STRATIKEY – Specifiche MVP e Architettura

1) Obiettivo del prodotto

Piattaforma multi-tenant B2B per gestire **marketing** e **commerciale** in modo autonomo, con **marketplace di servizi** (brand identity, stampa, video/foto, ecc.). Ogni azienda cliente ha il proprio spazio isolato (tenant) e può invitare utenti con ruoli e permessi granulari.

---

2) Ruoli e accessi (RBAC)

Ruoli globali (vendor): - **Super Admin (Vendor)**: visione e controllo di tutti i tenant; gestione fatturazione, piani, messaggi broadcast, ticketing, moderazione marketplace, analytics piattaforma.

Ruoli per tenant (azienda cliente): - **Org Admin**: amministratore dell'azienda cliente. Gestisce utenti, permessi, catalogo privato, integrazioni, pagamenti, visibilità dati dell'azienda. - **Marketing Manager**: accesso solo alla sezione **Marketing**. Pieno controllo su campagne, attività, asset, dati marketing. - **Commerciale (Sales Rep)**: accesso solo alla sezione **Commerciale**. Pieno controllo su pipeline, lead, opportunità, offerte. - **Viewer/Analyst (opzionale)**: sola lettura su "Panoramica" e "Dati".

Permessi chiave (per modulo): - *Marketing*: campaign.read/write, activity.read/write, assets.read/write, audiences.read/write, analytics.read - *Commerciale*: pipeline.read/write, lead.read/write, deal.read/write, tasks.read/write, reports.read - *Marketplace*: catalog.read, order.create, order.read (Org Admin può anche approvare/preferiti, gestire budget) - *Impostazioni*: users.invite, users.revoke, roles.assign, integrations.manage, billing.manage (solo Org Admin)

---

3) Struttura dell'app (IA/Navigation)

Sezioni principali per ogni tenant: 0. **Obiettivi aziendali** (nuova sezione iniziale) - … [testo invariato per Obiettivi aziendali] …

1. **Marketing**
2. Panoramica / Attività / Dati (viste contestuali che filtrano il Task Manager)
3. **Commerciale**
4. Panoramica / Attività / Dati (viste contestuali che filtrano il Task Manager)
5. **Attività (Task Manager globale)**
6. **Vista Lista**: simile a ClickUp, colonne personalizzabili (titolo, priorità, stato, assegnatario, scadenza, campagna/opportunità collegata).
7. **Vista Kanban**: per stato (Backlog, In corso, In review, Approvato, Fatto); drag&drop, WIP limits opzionali.

8. **Vista Calendario**: unico calendario per Marketing+CRM+Marketplace tasks, filtri per modulo (Marketing/CRM/Marketplace), tipo, priorità. Drag&drop con update scadenza.
9. **Vista Timeline/Gantt**: relazioni e dipendenze tra task, auto-layout orizzontale.
10. **Vista Tabella**: esportabile CSV; ordinamenti e filtri avanzati.
11. **Priorità**: P0..P3 con badge e colore; ordinamenti e SLA.
12. **Task linking**: sotto-task, dipendenze, relazione cross-modulo (es. task Marketing legato a Opportunità CRM).
13. **Assegnazioni multiple**: owner primario + collaborator.
14. **Commenti & thread**: stile ClickUp/Asana, mention, allegati, emoji.
15. **Automation base**: regole “se stato=Approvato → crea subtask Pubblicazione”, “se scadenza < oggi e status ≠ fatto → notifica P0 escalation”.
16. **AI Assist**: suggerimenti di breaking down, stima tempi, check list.
17. **Marketplace Servizi**
18. Catalogo servizi … [come prima]
19. **Chat & Notifiche**
20. … [come prima]
21. **Impostazioni (tenant)**
22. … [come prima]
23. Utenti & permessi, piani & fatturazione, metodi pagamento, integrazioni (Google, Meta, LinkedIn, Gmail/SMTP), domini/branding.

Area Vendor (Super Admin): - Dashboard globale, gestione piani e fatture, gestione catalogo marketplace, AML/KYC venditori esterni (se previsto), broadcast messaggi/promo, ticketing.

---

4) Multitenancy e sicurezza

- **Isolamento dati** via `organization_id` su tutte le tabelle tenant-scoped.
- **Row-Level Security** (se Postgres con RLS) o middleware RBAC a livello applicativo.
- **Audit log** per azioni sensibili (crea/modifica/elimina, login, ruoli).
- **GDPR**: consenso email/marketing, esportazione/cancellazione dati su richiesta, data processing agreement.

---

5) Stack consigliato (Replit-friendly)

- **Full-stack**: **Next.js 15 (App Router)**
- **UI**: React + Tailwind + shadcn/ui, icone lucide (lucide-react)
- **API**: Next.js Route Handlers (REST) + Zod per validazione
- **DB**: PostgreSQL (Neon/Supabase). ORM: **Prisma**
- **Auth**: **Auth.js** (email/password + OAuth Google), supporto multi-tenant con memberships
- **Payments**: **Stripe** (piani, one-off ordini marketplace)
- **File Upload**: UploadThing (asset, allegati) o storage S3-compatible

- **E-mail**: Resend/SMTP
- **Background jobs**: Inngest/Cloud Tasks o cron Next.js (MVP: code-based cron)
- **Telemetry**: Sentry + simple analytics (PostHog opzionale)

---

6) Modello dati (schema iniziale)

Core multi-tenant - … (come sopra)

Obiettivi aziendali / Marketing / CRM / Marketplace - … (come sopra)

Comunicazione & Notifiche / Supporto - `Thread`, `Message`, `Notification` (come sopra) - `Conversation` (id, organization_id, user_id, assignee_vendor_user_id?, status[open,pending,closed], priority[P0..P3], channel[widget,email], assigned_at?, closed_at?, last_message_at, created_at) - `ConversationMessage` (id, conversation_id, sender_user_id?, sender_role[user,agent,system], body, attachment_url?, created_at) - `AgentPresence` (id, vendor_user_id, status[online,away,offline], last_seen_at, capacity INT) - `EscalationRule` (id, organization_id?, condition JSON, action JSON) - `Feedback` (id, organization_id, user_id, rating[1..5], title, body, tags[], attachment_url?, status[new,triaged,planned,done,declined], created_at) - `SupportArticle` (id, category, title, slug, body, keywords[], is_published, updated_at) - `FAQCategory` (id, name, order)

---

7) API (Route Handlers) – v1 (selezione)

Supporto & Feedback - `POST /api/feedback` | `GET /api/feedback?status=&tags=` | `PATCH /api/feedback/:id`

FAQ / KB - `GET /api/faq/categories` | `GET /api/faq/articles?category=` | `GET /api/faq/articles/:slug`

Conversazioni (widget/email) - `POST /api/conversations` — crea conversazione (channel=widget|email) - `GET /api/conversations?status=&org=&assignee=` — lista con filtri - `POST /api/conversations/:id/messages` — aggiunge messaggio - `PATCH /api/conversations/:id` — cambia `status|assignee|priority` - `POST /api/conversations/:id/accept` — operatore prende in carico → `open` - `POST /api/conversations/:id/transfer` — trasferisce a altro operatore - `POST /api/conversations/:id/close?sendTranscript=true` — chiusura + invio trascrizione email - `POST /api/conversations/:id/escalate` — applica `EscalationRule`

Operatori - `GET /api/agents/presence` | `PATCH /api/agents/presence` — gestisce stato (online/away/offline) e capacità

---

8) Prisma – Schema iniziale (v1)

Copia/incolla in `prisma/schema.prisma`. Include le entità chiave per partire con MVP e le ultime aggiunte (chat/escalation, feedback, asset, social, marketplace). Alcune relazioni opzionali sono lasciate come `?` per flessibilità iniziale.

```
// prisma/schema.prisma

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

enum Role { ORG_ADMIN MARKETER SALES VIEWER }
enum Priority { P0 P1 P2 P3 }
enum CampaignStatus { DRAFT ACTIVE PAUSED COMPLETED }
enum CampaignType { ORGANICO ADV MIXED }
enum TaskStatus { BACKLOG IN_PROGRESS IN_REVIEW APPROVED DONE }
enum AssetType { IMAGE VIDEO DOC ARCHIVE }
enum ConversationStatus { OPEN PENDING CLOSED }
enum Channel { WIDGET EMAIL }
enum AgentPresenceStatus { ONLINE AWAY OFFLINE }
enum OrderStatus { REQUESTED CONFIRMED IN_PROGRESS DELIVERED CLOSED }
\ n// ---------------- Core ----------------
model Organization {
  id            String        @id @default(cuid())
  name          String
  plan          String?
  billingCustomerId String?
  createdAt     DateTime      @default(now())
  memberships   Membership[]
  users         User[]        @relation("OrgUsers")
  // settings & related
  settings      OrganizationSettings?
  domains       OrgDomain[]
  branding      BrandingTheme?
  subscription  Subscription?
  invoices      Invoice[]
  // modules
  campaigns     Campaign[]
  leads         Lead[]
  opportunities Opportunity[]
  assets        Asset[]
  audiences     Audience[]
  orders        OrgServiceOrder[]
  conversations Conversation[]
  feedback      Feedback[]
}

model User {
  id         String       @id @default(cuid())
  email      String       @unique
```

```
  name       String?
  image      String?
  createdAt  DateTime     @default(now())
  memberships Membership[]
  orgs       Organization[] @relation("OrgUsers")
  settings   UserSettings?
}

model Membership {
  id              String        @id @default(cuid())
  role            Role
  organizationId  String
  userId          String
  organization    Organization  @relation(fields: [organizationId],
references: [id])
  user            User          @relation(fields: [userId], references: [id])
  @@unique([organizationId, userId])
}

model AuditLog {
  id             String   @id @default(cuid())
  organizationId String
  userId         String?
  action         String
  entity         String
  entityId       String?
  metadata       Json?
  createdAt      DateTime @default(now())
}

// -------------- Marketing ----------------
model Campaign {
  id              String         @id @default(cuid())
  organizationId  String
  name            String
  type            CampaignType   @default(ORGANICO)
  status          CampaignStatus @default(DRAFT)
  objective       String?
  budget          Float?
  startAt         DateTime?
  endAt           DateTime?
  priority        Priority?      @default(P2)
  organization    Organization   @relation(fields: [organizationId],
references: [id])
  channels        CampaignChannel[]
  tasks           MarketingTask[]
  assetsLinks     AssetLink[]
  metrics         MarketingMetric[]
  content         ContentItem[]
  createdAt       DateTime       @default(now())
}
```

```
model CampaignChannel {
  id          String    @id @default(cuid())
  campaignId  String
  channel     String
  settings    Json?
  campaign    Campaign  @relation(fields: [campaignId], references: [id])
}

model MarketingTask {
  id              String     @id @default(cuid())
  organizationId  String
  campaignId      String?
  title           String
  type            String
  subtype         String?
  assigneeId      String?
  status          TaskStatus @default(BACKLOG)
  priority        Priority   @default(P2)
  dueAt           DateTime?
  organization    Organization @relation(fields: [organizationId],
references: [id])
  campaign        Campaign?  @relation(fields: [campaignId], references:
[id])
  createdAt       DateTime   @default(now())
}

model ContentItem {
  id              String    @id @default(cuid())
  organizationId  String
  kind            String    // post, ad_creative, email
  usage           String    // organico, adv, entrambi
  title           String?
  body            String?
  channel         String?
  scheduledAt     DateTime?
  publishedAt     DateTime?
  status          String?
  assetIds        Json?
  organization    Organization @relation(fields: [organizationId],
references: [id])
  createdAt       DateTime   @default(now())
}

model Audience {
  id              String   @id @default(cuid())
  organizationId  String
  name            String
  criteria        Json?
  sizeEstimate    Int?
  organization    Organization @relation(fields: [organizationId],
```

```
references: [id])
}

model MarketingMetric {
  id          String   @id @default(cuid())
  campaignId  String
  date        DateTime
  impressions Int?
  clicks      Int?
  ctr         Float?
  conversions Int?
  spend       Float?
  revenue     Float?
  campaign    Campaign @relation(fields: [campaignId], references: [id])
  @@index([campaignId, date])
}

// -------------- Social & Assets --------------
model SocialConnection {
  id             String   @id @default(cuid())
  organizationId String?
  userId         String?
  provider       String   // meta, linkedin, tiktok, x
  accountId      String
  accountType    String   // page, business, member
  displayName    String?
  scopes         Json?
  accessTokenEnc String
  refreshTokenEnc String?
  expiresAt      DateTime?
  status         String   // active, expired, revoked
  createdAt      DateTime @default(now())
}

model SocialPost {
  id              String   @id @default(cuid())
  organizationId  String
  createdByUserId String
  usage           String   // organico, adv
  channel         String   // fb, ig, li, tt, x
  connectionId    String
  content         String?
  assetIds        Json?
  scheduledAt     DateTime?
  publishedAt     DateTime?
  status          String   @default("draft") // draft, scheduled, publishing,
published, failed
  externalPostId  String?
  error           String?
  createdAt       DateTime @default(now())
}
```

```
model Asset {
  id              String    @id @default(cuid())
  organizationId  String
  type            AssetType
  mimeType        String
  sizeBytes       Int
  width           Int?
  height          Int?
  durationSec     Int?
  checksumSha256  String
  url             String
  thumbUrl        String?
  title           String?
  tags            String[]
  folder          String?
  rights          Json?
  ownerId         String?
  version         Int       @default(1)
  createdAt       DateTime  @default(now())
  organization    Organization @relation(fields: [organizationId],
references: [id])
  links           AssetLink[]
  @@index([organizationId])
}

model AssetLink {
  id              String   @id @default(cuid())
  organizationId  String
  assetId         String
  relatedType     String   // campaign, content, social_post, lead,
opportunity
  relatedId       String
  createdAt       DateTime @default(now())
  asset           Asset    @relation(fields: [assetId], references: [id])
}

model TranscodeJob {
  id              String   @id @default(cuid())
  organizationId  String
  assetId         String
  jobType         String   // image_opt, video_transcode, thumbnail
  status          String   // pending, processing, done, failed
  provider        String   // local, ffmpeg, cloud
  log             Json?
  createdAt       DateTime @default(now())
  updatedAt       DateTime @updatedAt
}

// ---------------- CRM ----------------
model Account {
```

```
  id              String   @id @default(cuid())
  organizationId  String
  name            String
  domain          String?
  industry        String?
  size            String?
  territory       String?
  ownerId         String?
  rating          String?
  createdAt       DateTime @default(now())
}

model Contact {
  id              String   @id @default(cuid())
  organizationId  String
  accountId       String?
  firstName       String?
  lastName        String?
  email           String?
  phone           String?
  title           String?
  ownerId         String?
  consentMarketing Boolean  @default(false)
  createdAt       DateTime @default(now())
}

model Lead {
  id              String   @id @default(cuid())
  organizationId  String
  source          String?
  firstName       String?
  lastName        String?
  email           String?
  phone           String?
  company         String?
  ownerId         String?
  status          String    @default("new")
  priority        Priority  @default(P2)
  createdAt       DateTime  @default(now())
}

model Opportunity {
  id              String   @id @default(cuid())
  organizationId  String
  accountId       String?
  contactId       String?
  leadId          String?
  title           String
  stage           String
  amount          Float?
  currency        String?
```

```
  closeDate       DateTime?
  ownerId         String?
  probability     Int?
  priority        Priority @default(P2)
  createdAt       DateTime @default(now())
}

model PipelineStage {
  id              String  @id @default(cuid())
  organizationId  String
  name            String
  order           Int
  winProbability  Int?
}

model ActivityLog {
  id          String   @id @default(cuid())
  organizationId String
  type        String   // call,email,meeting,task
  subject     String?
  notes       String?
  outcome     String?
  durationSec Int?
  relatedType String   // lead,contact,account,opportunity
  relatedId   String
  dueAt       DateTime?
  completedAt DateTime?
  ownerId     String?
}

model Sequence {
  id              String  @id @default(cuid())
  organizationId  String
  name            String
  description     String?
  isActive        Boolean @default(true)
  steps           SequenceStep[]
}

model SequenceStep {
  id          String  @id @default(cuid())
  sequenceId  String
  order       Int
  stepType    String  // email,call,task,linkedin
  waitDays    Int     @default(1)
  templateSubject String?
  templateBody    String?
  branching   Json?
  sequence    Sequence @relation(fields: [sequenceId], references: [id])
}
```

```
model SequenceEnrollment {
  id              String  @id @default(cuid())
  organizationId  String
  sequenceId      String
  leadId          String?
  contactId       String?
  ownerId         String?
  status          String  @default("active")
  currentStep     Int     @default(1)
  startedAt       DateTime @default(now())
  lastActionAt    DateTime?
}

model Product {
  id              String  @id @default(cuid())
  organizationId  String
  name            String
  sku             String?
  price           Float
  currency        String @default("EUR")
}

model Quote {
  id              String  @id @default(cuid())
  organizationId  String
  opportunityId   String
  subtotal        Float
  discount        Float?
  total           Float
  currency        String @default("EUR")
  pdfUrl          String?
}

// -------------- Marketplace --------------
model Service {
  id          String  @id @default(cuid())
  name        String
  description String?
  basePrice   Float
  category    String
  isActive    Boolean @default(true)
  options     ServiceOption[]
  packages    ServicePackage[]
}

model ServiceOption {
  id        String  @id @default(cuid())
  serviceId String
  name      String
  priceDelta Float  @default(0)
  service   Service @relation(fields: [serviceId], references: [id])
```

```
}

model ServicePackage {
  id        String  @id @default(cuid())
  serviceId String
  name      String
  tiers     String[] // S,M,L
  contents  Json?
  basePrice Float
  service   Service @relation(fields: [serviceId], references: [id])
}

model OrgServiceOrder {
  id              String  @id @default(cuid())
  organizationId  String
  serviceId       String
  options         Json?
  quantity        Int      @default(1)
  status          OrderStatus @default(REQUESTED)
  assigneeVendorUserId String?
  total           Float?
  stripePaymentIntentId String?
  createdAt       DateTime @default(now())
  messages        OrderMessage[]
  events          OrderEvent[]
}

model OrgServiceOrderItem {
  id        String  @id @default(cuid())
  orderId   String
  packageId String?
  optionId  String?
  name      String
  quantity  Int     @default(1)
  unitPrice Float   @default(0)
  total     Float   @default(0)
}

model OrderMessage {
  id        String  @id @default(cuid())
  orderId   String
  senderUserId String?
  body      String
  attachmentUrl String?
  createdAt DateTime @default(now())
}

model OrderEvent {
  id        String  @id @default(cuid())
  orderId   String
  type      String  // status_change,message,asset_uploaded
```

```
  payload   Json?
  createdAt DateTime @default(now())
}

// -------------- Supporto / Notifiche / Settings --------------
model Conversation {
  id              String  @id @default(cuid())
  organizationId  String
  userId          String?
  assigneeVendorUserId String?
  status          ConversationStatus @default(PENDING)
  priority        Priority @default(P2)
  channel         Channel  @default(WIDGET)
  assignedAt      DateTime?
  closedAt        DateTime?
  lastMessageAt   DateTime?
  createdAt       DateTime @default(now())
  messages        ConversationMessage[]
}

model ConversationMessage {
  id           String @id @default(cuid())
  conversationId String
  senderUserId String?
  senderRole   String  // user, agent, system
  body         String
  attachmentUrl String?
  createdAt    DateTime @default(now())
}

model AgentPresence {
  id           String @id @default(cuid())
  vendorUserId String
  status       AgentPresenceStatus @default(ONLINE)
  lastSeenAt   DateTime @default(now())
  capacity     Int      @default(3)
}

model EscalationRule {
  id              String @id @default(cuid())
  organizationId  String?
  condition       Json
  action          Json
}

model Feedback {
  id              String  @id @default(cuid())
  organizationId  String
  userId          String
  rating          Int
  title           String
```

```
  body            String
  tags            String[]
  attachmentUrl   String?
  status          String  @default("new")
  createdAt       DateTime @default(now())
}

model SupportArticle {
  id          String  @id @default(cuid())
  category    String
  title       String
  slug        String  @unique
  body        String
  keywords    String[]
  isPublished Boolean @default(true)
  updatedAt   DateTime @updatedAt
}

model FAQCategory {
  id    String @id @default(cuid())
  name  String
  order Int    @default(0)
}

model OrganizationSettings {
  id               String  @id @default(cuid())
  organizationId   String  @unique
  branding         Json?
  quotas           Json?
  approvalPolicy   Json?
  defaults         Json?
}

model UserSettings {
  id          String  @id @default(cuid())
  userId      String  @unique
  locale      String? @default("it-IT")
  timezone    String? @default("Europe/Rome")
  theme       String? @default("dark")
  notificationPrefs Json?
}

model NotificationPreference {
  id        String @id @default(cuid())
  userId    String
  category  String
  channel   String  // inapp,email
  enabled   Boolean @default(true)
  quietHours Json?
}
```

```
model ApiKey {
  id             String @id @default(cuid())
  organizationId String?
  userId         String?
  name           String
  hashedKey      String
  lastUsedAt     DateTime?
  createdAt      DateTime @default(now())
  revokedAt      DateTime?
}

model WebhookEndpoint {
  id             String @id @default(cuid())
  organizationId String
  url            String
  secret         String
  isActive       Boolean @default(true)
  createdAt      DateTime @default(now())
  logs           WebhookLog[]
}

model WebhookLog {
  id           String @id @default(cuid())
  endpointId   String
  event        String
  payload      Json
  status       Int
  attempts     Int      @default(1)
  lastAttemptAt DateTime @default(now())
}

model Subscription {
  id                 String @id @default(cuid())
  organizationId     String @unique
  plan               String
  status             String
  periodStart        DateTime
  periodEnd          DateTime
  stripeCustomerId   String?
  stripeSubscriptionId String?
}

model Invoice {
  id                String @id @default(cuid())
  organizationId    String
  stripeInvoiceId   String @unique
  amount            Int
  currency          String @default("EUR")
  status            String
  hostedUrl         String?
  createdAt         DateTime @default(now())
```

```
}

model OrgDomain {
  id              String @id @default(cuid())
  organizationId  String
  hostname        String
  status          String @default("pending")
  createdAt       DateTime @default(now())
}

model BrandingTheme {
  id              String @id @default(cuid())
  organizationId  String @unique
  primary         String @default("#390035")
  secondary       String @default("#901d6b")
  accent          String @default("#901d6b")
  updatedAt       DateTime @updatedAt
}

model ApprovalPolicy {
  id              String @id @default(cuid())
  organizationId  String
  scope           String  // post,campaign,order,spend
  rules           Json
}

model EmailTemplate {
  id              String @id @default(cuid())
  organizationId  String
  key             String
  subject         String
  body            String
  isActive        Boolean @default(true)
}

model SupportTicket {
  id              String @id @default(cuid())
  organizationId  String
  userId          String
  subject         String
  body            String
  status          String @default("open")
  priority        Priority @default(P2)
  createdAt       DateTime @default(now())
  updatedAt       DateTime @updatedAt
}
```

Comandi iniziali

```
# 1) installa prisma e client
npm i -D prisma
npm i @prisma/client

# 2) init + migrate
npx prisma init
npx prisma migrate dev -n init_stratikey

# 3) genera client
npx prisma generate
```
