# Due progetti semplici (5 entita) - Entita, relazioni, endpoint

Documento rapido per ripetizione/portfolio. Tutto pensato per un backend REST + DB relazionale (MySQL/PostgreSQL/SQL Server).

## Progetto A - HelpDesk Simple

Sistema base di ticket di assistenza: utenti aprono ticket, assegnano categoria/priorita, e commentano.

### Entita

| Entita | Campi principali (esempio) |
|---|---|
| User | id (uuid/int)<br>name<br>email (unique)<br>createdAt |
| TicketCategory | id<br>name (unique) |
| Priority | id<br>level (LOW\|MEDIUM\|HIGH)<br>rank (1..3) |
| Ticket | id<br>title<br>description<br>status (OPEN\|IN_PROGRESS\|CLOSED)<br>createdAt<br>updatedAt<br>userId (FK->User)<br>categoryId (FK->TicketCategory)<br>priorityId (FK->Priority) |
| TicketComment | id<br>message<br>createdAt<br>ticketId (FK->Ticket)<br>userId (FK->User) |

### Relazioni (testuale)

- User 1 - N Ticket (un utente può aprire più ticket)
- TicketCategory 1 - N Ticket
- Priority 1 - N Ticket
- Ticket 1 - N TicketComment

- User 1 - N TicketComment (autore del commento)

### Endpoint REST (CRUD + filtri base)

| Metodo | Path | Descrizione | Input | Output |
|---|---|---|---|---|
| **GET** | `/api/users` | Lista utenti (paginata) | query: page,size,search? | 200: User[] |
| **GET** | `/api/users/{id}` | Dettaglio utente | - | 200: User |
| **POST** | `/api/users` | Crea utente | body: {name,email} | 201: User |
| **PUT** | `/api/users/{id}` | Aggiorna utente | body: {name?,email?} | 200: User |
| **DELETE** | `/api/users/{id}` | Elimina utente | - | 204 |
| **GET** | `/api/categories` | Lista categorie ticket | - | 200: TicketCategory[] |
| **POST** | `/api/categories` | Crea categoria | body: {name} | 201 |
| **PUT** | `/api/categories/{id}` | Aggiorna categoria | body: {name} | 200 |
| **DELETE** | `/api/categories/{id}` | Elimina categoria | - | 204 |
| **GET** | `/api/priorities` | Lista priorità | - | 200: Priority[] |
| **PUT** | `/api/priorities/{id}` | Aggiorna priorità (opz.) | body: {level,rank} | 200 |
| **GET** | `/api/tickets` | Lista ticket (filtri) | query: page,size,status?,userId?,categoryId?,priorityId?,q? | 200: Ticket[] |
| **GET** | `/api/tickets/{id}` | Dettaglio ticket | - | 200: Ticket (con comments? opz.) |
| **POST** | `/api/tickets` | Crea ticket | body: {title,description,userId,categoryId,priorityId} | 201: Ticket |
| **PUT** | `/api/tickets/{id}` | Aggiorna ticket | body: {title?,description?,status?,categoryId?,priorityId?} | 200 |
| **DELETE** | `/api/tickets/{id}` | Elimina ticket | - | 204 |
| **GET** | `/api/tickets/{id}/comments` | Lista commenti del ticket | - | 200: TicketComment[] |

| Metodo | Path | Descrizione | Input | Output |
|---|---|---|---|---|
| **POST** | `/api/tickets/{id}/comments` | Aggiunge commento | body: {userId,message} | 201: TicketComment |
| **DELETE** | `/api/comments/{id}` | Elimina commento | - | 204 |

# Progetto B - EventPlanner Basic

Sistema base di gestione eventi: organizer crea eventi in una location; partecipanti si iscrivono tramite registrazioni.

## Entita

| Entita | Campi principali (esempio) |
|---|---|
| Organizer | id<br>name<br>email (unique)<br>phone?<br>createdAt |
| Location | id<br>name<br>address<br>city<br>capacity<br>notes? |
| Event | id<br>title<br>description?<br>startAt<br>endAt<br>status (DRAFT\|PUBLISHED\|CANCELLED)<br>organizerId (FK->Organizer)<br>locationId (FK->Location) |
| Participant | id<br>fullName<br>email (unique)<br>phone? |
| Registration | id<br>eventId (FK->Event)<br>participantId (FK->Participant)<br>registeredAt<br>attendance (PENDING\|CONFIRMED\|NO_SHOW) |

## Relazioni (testuale)

- Organizer 1 - N Event
- Location 1 - N Event (un luogo può ospitare più eventi in date diverse)
- Event N - N Participant (risolta con Registration)
- Event 1 - N Registration
- Participant 1 - N Registration

## Endpoint REST (CRUD + filtri base)

| Metodo | Path | Descrizione | Input | Output |
|---|---|---|---|---|
| **GET** | `/api/organizers` | Lista organizer | query: page,size,search? | 200: Organizer[] |
| **POST** | `/api/organizers` | Crea organizer | body: {name,email,phone?} | 201 |
| **GET** | `/api/organizers/{id}` | Dettaglio organizer | - | 200 |
| **PUT** | `/api/organizers/{id}` | Aggiorna organizer | body: {name?,email?,phone?} | 200 |
| **DELETE** | `/api/organizers/{id}` | Elimina organizer | - | 204 |
| **GET** | `/api/locations` | Lista location | query: city?,minCapacity? | 200: Location[] |
| **POST** | `/api/locations` | Crea location | body: {name,address,city,capacity,notes?} | 201 |
| **PUT** | `/api/locations/{id}` | Aggiorna location | body: {..} | 200 |
| **DELETE** | `/api/locations/{id}` | Elimina location | - | 204 |
| **GET** | `/api/events` | Lista eventi (filtri) | query: page,size,status?,from?,to?,organizerId?,locationId?,q? | 200: Event[] |
| **GET** | `/api/events/{id}` | Dettaglio evento | - | 200: Event |
| **POST** | `/api/events` | Crea evento | body: {title,description?,startAt,endAt,organizerId,locationId} | 201 |
| **PUT** | `/api/events/{id}` | Aggiorna evento | body: {title?,description?,startAt?,endAt?,status?,locationId?} | 200 |
| **DELETE** | `/api/events/{id}` | Elimina evento | - | 204 |
| **GET** | `/api/participants` | Lista partecipanti | query: page,size,search? | 200: Participant[] |
| **POST** | `/api/participants` | Crea partecipante | body: {fullName,email,phone?} | 201 |
| **PUT** | `/api/participants/{id}` | Aggiorna partecipante | body: {..} | 200 |
| **DELETE** | `/api/participants/{id}` | Elimina partecipante | - | 204 |
| **GET** | `/api/events/{id}/registrations` | Iscrizioni di un evento | - | 200: Registration[] |

| Metodo | Path | Descrizione | Input | Output |
|---|---|---|---|---|
| **POST** | `/api/events/{id}/registrations` | Iscrive un partecipante | body: {participantId} | 201 |
| **PUT** | `/api/registrations/{id}` | Aggiorna stato presenza | body: {attendance} | 200 |
| **DELETE** | `/api/registrations/{id}` | Rimuove iscrizione | - | 204 |