ContextMap RecordStoreContextMap {

contains SocioContext

contains CatalogoContext

contains EmprestimoContext

SocioContext -> EmprestimoContext : "Valida status de sócio antes de permitir empréstimos"

CatalogoContext -> EmprestimoContext : "Fornece informações sobre exemplares disponíveis"

EmprestimoContext -> CatalogoContext : "Atualiza status de exemplares após empréstimos e devoluções"

}

//////////////////////////

// Bounded Context: SÓCIOS

//////////////////////////

BoundedContext SocioContext {

type = CORE\_DOMAIN

implementationTechnology = "Java Spring"

DomainVisionStatement """

Gerencia os sócios (clientes), seus dados e status (ativo, bloqueado, suspenso),

permitindo controle de acesso às funcionalidades de empréstimo.

"""

Module Socios {

Aggregate SocioAggregate {

Entity Socio {

SocioId id

String nome

String email

StatusSocio status

List<EmprestimoId> emprestimos

def ativar()

def bloquear()

def suspender()

def podeRealizarEmprestimo() : boolean

def atualizarCadastro(String nome, String email)

}

ValueObject SocioId {

String value

}

Enum StatusSocio {

ATIVO, BLOQUEADO, SUSPENSO

}

Repository SocioRepository {

findById(SocioId id) : Socio

findByEmail(String email) : Socio

save(Socio socio)

delete(SocioId id)

findAll() : List<Socio>

}

Service SocioService {

def cadastrarSocio(String nome, String email) : Socio

def listarEmprestimosAtivos(SocioId id) : List<Emprestimo>

}

}

}

}

//////////////////////////////

// Bounded Context: CATÁLOGO

//////////////////////////////

BoundedContext CatalogoContext {

type = CORE\_DOMAIN

implementationTechnology = "Java Spring"

DomainVisionStatement """

Gerencia o acervo físico de mídias (CDs, vinis, fitas), incluindo cadastro, edição,

exclusão e controle de disponibilidade dos exemplares para empréstimo.

"""

Module Catalogo {

Aggregate MidiaAggregate {

Entity Midia {

MidiaId id

String titulo

String artista

String genero

int ano

TipoMidia tipo

List<Exemplar> exemplares

def adicionarExemplar(Exemplar exemplar)

def removerExemplar(Exemplar exemplar)

def atualizarDados(String titulo, String artista, String genero, int ano)

}

Entity Exemplar {

ExemplarId id

StatusExemplar status

CondicaoExemplar condicao

ref Midia midia

def marcarComoEmprestado()

def marcarComoDisponivel()

def marcarComoIndisponivel()

def atualizarCondicao(CondicaoExemplar condicao)

}

ValueObject MidiaId { String value }

ValueObject ExemplarId { String value }

Enum TipoMidia { CD, VINIL, FITA }

Enum StatusExemplar { DISPONIVEL, EMPRESTADO, INDISPONIVEL }

Enum CondicaoExemplar { NOVO, BOM, REGULAR, RUIM }

Repository MidiaRepository {

findById(MidiaId id) : Midia

findAll() : List<Midia>

save(Midia midia)

delete(MidiaId id)

}

Repository ExemplarRepository {

findDisponivelPorMidia(MidiaId id) : Exemplar

save(Exemplar exemplar)

delete(ExemplarId id)

}

Service CatalogoService {

def catalogarMidia(Midia midia)

def catalogarExemplar(MidiaId midiaId, CondicaoExemplar condicao) : Exemplar

def listarAcervo() : List<Midia>

def verificarDisponibilidade(ExemplarId id) : boolean

}

}

}

}

//////////////////////////////////

// Bounded Context: EMPRÉSTIMOS

//////////////////////////////////

BoundedContext EmprestimoContext {

type = SUPPORTING\_DOMAIN

implementationTechnology = "Java Spring"

DomainVisionStatement """

Orquestra o processo de empréstimos, renovações, devoluções e multas,

garantindo que sócios ativos possam alugar exemplares disponíveis

e que regras de atraso sejam aplicadas corretamente.

"""

Module Emprestimos {

Aggregate EmprestimoAggregate {

Entity Emprestimo {

EmprestimoId id

ref Socio socio

ref Exemplar exemplar

LocalDate dataEmprestimo

LocalDate dataPrevistaDevolucao

LocalDate? dataDevolucaoReal

Multa? multa

StatusEmprestimo status

def renovar(LocalDate novaData)

def registrarDevolucao(LocalDate data, boolean danificado)

def calcularMulta(BigDecimal valorDiario) : Multa

}

ValueObject EmprestimoId { String value }

ValueObject Multa {

BigDecimal valorPorDia

int diasAtraso

BigDecimal valorTotal

def calcularValorTotal() : BigDecimal

}

Enum StatusEmprestimo { ATIVO, FINALIZADO, ATRASADO }

Repository EmprestimoRepository {

findById(EmprestimoId id) : Emprestimo

save(Emprestimo emprestimo)

findAtivosPorSocio(SocioId id) : List<Emprestimo>

findAtrasados() : List<Emprestimo>

}

Service EmprestimoService {

def realizarEmprestimo(SocioId socioId, MidiaId midiaId) : Emprestimo

def renovarEmprestimo(EmprestimoId emprestimoId, LocalDate novaData)

def registrarDevolucao(EmprestimoId emprestimoId, LocalDate dataDevolucao, boolean danificado)

def calcularMulta(EmprestimoId emprestimoId)

}

}

}

}