Modelo de Domínio + Testes Integrados (AssistenteExecutivo)

Objetivo

- Derivar do PRD do **Assistente Executivo de IA** um **modelo de domínio detalhado (DDD)** e uma **estratégia de testes integrada**, priorizando **testabilidade no domínio**.
- Target stack: **C# / .NET 10** (VS2026), **web Next.js**, **app Flutter**.

Fonte de Requisitos

- Documento: `documents/Assistente_Executivo_IA_PRD_Completo.htm` (derivado do PRD original `.docx`).

Assunções (para destravar)

- Nome do produto/projeto/backend: **AssistenteExecutivo**.
- Backend: **modular monolith** com Clean Architecture (evolui para microserviços se necessário).
- Auth inicial via **JWT** (com refresh) e armazenamento em **PostgreSQL**.

Escopo funcional coberto (a partir do PRD)

MVP (Módulo 1)

- **Autenticação**:
  - `POST /auth/register`
  - `POST /auth/login`
  - `GET /auth/me`
- **CRM + Captura**:
  - Digitalização de cartão (OCR): `POST /contacts/upload-card`
  - Nota por áudio com transcrição + resumo + extração de tarefas: `POST /contacts/{id}/audio-note`
  - CRUD de contatos:
    - `GET /contacts`
    - `GET /contacts/{id}`
    - `PUT /contacts/{id}`
    - `DELETE /contacts/{id}`

Pós-MVP (Módulos 2–3)

- **Enriquecimento de perfil**: mapear redes (LinkedIn/X), consolidar histórico.
- **Árvore de relacionamentos**: conexões em comum, grafo.
- **KYC/Due Diligence**: varredura em fontes públicas e relatório.
- **Automação e ação estratégica**:
  - Estratégias de aproximação.
  - Lembretes inteligentes (nutrição).

    - Assistente de redação (emails/ofícios/invites) + aplicação de **papel timbrado** via template.
    - Sugestão de conteúdo para redes sociais.

Negócio

- **Assinatura (tiers)** + **créditos** para ações "custosas" de IA (transcrição longa, enriquecimento, KYC).

Bounded Contexts (DDD)

- **Identity**: contas, sessões, tokens.
- **CRM**: contatos, empresas, interações, tags, relacionamentos.
- **Capture**: ingestão de mídia (imagem/áudio), jobs, resultados, anexos.
- **Intelligence**: enriquecimento, grafo social, KYC.
- **Automation**: lembretes, drafts (email/ofício/invite), templates.
- **Billing**: planos, assinaturas, carteira de créditos, consumo.

Modelo de Domínio (detalhado)

Linguagem Ubíqua (termos do domínio)

- **Usuário**: pessoa que usa o sistema (executivo/gestor/etc.).
- **Contato**: pessoa/lead/cliente/parceiro cadastrado no CRM interno.
- **Empresa**: organização associada a um contato.
- **Interação**: registro estruturado de contato com o contato (nota, reunião, e-mail, ligação, tarefa, documento enviado).
- **Captura**: ato de transformar mídia (imagem/áudio) em dados estruturados no CRM.
- **Mídia**: arquivo (imagem do cartão, áudio) armazenado e referenciado.
- **Job**: processamento assíncrono (OCR, transcrição, enriquecimento, KYC).
- **Enriquecimento**: coleta/consolidação de informações públicas (ex: LinkedIn) para complementar o contato.
- **Grafo de relacionamentos**: conexões em comum e caminhos relevantes.
- **KYC/Due diligence**: verificação automatizada em fontes públicas para identificar riscos e "red flags".
- **Lembrete**: recomendação/alerta para reativar ou nutrir relacionamento (com motivo e timing).
- **Draft**: rascunho de e-mail/ofício/invite gerado com base em contexto e template.
- **Template**: modelo reutilizável (texto + placeholders) e/ou papel timbrado.
- **Plano (tier)**: assinatura com limites (contatos/armazenamento/recursos premium).
- **Créditos**: unidade interna consumida por operações intensivas (transcrição longa, perfil complexo, KYC).

Princípios de modelagem (para testabilidade)

- **Domínio determinístico**: sem dependência direta de tempo/IO; usar `IClock`, `IIdGenerator` e "ports" para provedores externos.
- **Eventos de domínio** como saída principal do agregado (para disparar jobs, notificações e integrações).
- **Regras explícitas**: invariantes em agregados, políticas em `*Policy`/`*Specification`, regras multi-agregado em `DomainServices`.

Value Objects (VOs)

- **Identity**
  - `UserId`, `EmailAddress`, `PasswordHash`, `PasswordPolicy` (VO/config).
  - `SessionId`, `RefreshToken`, `JwtSubject` (se necessário).
- **CRM**
  - `ContactId`, `PersonName`, `PhoneNumber`, `CompanyId`, `CompanyName`.
  - `Tag` (VO), `InteractionId`, `InteractionType` (enum), `InteractionSummary`.
- **Capture**
  - `MediaId`, `MediaRef` (storageKey + hash + mimeType + size), `AudioDuration`.
  - `JobId`, `JobStatus`, `JobError` (code + message).
  - `OcrExtract` (campos + confiança), `Transcript`, `TranscriptSegment`, `ExtractedTask`.
- **Intelligence**
  - `SocialProfileId`, `ExternalProfileRef`, `ProfileSnapshot` (VO imutável).
  - `GraphSnapshotId`, `ConnectionEdge`, `ConnectionPath`.
  - `ReportId`, `RiskScore`, `Finding` (tipo/fonte/gravidade).
- **Automation**
  - `ReminderId`, `ReminderReason`, `ReminderSchedule` (data/recorrência), `NotificationChannel`.
  - `DraftId`, `DocumentType`, `TemplateId`, `LetterheadId`.
- **Billing**
  - `PlanId`, `PlanLimits` (maxContacts, maxStorageBytes, featureFlags).
  - `SubscriptionId`, `BillingInterval`, `CreditAmount`, `CreditTransactionId`, `IdempotencyKey`.

Bounded Context: Identity

Agregados

- **UserAccount**
  - **Estado**: `Active | Suspended | Deleted` (soft delete).
  - **Campos**: `UserId`, `Email`, `PasswordHash`, `CreatedAt`, `LastLoginAt?`.
  - **Comportamentos**:
    - `Register(email, password)`
    - `ChangePassword(old, new)`
    - `RecordLogin(successAt)`
  - **Invariantes**:
    - email deve ser válido e único.
    - senha deve cumprir política.
- **Session**
  - **Campos**: `SessionId`, `UserId`, `RefreshTokenHash`, `CreatedAt`, `ExpiresAt`, `RevokedAt?`.
  - **Comportamentos**:
    - `Create(userId)`
    - `RotateRefreshToken()`
    - `Revoke(reason)`
  - **Invariantes**:
    - sessão revogada não pode ser usada/rotacionada.

Eventos de domínio (Identity)

- `UserRegistered(UserId, EmailAddress)`
- `UserLoggedIn(UserId, SessionId)`
- `SessionRevoked(SessionId, Reason)`

Bounded Context: CRM

Agregados

- **Contact**
  - **Campos**: `ContactId`, `OwnerUserId`, `Name`, `Emails[]`, `Phones[]`, `CompanyId?`, `Tags[]`, `CreatedAt`, `UpdatedAt`.
  - **Subentidades**:
    - `Interaction` (nota/reunião/documento/tarefa vinculada)
    - `AttachmentRef` (para mídia/documentos)
  - **Comportamentos**:
    - `CreateFromCardExtract(extract)`
    - `UpdateDetails(patch)`
    - `AddTextNote(text, createdAt)`
    - `AddAudioNote(mediaRef, transcript, summary, tasks[])`
    - `AddInteraction(interaction)`
    - `Delete()` (soft)
  - **Invariantes**:
    - ao menos 1 identificador de contato (email ou telefone) **quando criado via OCR** (se OCR não extrair, exigir revisão manual).
    - emails/telefones não podem duplicar dentro do contato.
    - `OwnerUserId` sempre definido.
- **Company**
  - **Campos**: `CompanyId`, `Name`, `Domains[]`, `Notes?`
  - **Regras**:
    - pode ser criada automaticamente a partir de OCR/enriquecimento, mas deve suportar merge.

Domain Services (CRM)

- **ContactDeduplicationService**
  - resolve “criar vs mesclar” com base em heurística (email/telefone domínio da empresa + similaridade).
  - produz decisão determinística (e testável) e emite `ContactMerged` quando aplicável.

Eventos de domínio (CRM)

- `ContactCreated(ContactId, OwnerUserId, Source)`
- `ContactUpdated(ContactId)`
- `ContactDeleted(ContactId)`
- `ContactMerged(SourceContactId, TargetContactId)`
- `InteractionAdded(ContactId, InteractionId, InteractionType)`

Bounded Context: Capture

Agregados

- **MediaAsset**
  - **Campos**: `MediaId`, `OwnerUserId`, `MediaRef`, `CreatedAt`, `Kind(Image|Audio)`, `Metadata`.
  - **Invariantes**:
    - hash obrigatório (para dedup e integridade).
    - tamanho máximo governado por `PlanLimits`.
- **CaptureJob**
  - **Tipos**: `CardScan`, `AudioNoteTranscription`.
  - **Campos**: `JobId`, `OwnerUserId`, `ContactId?`, `MediaId`, `Status`, `RequestedAt`, `CompletedAt?`, `Error?`.
  - **Resultados**:
    - `CardScanResult`: `OcrExtract`, `SuggestedContactPatch`, `ConfidenceScores`
    - `AudioResult`: `Transcript`, `Summary`, `ExtractedTasks[]`
  - **Comportamentos**:
    - `RequestCardScan(mediaId)`
    - `RequestAudioProcessing(contactId, mediaId)`
    - `MarkProcessing()`, `Complete(result)`, `Fail(error)`
  - **Invariantes**:
    - transições válidas de status.
    - job concluído não pode ser reprocessado (novo job para retry).

Eventos de domínio (Capture)

- `MediaUploaded(MediaId, Kind)`
- `CaptureJobRequested(JobId, Type)`
- `CaptureJobCompleted(JobId, Type)`
- `CaptureJobFailed(JobId, ErrorCode)`

Bounded Context: Intelligence

Agregados

- **ProfileEnrichmentJob**
  - **Inputs**: `ContactId`, email/nome (derivados).
  - **Outputs**: `ProfileSnapshotId`, `ExternalProfileRefs[]`, `EnrichmentSummary`.
  - **Regras**:
    - pode consumir créditos quando “perfil complexo” (configurável).
    - deve registrar fontes e timestamps (auditoria mínima).
- **RelationshipGraphSnapshot**
  - **Campos**: `SnapshotId`, `OwnerUserId`, `ContactId`, `Edges[]`, `ComputedAt`, `SourceRefs`.
  - **Regras**:
    - snapshot é imutável (nova versão em cada recomputação).
- **DueDiligenceReport**
  - **Campos**: `ReportId`, `ContactId?`, `CompanyId?`, `Findings[]`, `RiskScore`, `GeneratedAt`, `ValidUntil?`.
  - **Regras (PRD)**:

    - deve buscar processos judiciais, menções negativas na mídia e outros “red flags” em fontes públicas.
    - relatório deve referenciar fontes (URLs/ids de fonte).

Eventos de domínio (Intelligence)

- `EnrichmentRequested(ContactId, JobId)`
- `EnrichmentCompleted(ContactId, SnapshotId)`
- `DueDiligenceRequested(TargetType, TargetId, ReportId)`
- `DueDiligenceCompleted(ReportId, RiskScore)`

Bounded Context: Automation

Agregados

- **Reminder**
    - **Campos**: `ReminderId`, `OwnerUserId`, `ContactId`, `Reason`, `SuggestedMessage?`, `ScheduledFor`, `Status(Pending|Sent|Dismissed|Snoozed)`.
    - **Regras (PRD)**:
        - lembrete deve ter **motivo** (ex: aniversário da empresa, notícia relevante).
        - pode depender de integrações (notícias, calendário) — via ports.
- **DraftDocument**
    - **Tipos**: `Email`, `Oficio`, `Invite`.
    - **Campos**: `DraftId`, `OwnerUserId`, `ContactId?`, `CompanyId?`, `DocumentType`, `TemplateId?`, `LetterheadId?`, `Content`, `Status(Draft|Approved|Sent)`.
    - **Regras (PRD)**:
        - geração baseada em contexto do relacionamento + templates.
        - suportar aplicação de papel timbrado (template estilo Canva) como etapa de renderização.
- **Template**
    - **Campos**: `TemplateId`, `OwnerUserId`, `Name`, `PlaceholdersSchema`, `Body`, `Active`.
    - **Variações**:
        - `LetterheadTemplate` (branding/layout) separado do `TextTemplate` (conteúdo).

Eventos de domínio (Automation)

- `ReminderScheduled(ReminderId, ContactId, ScheduledFor)`
- `ReminderStatusChanged(ReminderId, NewStatus)`
- `DraftCreated(DraftId, DocumentType)`
- `DraftApproved(DraftId)`

Bounded Context: Billing

Agregados

- **Plan**
    - **Campos**: `PlanId`, `Name`, `Limits(contacts/storage/features)`, `IsActive`.
    - **Regras**:
        - limites dirigem validações em CRM/Capture (max contatos, max storage).

- **Subscription**
  - **Campos**: `SubscriptionId`, `OwnerUserId`, `PlanId`, `Interval(Monthly|Yearly)`, `Status(Active|PastDue|Canceled)`, `StartedAt`, `RenewsAt?`.
  - **Regras (PRD)**:
    - assinatura define acesso a funcionalidades premium + cota de créditos.
- **CreditWallet**
  - **Modelo**: ledger (imutável) + saldo derivado.
  - **Transações**: `Grant`, `Purchase`, `Reserve`, `Consume`, `Refund`, `Expire`.
  - **Invariantes**:
    - saldo nunca pode ficar negativo.
    - idempotência obrigatória para operações cobradas (`IdempotencyKey`).
  - **Regras (PRD)**:
    - transcrições longas, enriquecimento complexo e KYC consomem créditos.
    - usuário pode comprar pacotes adicionais.

Eventos de domínio (Billing)

- `SubscriptionActivated(SubscriptionId, PlanId)`
- `SubscriptionCanceled(SubscriptionId)`
- `CreditsGranted(OwnerUserId, Amount)`
- `CreditsReserved(OwnerUserId, Amount, Purpose)`
- `CreditsConsumed(OwnerUserId, Amount, Purpose)`
- `CreditsRefunded(OwnerUserId, Amount, Purpose)`

Regras transversais (Policies/Specifications)

- **PlanLimitPolicy**
  - `CanCreateContact(owner)` / `CanStoreMedia(owner, size)` / `HasFeature(owner, featureFlag)`.
- **CreditCostPolicy**
  - define custo por operação e parâmetros (ex: minutos de áudio, “complexidade” de perfil, tipo de KYC).
- **IdempotencyPolicy**
  - garante que endpoints “custosos” não cobrem duas vezes (mesma requisição).

Mapeamento MVP: Endpoints -> Casos de Uso -> Domínio

- `POST /auth/register`
  - **Use case**: `RegisterUser`
  - **Agregados**: `UserAccount`
  - **Eventos**: `UserRegistered`
- `POST /auth/login`
  - **Use case**: `LoginUser`
  - **Agregados**: `UserAccount`, `Session`
  - **Eventos**: `UserLoggedIn`
- `GET /auth/me`
  - **Use case**: `GetCurrentUser`
  - **Consulta** (read model)

- `POST /contacts/upload-card`
  - **Use case**: `UploadCardAndRequestOcr`
  - **Agregados**: `MediaAsset`, `CaptureJob`
  - **Eventos**: `MediaUploaded`, `CaptureJobRequested`
- `POST /contacts/{id}/audio-note`
  - **Use case**: `AddAudioNoteAndProcess`
  - **Agregados**: `MediaAsset`, `CaptureJob`, `Contact`, `CreditWallet` (se aplicável)
  - **Eventos**: `CaptureJobRequested`, `InteractionAdded`, `CreditsReserved/Consumed`
- `GET /contacts`, `GET /contacts/{id}`
  - **Use cases**: `ListContacts`, `GetContactById`
  - **Consulta** (read model)
- `PUT /contacts/{id}`
  - **Use case**: `UpdateContact`
  - **Agregado**: `Contact`
  - **Eventos**: `ContactUpdated`
- `DELETE /contacts/{id}`
  - **Use case**: `DeleteContact`
  - **Agregado**: `Contact`
  - **Eventos**: `ContactDeleted`

Arquitetura proposta (C# / .NET 10)

Estrutura de solução (monorepo)

- `backend/AssistenteExecutivo.sln`
- `backend/src/AssistenteExecutivo.Domain`
- `backend/src/AssistenteExecutivo.Application`
- `backend/src/AssistenteExecutivo.Infrastructure`
- `backend/src/AssistenteExecutivo.Api`
- `backend/tests/AssistenteExecutivo.Domain.Tests`
- `backend/tests/AssistenteExecutivo.Application.IntegrationTests`
- `backend/tests/AssistenteExecutivo.Api.ContractTests`
- `web/` (Next.js)
- `app/` (Flutter)

Decisões de testabilidade no domínio (regras)

- **Domínio sem IO**: sem EF, sem HttpClient, sem `DateTime.UtcNow` direto.
- Dependências externas apenas via **ports**:
  - `IOcrProvider`, `ISpeechToTextProvider`, `ILLMProvider`, `IKycProvider`, `INewsProvider`, `IFileStore`, `IClock`.
- **Determinismo**: injetar clock, ids via `IIdGenerator` onde necessário.
- Regras de negócio concentradas em:
  - métodos do agregado (invariantes)
  - domain services (regras que cruzam agregados)
  - policies/specifications (ex: elegibilidade a lembrete)

Estratégia de Testes (integrada)

1) Testes do Domínio (rápidos)

- xUnit + FluentAssertions.
- Cobrir:
  - invariantes de agregados (ex: contato não pode ter email inválido)
  - transições de estado de `CaptureJob`
  - ledger de créditos (idempotência, reserva/consumo/refund)
  - emissão de eventos de domínio

2) Testes de Integração (Application)

- Handler completo (CQRS): comando -> persistência -> outbox/eventos.
- Banco real com **Testcontainers PostgreSQL** (se Docker disponível).
- Stubs para providers externos (OCR/STT/LLM/KYC) com contratos estáveis.

3) Testes de Contrato (API)

- `Microsoft.AspNetCore.Mvc.Testing` (WebApplicationFactory).
- Validar endpoints do PRD e schemas.
- Geração de **OpenAPI** para consumo por Next.js/Flutter.

Diagramas (para guiar implementação)

Fluxo: captura de cartão (OCR)

```
sequenceDiagram
participant App as FlutterApp
participant Api as Api
participant Cmd as UploadCardCommand
participant Job as CardScanJob
participant Ocr as IOcrProvider
participant Crm as Contact
App->>Api: POST_/contacts/upload-card
Api->>Cmd: dispatch
Cmd->>Job: Create(Requested)
Cmd->>Ocr: ExtractFields(image)
Ocr-->>Cmd: ExtractedContactData
Cmd->>Crm: CreateOrUpdate
Cmd-->>Api: 201(ContactId,JobId)
```

Fluxo: nota por áudio

```
sequenceDiagram
participant App as FlutterApp
participant Api as Api
participant Cmd as AddAudioNoteCommand
participant Stt as ISpeechToTextProvider
participant Llm as ILLMProvider
```

```
participant Crm as Contact
participant Bill as CreditWallet
App->>Api: POST_/contacts/{id}/audio-note
Api->>Cmd: dispatch
Cmd->>Bill: ReserveCredits
Cmd->>Stt: Transcribe(audio)
Stt-->>Cmd: Transcript
Cmd->>Llm: SummarizeAndExtractTasks(transcript)
Llm-->>Cmd: Summary+Tasks
Cmd->>Crm: AddAudioNote(transcript,summary,tasks)
Cmd->>Bill: ConsumeCredits
Cmd-->>Api: 200
```

Entregáveis

- Documento de **modelo de domínio** (agregados, VOs, eventos, invariantes, regras) cobrindo todos os módulos do PRD.
- Esqueleto de solução .NET 10 (Clean Architecture) pronto para evoluir.
- Suite de testes: domínio + integração + contrato.
- OpenAPI + convenções para clientes Next.js/Flutter.

Próximos passos (execução)

- Confirmar estrutura do monorepo e naming final (`AssistenteExecutivo.*`).
- Implementar esqueleto + domínio MVP (Auth + Contacts + Capture jobs + CreditWallet).
- Adicionar testes do domínio e integração para os fluxos do MVP.
- Evoluir para módulos 2–3 com providers pluggáveis e testes focados em regras.

TODOs (implementação)

- **parse-prd**: Extrair e normalizar requisitos do PRD (MVP + módulos 2–3 + billing) e transformar em backlog técnico e regras de domínio.
- **solution-skeleton**: Criar solução .NET 10 (Domain/Application/Infrastructure/Api) + padrões de testabilidade (`IClock`, ports) + configuração de testes.
- **domain-mvp**: Implementar domínio MVP (Identity mínimo, `Contact`, `CaptureJob`, pipeline de áudio, `CreditWallet`) com eventos de domínio.
- **tests-domain**: Criar suíte de testes do domínio para invariantes, transições de jobs e ledger de créditos.
- **tests-integration**: Criar testes de integração (handlers + persistência) e testes de contrato da API (endpoints do PRD).
- **openapi-clients**: Gerar/validar OpenAPI e definir convenções de consumo para Next.js e Flutter (DTOs/serialização/erros).