**Backend**

* **Node.js + Express per API REST.**
* **Auth: JWT access (breve) in header + refresh in cookie httpOnly (SameSite=Strict, Secure in prod).**
* **Ruoli: USER/ADMIN + controllo ownership (solo owner o admin modifica).**
* **Password: bcrypt (12 round prod / 10 dev) + *pepper* opzionale da .env.**
* **DB: PostgreSQL via Prisma (migrazioni, query parametrizzate).**
* **Geo: PostGIS su Event.location (geography(Point,4326) + indice GIST).**
* **Realtime: Socket.IO (1 istanza; niente Redis).**
* **Upload: S3 con presigned URL (browser carica diretto).**
* **Docs: Swagger/OpenAPI su /api-docs.**
* **Sicurezza: helmet, cors (whitelist), express-rate-limit, hpp, xss-clean, express-validator.**
* **Logging: Pino + pino-http (redaction campi sensibili, requestId).**

**Frontend**

* **React + Vite con SASS, Bootstrap, FontAwesome.**
* **Validazione form: react-hook-form + zod.**
* **Dati: axios + React Query.**
* **Realtime: socket.io-client.**
* **Sicurezza UI: niente HTML in chat; se serve, DOMPurify.**

**Docker (dev & prod)**

* **Dev (docker-compose.dev.yml)**
  + **postgres (PostGIS) + pgadmin**
  + **server (Node 20, nodemon, mount del codice)**
  + **client (Vite dev server, mount del codice)**
  + **profili: db (solo DB) e app (server+client). Così puoi avviare prima il DB.**
* **Prod (docker-compose.prod.yml)**
  + **server immagine multi-stage (Prisma generate + migrate deploy)**
  + **client build statico e Nginx che serve il frontend e proxy /api al backend**
  + **postgres (per demo; in produzione vero meglio DB gestito)**

**Scaffolding**

eventhub/

├─ .gitignore

├─ .nvmrc

├─ docker-compose.yml # Postgres+PostGIS + pgAdmin

├─ package.json # script root (server+client)

├─ .env.example # (opzionale root)

├─ README-eng.md

├─ README-ita.md

│

├─ server/

│ ├─ package.json

│ ├─ .env.example

│ ├─ prisma/

│ │ ├─ schema.prisma # modelli Prisma (User, Event, ecc.)

│ │ └─ migrations/ # generate da Prisma (incl. PostGIS)

│ └─ src/

│ ├─ app.js # Express + middlewares

│ ├─ server.js # avvio HTTP + Socket.IO

│ ├─ config/

│ │ ├─ index.js # lettura env + validazioni

│ │ ├─ logger.js # Pino logger

│ │ └─ cors.js # policy CORS

│ ├─ load/

│ │ ├─ express.js # registra middlewares sicuri

│ │ ├─ routes.js # monta le rotte

│ │ └─ sockets.js # bootstrap Socket.IO

│ ├─ routes/

│ │ ├─ index.js

│ │ ├─ auth.routes.js

│ │ ├─ users.routes.js

│ │ ├─ events.routes.js

│ │ ├─ registrations.routes.js

│ │ ├─ chat.routes.js

│ │ └─ admin.routes.js

│ ├─ controllers/

│ │ ├─ auth.controller.js

│ │ ├─ users.controller.js

│ │ ├─ events.controller.js

│ │ ├─ registrations.controller.js

│ │ ├─ chat.controller.js

│ │ └─ admin.controller.js

│ ├─ services/

│ │ ├─ auth.service.js # bcrypt, JWT, refresh rotation

│ │ ├─ token.service.js

│ │ ├─ event.service.js

│ │ ├─ s3.service.js # presigned URL

│ │ └─ notification.service.js

│ ├─ middlewares/

│ │ ├─ auth.middleware.js

│ │ ├─ roles.middleware.js

│ │ ├─ validate.middleware.js

│ │ ├─ error.middleware.js

│ │ └─ rateLimit.middleware.js

│ ├─ validators/

│ │ ├─ auth.validators.js

│ │ ├─ user.validators.js

│ │ ├─ event.validators.js

│ │ └─ registration.validators.js

│ ├─ sockets/

│ │ ├─ index.js

│ │ ├─ chat.socket.js

│ │ └─ notifications.socket.js

│ ├─ docs/

│ │ ├─ swagger.js

│ │ └─ schemas/ # component schemas OpenAPI

│ ├─ utils/

│ │ ├─ ApiError.js

│ │ ├─ asyncHandler.js

│ │ └─ constants.js

│ └─ uploads/ # usato solo se fai upload via server (gitignored)

│

└─ client/

├─ package.json

├─ index.html

└─ src/

├─ main.jsx

├─ App.jsx

├─ api/

│ ├─ http.js

│ ├─ auth.api.js

│ ├─ events.api.js

│ ├─ users.api.js

│ └─ chat.api.js

├─ components/

│ ├─ Navbar.jsx

│ ├─ ProtectedRoute.jsx

│ ├─ EventCard.jsx

│ ├─ EventFilters.jsx

│ ├─ ChatBox.jsx

│ └─ FormInput.jsx

├─ pages/

│ ├─ Home.jsx

│ ├─ Login.jsx

│ ├─ Register.jsx

│ ├─ Dashboard.jsx

│ ├─ EventDetail.jsx

│ ├─ CreateEvent.jsx

│ └─ AdminPanel.jsx

├─ store/

│ ├─ auth.store.js

│ └─ ui.store.js

├─ hooks/

│ ├─ useAuth.js

│ └─ useSocket.js

├─ styles/ # SASS

│ ├─ \_variables.scss

│ ├─ \_mixins.scss

│ ├─ \_bootstrap.scss # import bootstrap SCSS + overrides

│ └─ main.scss # entry SASS

└─ utils/

└─ sanitize.js