# Documentazione di Progetto — EventManagement

# (bozza 0.1)

**Versione:** 0.1 (prima stesura)  
**Autore:** Mohamed Aly  
**Data:**

## 1. Panoramica

**EventManagement** è una web app (ASP.NET Core + Blazor) per la gestione di eventi, iscrizioni e commenti, con ruoli differenziati (**Utente**, **Supervisore**, **Admin**) e notifiche via **SMTP**. Persistenza con **EF Core** (migrazioni presenti) e database **SQLite** per ambiente di sviluppo.

### Obiettivi

* Permettere agli utenti di esplorare eventi, vedere i dettagli e **iscriversi**.
* Abilitare commenti e **moderazione** dei contenuti.
* Fornire al personale autorizzato la **gestione completa** di eventi, iscritti, permessi e utenti.

### Ambito

**In scope** - Autenticazione/registrazione con **conferma email**. - CRUD eventi, iscrizioni (con **disiscrizione**), commenti e **domande Q&A**. - Gestione utenti/ruoli/permessi con **matrice** (a cura del Supervisore).

**Fuori ambito (non implementato)** - **Pagamenti online** e monetizzazione. - Notifiche email **transazionali aggiuntive** (reminder, annullamenti, ecc.). - Integrazioni con calendari/API esterne. - Requisiti formali di **accessibilità** e **localizzazione** (i18n) avanzati. —

## 2. Stakeholder e Ruoli

* **Visitatore**: non autenticato; consulta lista eventi e dettagli base.
* **Utente registrato**: può iscriversi, **disiscriversi**, commentare e **porre domande** sugli eventi.
* **Supervisore**: **gestisce la matrice dei ruoli/permessi**; per policy può operare su tutte le aree dell’applicazione.
* **Admin**: può operare **entro i permessi concessi dal Supervisore** (non ha poteri illimitati di default).
* **SMTP Provider**: usato **solo** per email di conferma (registrazione, iscrizione).

## 3. Mappa Pagine e Navigazione

**Router**: Components/Routes.razor con MainLayout.

**Pagine e route** - / → Components/Pages/Home.razor - /eventdetails/{Id:int} → Components/Pages/EventDetails.razor - /me/subscriptions → Components/Pages/MySubscriptions.razor - /account/login → Components/Pages/Account/Login.razor - /account/register → Components/Pages/Account/Register.razor - /account/confirm → Components/Pages/Account/Confirm.razor - /account/resend-confirmation → Components/Pages/Account/ResendConfirmation.razor - /admin/events → Components/Pages/Admin/EventsAdmin.razor - /admin/events/edit e /admin/events/edit/{Id:int} → Components/Pages/Admin/EventEdit.razor - /admin/events/{Id:int}/subscribers → Components/Pages/Admin/EventSubscribers.razor - /admin/events/{EventId:int}/comments → Components/Pages/Admin/EventComments.razor - /admin/users → Components/Pages/Admin/UserManagement.razor - /admin/rolepermissions → Components/Pages/Admin/RolePermissions.razor - /supervisor/users → Components/Pages/Supervisor/UserManagement.razor - /Error → Components/Pages/Error.razor

**Componenti commenti** (non route dedicate): Components/CommentItem.razor, Components/UserCommentItem.razor usate dentro EventDetails.razor.

## 4. Requisiti Funzionali (FR)

**FR-1 Autenticazione e Identity** - Registrazione utente con email e password, **conferma email** via link.

**FR-2 Gestione Eventi (Admin/Supervisore)** - Creazione, modifica, pubblicazione, archiviazione/cancellazione evento. - Campi minimi: Titolo, Descrizione, Luogo, Data/Ora inizio-fine, **Categoria**, **Numero massimo di partecipanti**.

**FR-3 Iscrizioni (Utente)** - Iscrizione/annullamento (**disiscrizione**) a un evento con controllo capienza. - **Email inviata solo per la conferma registrazione** (non per la singola iscrizione), - Vista personale “I miei eventi”.

**FR-4 Commenti/Discussioni (Q&A via commenti)** - Commenti con **thread** (campo ParentId) su ogni evento. - Le **domande** vengono poste come commenti; **Admin** o **Supervisore** rispondono tramite reply (flag FromAdmin). - Moderazione: hide/show (IsHidden), flags in db.

**FR-5 Amministrazione & Permessi** - Gestione utenti, ruoli e **matrice di permessi** granulari (Security/PermissionHandler.cs, Permissions.cs). - Il **Supervisore** vede e modifica la matrice; l’**Admin** opera entro i permessi concessi. - Elenco iscritti per evento, statistiche base (n. iscritti per evento).

**FR-6 Email** - **Solo**: conferma registrazione. - **Non** inclusi: reminder e altre notifiche transazionali.

**FR-7 Ricerca/Filtri** - Ricerca eventi per **titolo, data, luogo, categoria**.

## 5. Requisiti Non Funzionali (NFR)

* **Sicurezza**: Identity + DataProtection keys (Data/keys), enforcement permessi.
* **Persistenza**: **SQLite** (sviluppo/training) via EF Core.
* **Affidabilità**: gestione errori (Error.razor), logging standard ASP.NET.

**Non coperti in questa versione (training)** - Target di **performance** specifici (es. TTI < 2s con 100 eventi). - **Scalabilità**/linee guida prod (es. SQL Server, replica, ecc.). - **Accessibilità** formale (WCAG) e **internazionalizzazione** (i18n).

## 6. Casi d’Uso (UML Use Case)

@startuml  
left to right direction  
actor Visitatore  
actor Utente  
actor Supervisore  
actor Admin  
actor SMTP as S  
  
rectangle "EventManagement" {  
 usecase UC1 as "Registrazione"  
 usecase UC2 as "Conferma email"  
 usecase UC3 as "Login/Logout"  
 usecase UC4 as "Cercare/Consultare eventi"  
 usecase UC5 as "Iscriversi a un evento"  
 usecase UC5b as "Disiscriversi da un evento"  
 usecase UC6 as "Commentare / Chiedere (thread)"  
 usecase UC7 as "Rispondere (Admin/Supervisore)"  
 usecase UC8 as "Moderare commenti"  
 usecase UC9 as "Gestire eventi (CRUD)"  
 usecase UC10 as "Gestire utenti/ruoli/permessi"  
}  
  
Visitatore --> UC4  
Utente --> UC1  
Utente --> UC2  
Utente --> UC3  
Utente --> UC4  
Utente --> UC5  
Utente --> UC5b  
Utente --> UC6  
Admin --> UC7  
Supervisore --> UC7  
Supervisore --> UC8  
Admin --> UC8  
Admin --> UC9  
Supervisore --> UC9  
Admin --> UC10  
Supervisore --> UC10  
UC2 --> S : invio link  
@enduml

## 7. Modello Dati (UML Class / ER) — allineato al codice

@startuml  
class Event {  
 +Id: int  
 +Title: string  
 +Description: string  
 +StartDateTime: DateTime  
 +Location: string  
 +Category: string  
 +MaxParticipants: int  
 +Subscriptions: List<Subscription>  
}  
  
class Subscription {  
 +Id: int  
 +UserId: string  
 +EventId: int  
 +DataSubscription: DateTime  
}  
  
class Comment {  
 +Id: int  
 +EventId: int  
 +ParentId: int?  
 +Body: string  
 +UserId: string  
 +UserDisplayName: string  
 +FromAdmin: bool  
 +IsHidden: bool  
 +CreatedAt: DateTime  
}  
  
Event "1" --> "\*" Subscription  
Event "1" --> "\*" Comment  
Comment "1" --> "\*" Comment : replies (ParentId)  
@enduml

## 8. Architettura

### 8.1 Vista logica a strati

* **Presentazione (Blazor .razor)**: Pages, Components, layout e NavMenu.
* **Applicazione/Servizi**: Services/EventService.cs, SmtpEmailSender.cs, utilità Shared/Md.cs, UiSignal.cs .
* **Sicurezza**: Security/PermissionHandler.cs, Permissions.cs con enforcement a livello di UI e servizio.
* **Persistenza**: Data/AppDbContext.cs con **EF Core**; Migrations/\* (code-first); file DB eventmanagement.db (sviluppo).
* **Configurazione**: appsettings\*.json (incluso SmtpOptions).

### 8.2 Diagramma Component (UML)

@startuml  
package WebUI {  
 [Razor Pages/Components]  
 [MainLayout]  
}  
package Application {  
 [EventService]  
 [PermissionHandler]  
 [SmtpEmailSender]  
}  
package Infrastructure {  
 [AppDbContext]  
 [EF Core Migrations]  
 [Identity]  
}  
[Razor Pages/Components] --> [EventService]  
[Razor Pages/Components] --> [PermissionHandler]  
[EventService] --> [AppDbContext]  
[SmtpEmailSender] ..> SMTP : SMTP  
[PermissionHandler] ..> [Identity]  
[AppDbContext] ..> SQLite : connection  
@enduml

### 8.3 Diagramma Deployment (UML)

@startuml  
node "Server ASP.NET Core" {  
 component "EventManagement" <<Blazor Server>>  
}  
node "Database" {  
 database "SQLite (dev)" #DDDDDD  
}  
node "SMTP Provider" #DDDDFF  
  
"EventManagement" --> "Database"  
"EventManagement" --> "SMTP Provider"  
@enduml

## 9. Sequenze principali (UML Sequence)

**9.1 Registrazione e conferma email**

@startuml  
actor Utente  
participant UI as "Blazor UI"  
participant Identity  
participant SMTP  
  
Utente -> UI: Compila form registrazione  
UI -> Identity: CreateUser(email,pwd)  
Identity --> UI: UserCreated(Id, EmailConfirmed=false)  
UI -> SMTP: SendConfirmationEmail(link)  
Utente -> UI: Clic link di conferma  
UI -> Identity: ConfirmEmail(token)  
Identity --> UI: EmailConfirmed=true  
UI --> Utente: Accesso abilitato  
@enduml

**9.2 Iscrizione/Disiscrizione**

@startuml  
actor Utente  
participant UI  
participant EventService  
participant DB  
  
Utente -> UI: Subscribe / Unsubscribe  
UI -> EventService: SubscribeAsync/UnsubscribeAsync(eventId, userId)  
EventService -> DB: Insert/Remove Subscription  
DB --> EventService: OK  
UI --> Utente: Stato aggiornato  
@enduml

**9.3 Thread commenti (Q&A)**

@startuml  
actor Utente  
actor Admin  
actor Supervisore  
participant UI  
participant DB  
  
Utente -> UI: Post comment (Body)  
UI -> DB: Insert Comment(ParentId=null)  
Admin -> UI: Reply (FromAdmin=true)  
UI -> DB: Insert Comment(ParentId=Id, FromAdmin=true)  
Supervisore -> UI: Hide/Show  
UI -> DB: Update Comment.IsHidden  
@enduml

## 10. Activity / State

**10.1 Activity — Workflow moderazione**

@startuml  
start  
:Nuovo commento;  
if (Flag automatico?) then (sì)  
 :Marca come flagged;  
 :In coda revisione;  
else (no)  
 :Pubblica direttamente;  
endif  
:Supervisore rivede;  
if (Approva?) then (sì)  
 :Visible=true;  
else (no)  
 :Visible=false;  
endif  
stop  
@enduml

**10.2 State — Stato evento (proposto)**

@startuml  
[\*] --> Draft  
Draft --> Published : Pubblica  
Published --> Closed : Raggiunta data fine o capienza  
Published --> Cancelled : Annulla  
Closed --> [\*]  
Cancelled --> [\*]  
@enduml

## 11. Sicurezza e Permessi

* **Identity** per autenticazione e conferma email.
* **Authorization**: permessi definiti in Security/Permissions.cs:
  + events.manage, permissions.configure, roles.manage, subscribers.view, users.manage.
* **UI protette** (estratto):
  + /admin/events → policy events.manage.
  + /admin/events/edit\* → **[Authorize(Roles=“Admin”)]** (uso del ruolo Admin diretto) + operazioni di gestione.
  + /admin/events/{id}/subscribers → policy subscribers.view.
  + /admin/events/{eventId}/comments → policy events.manage.
  + /admin/rolepermissions e /supervisor/users → policy roles.manage.
  + /admin/users → policy users.manage.
* **Supervisore**: gestisce la **matrice permessi** (ruoli/claim) e può operare su tutte le aree.
* **Admin**: opera **entro i permessi/ruoli** concessi dal Supervisore (nota: in EventEdit.razor è richiesto il ruolo Admin).

—|—:|—:|—:| | Vedere/Ricercare eventi | ✓ | ✓ | ✓ | | Iscriversi/Disiscriversi | ✓ | ✓ | ✓ | | Commentare / Porre domanda | ✓ | ✓ | ✓ | | Rispondere domanda | | ✓ | ✓\* | | Moderare commenti/domande | | ✓ | ✓\* | | CRUD eventi | | ✓ | ✓\* | | Vedere/gestire iscritti + stats | | ✓ | ✓\* | | Gestire utenti/ruoli/matrice | | ✓ | | \* **in base ai permessi concessi dal Supervisore**

—|—:|—:|—:| | Vedere/Ricercare eventi | ✓ | ✓ | ✓ | | Iscriversi/Disiscriversi | ✓ | ✓ | ✓ | | Commentare | ✓ | ✓ | ✓ | | Porre domanda | ✓ | ✓ | ✓ | | Rispondere domanda | | ✓ | ✓ | | Moderare commenti/domande | | ✓ | ✓ | | CRUD eventi | | ✓ | ✓ | | Vedere/gestire iscritti + stats | | ✓ | ✓ | | Gestire utenti/ruoli/matrice | | ✓ | ✓ |

—|—:|—:|—:| | Vedere eventi | ✓ | ✓ | ✓ | | Iscriversi a evento | ✓ | ✓ | ✓ | | Commentare | ✓ | ✓ | ✓ | | Moderare commenti | | ✓ | ✓ | | CRUD eventi | | | ✓ | | Vedere/gestire iscritti | | ✓ | ✓ || Gestire utenti/ruoli | | | ✓ |

## 12. Configurazione & Operatività

* **DB**: SQLite (stringa conn. Data Source=eventmanagement.db).
* **Email**: SmtpEmailSender con SmtpOptions in appsettings.json (**segreti da NON committare**; usare Secret Manager/variabili ambiente in prod).
* **Migrazioni EF**: cartella Migrations.
* **Data Protection Keys**: cartella keys.
* **Hosting**: Blazor Server (router in Components/Routes.razor).

## 13. Piano di Test

* **Unit**: EventService (subscribe/unsubscribe, limiti capienza, duplication check), PermissionHandler.
* **Integration**: AppDbContext + migrazioni su DB test; invio SMTP con stub.
* **UI**: test di navigazione componenti principali (Home, EventDetails, flow registrazione/iscrizione).
* **Sicurezza**: verify route protection per ruoli.

## 14. Roadmap & Migliorie

* Tag/categorie eventi, ricerca full-text.
* Esportazione iscritti (CSV) da EventSubscribers.
* Log audit operazioni admin.
* Pagina reportistica base (numero iscritti/engagement) per Supervisor.

## 15. Allegati (da generare)

* Manuale utente breve (screens + flussi).
* Manuale operatore (Admin/Supervisor).
* Checklist di rilascio.

## 16. Domande aperte (per completare la documentazione)

1. **Campi evento**: immagini/banner, tag multipli e descrizione estesa? (Categoria confermata)
2. **Waitlist**: prevista lista d’attesa oltre capienza?
3. **Disiscrizione**: finestra temporale limite/penali? (flow confermato)
4. **Q&A**: approvazione preventiva o post-pubblicazione? notifica risposta via email?
5. **Ruoli**: esiste un ruolo “Organizer” per singolo evento che può rispondere alle domande del proprio evento?
6. **Permessi**: lista effettiva in Permissions.cs da mappare alla matrice (posso proporre un set di default).
7. **Notifiche**: template email e branding (logo/firma) + reminder calendario (.ics)?
8. **NFR**: target prestazioni, SLA, browser support, accessibilità e localizzazione.
9. **Ambienti**: DB prod, hosting, domini, backup/retention, gestione segreti.
10. **Privacy**: retention dati personali, consenso e informative.
11. **Analytics**: metriche (iscritti per evento, engagement Q&A, apertura email).
12. **API**: servono endpoint REST/integrazioni?
13. **Media**: storage per allegati/immagini evento?

## 17. Tracciabilità requisiti ↔ componenti (estratto)

* FR-2 (CRUD eventi) → Pages/Admin/EventsAdmin.razor, EventEdit.razor, EventService, AppDbContext, Models/Event.cs.
* FR-3 (Iscrizioni) → Pages/MySubscriptions.razor, EventDetails.razor, EventService, Models/Subscription.cs.
* FR-4 (Commenti) → Pages/Admin/EventComments.razor, CommentItem.razor, Models/Comment.cs.
* FR-5 (Utenti/permessi) → Pages/Admin/UserManagement.razor, RolePermissions.razor, Security/\*.
* FR-6 (Email) → Services/SmtpEmailSender.cs, SmtpOptions.cs, Account/\* per conferme.

## 18. Istruzioni per generare i diagrammi

Tutti i blocchi plantuml in questo documento possono essere copiati in un renderer PlantUML (o estensioni VS Code) per ottenere i diagrammi. Se preferisci, posso esportarli in PNG/SVG e allegarli alla documentazione definitiva.

*Fine della bozza 0.1.*