazione.

5.4.1 Passo 1: Creazione Pagamento

```
1 POST /api/payments
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
4
5 {
6   "booking_id": 25,
7   "payment_method": "credit_card",
8   "amount": 360.00,
9   "transaction_id": "txn_cc_1234567890"
10 }
```

Listing 49: Processamento Pagamento

```
1 {
2   "success": true,
3   "message": "Pagamento completato con successo",
4   "data": {
5     "payment": {
6       "payment_id": 18,
7       "booking_id": 25,
8       "amount": 360.00,
9       "payment_method": "credit_card",
10      "status": "completed",
11      "transaction_id": "txn_cc_1234567890",
12      "payment_date": "2024-01-15T11:05:00Z"
13    },
14    "booking_updated": {
15      "booking_id": 25,
16      "status": "confirmed",
17      "payment_status": "completed"
18    },
19    "notifications_sent": [
20      "booking_confirmation_email",
21      "payment_success_email"
22    ]
23  }
24 }
```

Listing 50: Risposta Pagamento Completato

Caso d'Uso 4: Dashboard Utente

Visualizzazione completa del profilo utente con statistiche.

```
1 GET /api/users/dashboard
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
```

Listing 51: Richiesta Dashboard Utente

```
1 {
2   "success": true,
3   "data": {
4     "user": {
5       "user_id": 15,
6       "name": "Mario",
7       "surname": "Rossi",
8       "email": "mario.rossi@email.com",
9       "role": "user",
10      "created_at": "2024-01-15T10:30:00Z"
11    },
12    "statistics": {
13      "total_bookings": 5,
14      "completed_bookings": 3,
15      "confirmed_bookings": 1,
16      "cancelled_bookings": 1,
17      "total_spent": 890.00,
18      "total_days_booked": 12,
19      "locations_visited": 2,
20      "space_types_used": 3
21    }
22  }
```

```

21 },
22 "bookings": {
23   "upcoming": [
24     {
25       "booking_id": 25,
26       "start_date": "2024-01-20",
27       "end_date": "2024-01-22",
28       "space_name": "Stanza 101",
29       "location_name": "CoWork Milano Centro",
30       "status": "confirmed"
31     }
32   ],
33   "recent": [
34     {
35       "booking_id": 24,
36       "start_date": "2024-01-10",
37       "end_date": "2024-01-12",
38       "space_name": "Sala Riunioni A",
39       "location_name": "CoWork Roma",
40       "status": "completed"
41     }
42   ],
43   "total": 5
44 },
45 "preferences": {
46   "top_locations": [
47     {
48       "location_name": "CoWork Milano Centro",
49       "city": "Milano",
50       "booking_count": 3
51     }
52   ],
53   "top_space_types": [
54     {
55       "type_name": "Ufficio Privato",
56       "booking_count": 3
57     }
58   ]
59 }
60 }
61 }

```

Listing 52: Risposta Dashboard Completa

Caso d'Uso 5: Workflow Manager

Gestione degli spazi da parte di un manager di location.

5.6.1 Passo 1: Dashboard Manager

```

1 GET /api/manager/dashboard
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

```

Listing 53: Dashboard Manager

5.6.2 Passo 2: Visualizza Prenotazioni Location

```
1 GET /api/manager/bookings?status=pending&from_date=2024-01-15
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Listing 54: Prenotazioni da Gestire

5.6.3 Passo 3: Conferma Prenotazione

```
1 PUT /api/bookings/25/confirm
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
4
5 {
6   "manager_notes": "Prenotazione approvata - cliente verificato"
7 }
```

Listing 55: Conferma Prenotazione Manager

Caso d'Uso 6: Gestione Notifiche

Sistema di notifiche e comunicazioni utente.

5.7.1 Salvataggio Token FCM

```
1 POST /api/users/fcm-token
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
4
5 {
6   "fcm_token": "fGH1jK2L3m4N5o6P7q8R9s0T1u2V3w4X5y6Z..."
7 }
```

Listing 56: Registrazione per Notifiche Push

5.7.2 Recupero Notifiche

```
1 GET /api/notifications?status=unread&limit=10
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Listing 57: Lista Notifiche Utente

Caso d'Uso 7: Workflow Admin

Operazioni amministrative di sistema.

5.8.1 Dashboard Amministrativa

```
1 GET /api/admin/dashboard
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Listing 58: Statistiche Sistema

5.8.2 Promozione Manager

```
1 PUT /api/admin/users/15/promote-manager
2 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
3 Content-Type: application/json
4
5 {
6   "location_id": 1,
7   "admin_notes": "Utente qualificato per gestione location Milano"
8 }
```

Listing 59: Promuovi Utente a Manager

Gestione Errori Comuni

5.9.1 Errore Autenticazione

```
1 {
2   "success": false,
3   "message": "Token scaduto, effettua nuovamente il login",
4   "error": "TokenExpiredError",
5   "code": "AUTH_TOKEN_EXPIRED"
6 }
```

Listing 60: Token Scaduto

5.9.2 Errore Disponibilità

```
1 {
2   "success": false,
3   "message": "Lo spazio non è disponibile nelle date selezionate",
4   "error": "SPACE_NOT_AVAILABLE",
5   "details": {
6     "space_id": 1,
7     "requested_period": {
8       "start_date": "2024-01-20",
9       "end_date": "2024-01-22"
10    },
11    "conflicting_bookings": [
12      {
13        "booking_id": 23,
14        "start_date": "2024-01-21",
15        "end_date": "2024-01-23"
16      }
17    ],
18    "alternative_dates": [
19      "2024-01-24",
20      "2024-01-25",
21      "2024-01-26"
22    ]
23  }
24 }
```

Listing 61: Spazio Non Disponibile

Best Practices per Integrazione

5.10.1 Autenticazione

- Memorizza il JWT token in modo sicuro (localStorage per web, KeyChain per mobile)
- Implementa refresh automatico prima della scadenza
- Gestisci logout automatico su errori 401
- Non inviare credenziali in query parameters

5.10.2 Chiamate API

- Usa sempre HTTPS in produzione
- Implementa retry logic per errori 5xx
- Rispetta i limiti di rate limiting
- Valida input lato client prima dell'invio

5.10.3 Gestione Errori

- Implementa handling specifico per ogni codice di errore
- Mostra messaggi user-friendly basati su `message`
- Logga dettagli tecnici da `error` per debugging
- Implementa fallback per errori di rete

Esempi SDK/Client

5.11.1 JavaScript/TypeScript Client

```
1 class CoWorkSpaceAPI {
2   constructor(baseUrl, token = null) {
3     this.baseUrl = baseUrl;
4     this.token = token;
5   }
6
7   async login(email, password) {
8     const response = await fetch(`${this.baseUrl}/users/login`, {
9       method: 'POST',
10      headers: { 'Content-Type': 'application/json' },
11      body: JSON.stringify({ email, password })
12    });
13
14    const data = await response.json();
15    if (data.success) {
16      this.token = data.data.token;
17    }
18    return data;
19  }
```

```
20
21 async getSpaces(filters = {}) {
22   const queryString = new URLSearchParams(filters).toString();
23   const response = await
24     fetch(`${this.baseURL}/spaces?${queryString}`);
25   return response.json();
26 }
27
28 async createBooking(spaceId, startDate, endDate, notes = '') {
29   const response = await fetch(`${this.baseURL}/bookings`, {
30     method: 'POST',
31     headers: {
32       'Content-Type': 'application/json',
33       'Authorization': 'Bearer ${this.token}'
34     },
35     body: JSON.stringify({
36       space_id: spaceId,
37       start_date: startDate,
38       end_date: endDate,
39       notes
40     })
41   });
42   return response.json();
43 }
44
45 // Utilizzo
46 const api = new CoWorkSpaceAPI('https://api.coworkspace.com/api');
47 await api.login('user@email.com', 'password');
48 const spaces = await api.getSpaces({ city: 'Milano', capacity_min: 4 });
```

Listing 62: Esempio Client JavaScript

Testing API

Esempi di test per validare le funzionalità API.

5.12.1 Test con curl

```
1 curl -X POST https://api.coworkspace.com/api/users/login \
2   -H "Content-Type: application/json" \
3   -d '{"email":"user@email.com","password":"password123}"'
```

Listing 63: Test Login con curl

5.12.2 Test con Postman

Importa la collezione Postman disponibile all'endpoint:

```
1 GET /api/postman-collection
```