# Diagramas UML - Sistema de Biblioteca Digital

## Documentação Técnica Completa

## 1. Diagrama de Classes - Modelo de Domínio Completo

classDiagram class Event { +int id +CharField name[255] +TextField description +DateTime created_at +DateTime updated_at +__str__() string +get_absolute_url() string } class Edition { +int id +PositiveIntegerField year +CharField location[255] +DateField start_date +DateField end_date +DateTime created_at +DateTime updated_at +ForeignKey event +__str__() string +get_duration() int +is_current() bool } class Author { +int id +CharField name[255] +EmailField email +TextField bio +URLField website +DateTime created_at +DateTime updated_at +__str__() string +get_article_count() int +get_slug() string } class Article { +int id +CharField title[500] +TextField abstract +CharField pdf_url[500] +FileField pdf_file +TextField bibtex +DateTime created_at +DateTime updated_at +ForeignKey edition +ManyToManyField authors +__str__() string +get_absolute_url() string +get_file_size() int +has_pdf() bool } class Subscription { +int id +EmailField email +ForeignKey author +ForeignKey event +BooleanField is_active +DateTime created_at +DateTime updated_at +__str__() string +get_subscription_type() string +deactivate() void } class NotificationLog { +int id +ForeignKey article +EmailField recipient +CharField status[20] +TextField error_message +DateTime sent_at +DateTime created_at } %% Relacionamentos Event "1" --> "*" Edition : has many Event "1" --> "*" Subscription : receives subscriptions Edition "1" --> "*" Article : contains Author "*" --> "*" Article : writes Author "1" --> "*" Subscription : subscribes to Article "1" --> "*" NotificationLog : generates notifications Subscription "1" --> "*" NotificationLog : receives notifications

## 2. Diagrama de Classes - Arquitetura de Views/Controllers

classDiagram class BaseView { <<abstract>> +HttpRequest request +get(request) HttpResponse +post(request) HttpResponse +put(request) HttpResponse +delete(request) HttpResponse +dispatch(request) HttpResponse #get_object_or_404(model, pk) Model #validate_json(request) dict #handle_errors(exception) JsonResponse } class EventListCreateView { +get(request) JsonResponse +post(request) JsonResponse -_validate_event_data(data) bool -_serialize_event(event) dict } class EventDetailView { +get(request, pk) JsonResponse +put(request, pk) JsonResponse +delete(request, pk) JsonResponse -_update_event(event, data) Event } class EditionListCreateView { +get(request) JsonResponse +post(request) JsonResponse -_parse_date(date_str) Date -_validate_edition_data(data) bool -_serialize_edition(edition) dict } class EditionDetailView { +get(request, pk) JsonResponse +put(request, pk) JsonResponse +delete(request, pk) JsonResponse -_update_edition(edition, data) Edition } class

ArticleListCreateAPIView { +get(request) JsonResponse +post(request) JsonResponse -_handle_multipart_upload(request) dict -_handle_json_payload(request) dict -_process_authors(authors_data) list -_serialize_article(article) dict -_apply_filters(queryset, params) QuerySet } class ArticleDetailView { +get(request, pk) JsonResponse +put(request, pk) JsonResponse +delete(request, pk) JsonResponse -_handle_file_update(article, files) void -_update_authors(article, authors) void -_cleanup_files(article) void } class BulkImportArticlesView { +post(request) JsonResponse -_parse_bibtex(content) list -_extract_pdfs_from_zip(zip_file) dict -_validate_article_data(data, index) dict -_find_matching_pdf(article_data, pdf_files) ContentFile -_create_article_from_bibtex(data, edition) Article -_process_authors_from_bibtex(article, authors) void -_generate_import_report(results) dict -_log_import_statistics(stats) void } class AuthorArticlesView { +get(request, pk) JsonResponse -_group_articles_by_year(articles) dict -_serialize_author_profile(author) dict } class AuthorByNameView { +get(request, author_name) JsonResponse -_find_author_by_name(name) Author -_generate_author_statistics(author) dict } class SubscriptionCreateView { +post(request) JsonResponse -_validate_subscription_data(data) bool -_check_existing_subscription(email, target) Subscription -_create_subscription(data) Subscription } class SubscriptionListView { +get(request) JsonResponse -_serialize_subscriptions(subscriptions) list } %% Herança BaseView <|-- EventListCreateView BaseView <|-- EventDetailView BaseView <|-- EditionListCreateView BaseView <|-- EditionDetailView BaseView <|-- ArticleListCreateAPIView BaseView <|-- ArticleDetailView BaseView <|-- BulkImportArticlesView BaseView <|-- AuthorArticlesView BaseView <|-- AuthorByNameView BaseView <|-- SubscriptionCreateView BaseView <|-- SubscriptionListView %% Dependências ArticleListCreateAPIView ..> NotificationService : uses BulkImportArticlesView ..> BibtexParser : uses BulkImportArticlesView ..> FileProcessor : uses

# 3. Diagrama de Classes - Sistema de Notificações

classDiagram class NotificationService { <<service>> +send_article_notification(article) void +find_subscribers(article) list +build_email_content(article, subscriber) dict +send_email(recipient, content) bool +log_notification(article, recipient, status) void -_get_event_subscribers(event) list -_get_author_subscribers(authors) list -_get_general_subscribers() list -_format_email_template(article) string } class SignalHandler { <<singleton>> +notify_subscribers_on_article(sender, instance, created) void -_is_notification_enabled() bool -_should_notify(article) bool } class EmailBackend { <<interface>> +send_mail(subject, message, from_email, recipients) bool +configure_smtp() void +validate_email(email) bool } class SMTPEmailBackend { +send_mail(subject, message, from_email, recipients) bool +configure_smtp() void +validate_email(email) bool -_connect_to_server() SMTPConnection -_authenticate() bool } class ConsoleEmailBackend { +send_mail(subject, message, from_email, recipients) bool +configure_smtp() void +validate_email(email) bool -_print_to_console(email_data) void } class SubscriptionManager { +create_subscription(email, target) Subscription +delete_subscription(email, target) bool +get_active_subscriptions(target) list +validate_subscription_data(data) bool -_check_duplicate(email, target) bool -_sanitize_email(email) string } %% Relacionamentos SignalHandler --> NotificationService : triggers NotificationService --> EmailBackend : uses NotificationService --> SubscriptionManager : queries EmailBackend <|-- SMTPEmailBackend EmailBackend <|-- ConsoleEmailBackend NotificationService --> Article : observes NotificationService --> Subscription : reads

# 4. Diagrama de Sequência - Sistema de Notificações

sequenceDiagram participant Admin as Administrador participant System as Sistema Django participant Signal as Signal Handler participant NotifSvc as Notification Service participant SubMgr as Subscription Manager participant Email as Email Backend participant DB as Database Admin->>System: Cria novo artigo System->>DB: Salva Article DB-->>System: Confirma salvamento System->>Signal: post_save signal Signal->>Signal: Verifica se é criação (created=True) Signal->>NotifSvc: notify_subscribers_on_article(article) NotifSvc->>SubMgr: find_subscribers(article) SubMgr->>DB: Busca subscriptions por evento DB-->>SubMgr: Lista de event subscriptions SubMgr->>DB: Busca subscriptions por autores DB-->>SubMgr: Lista de author subscriptions SubMgr->>DB: Busca subscriptions gerais DB-->>SubMgr: Lista de general subscriptions SubMgr-->>NotifSvc: Lista única de emails loop Para cada subscriber NotifSvc->>NotifSvc: build_email_content(article, email) NotifSvc->>Email: send_mail(subject, content, recipients) alt Email enviado com sucesso Email-->>NotifSvc: Success NotifSvc->>DB: Log notification (status: sent) else Falha no envio Email-->>NotifSvc: Error NotifSvc->>DB: Log notification (status: failed) end end NotifSvc-->>Signal: Relatório de envio Signal-->>System: Notificações processadas

# 5. Diagrama de Sequência - Importação Bulk com Validação

sequenceDiagram participant Admin as Administrador participant Frontend as React Frontend participant API as Bulk Import API participant Parser as BibTeX Parser participant Validator as Data Validator participant FileProc as File Processor participant DB as Database participant NotifSys as Notification System Admin->>Frontend: Seleciona arquivos (BibTeX + ZIP) Admin->>Frontend: Escolhe edição Admin->>Frontend: Inicia importação Frontend->>API: POST /api/articles/bulk-import/ Note over Frontend,API: multipart/form-data:<br/>bibtex_file, pdf_zip, edition_id API->>FileProc: extract_pdfs_from_zip(zip_file) FileProc-->>API: Dict[filename -> ContentFile] API->>Parser: parse_bibtex(bibtex_content) Parser-->>API: List[article_data] loop Para cada entrada BibTeX API->>Validator: validate_article_data(entry, index) Validator-->>API: ValidationResult alt Validação OK API->>FileProc: find_matching_pdf(entry, pdf_files) FileProc-->>API: ContentFile ou None API->>DB: Create Article DB-->>API: Article instance alt PDF encontrado API->>DB: Attach PDF file end loop Para cada autor API->>DB: get_or_create Author DB-->>API: Author instance API->>DB: Add author to article end API->>DB: Save final article DB-->>NotifSys: Trigger post_save signal NotifSys->>NotifSys: Process notifications else Validação falhou API->>API: Add to skipped_list end end API->>API: generate_import_report(results) API-->>Frontend: JsonResponse com relatório Frontend-->>Admin: Exibe relatório detalhado

# 6. Diagrama de Atividades - Fluxo de Pesquisa Avançada

flowchart TD A[Usuário acessa pesquisa] --> B[Digite termo de busca] B --> C{Tipo de pesquisa?} C -->|Título| D[Aplicar filtro title__icontains] C -->|Autor| E[Aplicar filtro authors__name__iregex] C -->|Evento| F[Aplicar filtro edition__event__name__icontains] C -->|Todos os campos| G[Combinar múltiplos filtros] D --> H[Query no banco] E --> I[Query com regex para palavras completas] F --> J[Query relacionada com evento] G --> K[Query complexa com OR/AND] H --> L[Aplicar distinct] I --> L J --> L K --> L L --> M[Ordenar resultados] M --> N[Paginar resultados] N --> O[Serializar para JSON] O --> P[Retornar JsonResponse] P --> Q[Frontend renderiza resultados] Q --> R{Usuário satisfeito?} R -->|Não| S[Refinar busca] R -->|Sim| T[Ver detalhes do artigo] S --> B T --> U[Carregar página do artigo] U --> V[Exibir PDF se disponível] V --> W[Mostrar autores e links] W --> X[Fim]

# 7. Diagrama de Componentes - Arquitetura Completa

graph TB subgraph "Camada de Apresentação" UI[Interface React] COMP[Componentes UI] HOOKS[Custom Hooks] FORMS[Formulários] ROUTING[React Router] end subgraph "Camada de API/Cliente" API_CLIENT[API Client] HTTP[HTTP Client] CACHE[Cache Local] VALIDATION[Validação Frontend] end subgraph "Camada de Aplicação (Django)" URLS[URL Dispatcher] VIEWS[Class-Based Views] MIDDLEWARE[Middleware Stack] AUTH[Authentication] end subgraph "Camada de Domínio" MODELS[Django Models] SIGNALS[Django Signals] MANAGERS[Custom Managers] VALIDATORS[Model Validators] end subgraph "Camada de Serviços" NOTIFICATION[Notification Service] EMAIL[Email Service] FILE_PROC[File Processing] BIBTEX[BibTeX Parser] PDF_PROC[PDF Processor] end subgraph "Camada de Dados" ORM[Django ORM] DB[(SQLite Database)] MEDIA[Media Storage] LOGS[Log Files] end subgraph "Infraestrutura Externa" SMTP[SMTP Server] FILE_SYS[File System] BACKUP[Backup System] end %% Conexões da Apresentação UI --> COMP COMP --> HOOKS HOOKS --> FORMS FORMS --> ROUTING %% Conexões API/Cliente HOOKS --> API_CLIENT API_CLIENT --> HTTP HTTP --> CACHE CACHE --> VALIDATION %% Conexões Backend HTTP -.->|HTTP/JSON| URLS URLS --> VIEWS VIEWS --> MIDDLEWARE MIDDLEWARE --> AUTH %% Conexões Domínio VIEWS --> MODELS MODELS --> SIGNALS SIGNALS --> MANAGERS MANAGERS --> VALIDATORS %% Conexões Serviços SIGNALS --> NOTIFICATION NOTIFICATION --> EMAIL VIEWS --> FILE_PROC FILE_PROC --> BIBTEX BIBTEX --> PDF_PROC %% Conexões Dados MODELS --> ORM ORM --> DB FILE_PROC --> MEDIA NOTIFICATION --> LOGS %% Conexões Externas EMAIL --> SMTP MEDIA --> FILE_SYS DB --> BACKUP

# 8. Diagrama de Estados - Ciclo de Vida do Artigo

stateDiagram-v2 [*] --> Draft : Admin inicia cadastro Draft --> Validating : Submete formulário Validating --> Draft : Dados inválidos Validating --> Processing : Dados válidos Processing --> PendingFiles : Salvando metadados PendingFiles --> Processing : Erro no upload PendingFiles --> PendingAuthors : PDF processado PendingAuthors --> PendingNotification : Autores vinculados PendingNotification --> Published : Notificações enviadas Published --> Updating : Admin edita Updating --> Published : Atualização salva Updating --> Error : Erro na atualização Error --> Draft : Corrigir dados Error --> Published : Ignorar erro Published --> Archived : Admin arquiva Archived --> Published : Admin restaura Published --> [*] : Admin deleta note right of Draft Artigo sendo criado Dados ainda não validados end note note right of Published Artigo visível publicamente Notificações enviadas PDF disponível end note note right of Error Estado de erro Requer intervenção end note

# 9. Diagrama de Casos de Uso Detalhado

graph TB subgraph "Atores" ADMIN[Administrador] USER[Usuário Final] SYSTEM[Sistema] EMAIL_SYS[Sistema de Email] end subgraph "Casos de Uso - Gestão" UC1[Gerenciar Eventos] UC2[Gerenciar Edições] UC3[Cadastrar Artigo Individual] UC4[Importar Artigos em Massa] UC5[Validar Dados de Importação] UC6[Processar Arquivos PDF] UC7[Gerenciar Autores] end subgraph "Casos de Uso - Pesquisa" UC8[Pesquisar Artigos] UC9[Filtrar por Autor] UC10[Filtrar por Evento] UC11[Visualizar Detalhes do Artigo] UC12[Download de PDF] UC13[Visualizar Página do Autor] end subgraph "Casos de Uso - Notificações" UC14[Inscrever-se para Notificações] UC15[Receber Notificações por Email] UC16[Cancelar Inscrições] UC17[Enviar Notificações Automáticas] end %% Relacionamentos Administrador ADMIN --> UC1 ADMIN --> UC2 ADMIN --> UC3 ADMIN --> UC4 ADMIN --> UC7 %% Relacionamentos Usuário USER --> UC8 USER --> UC9 USER --> UC10 USER --> UC11 USER --> UC12 USER --> UC13 USER --> UC14 USER --> UC16 %% Relacionamentos Sistema SYSTEM --> UC5 SYSTEM --> UC6 SYSTEM --> UC17 EMAIL_SYS --> UC15 %% Dependências (includes/extends/triggers) UC4 -.-> UC5 UC4 -.-> UC6 UC3 -.-> UC17 UC4 -.-> UC17 UC17 -.-> UC15 UC14 -.-> UC15 UC8 -.-> UC9 UC8 -.-> UC10 UC11 -.-> UC12

# Resumo da Arquitetura e Padrões

## Arquitetura em Camadas:

1. **Apresentação**: React/TypeScript com componentes reutilizáveis
2. **API/Cliente**: Cliente HTTP com cache e validação
3. **Aplicação**: Django views com autenticação e middleware
4. **Domínio**: Models Django com signals e managers customizados
5. **Serviços**: Processamento de arquivos, notificações, parsing
6. **Dados**: Django ORM com SQLite e storage de arquivos

## Padrões de Design Implementados:

- **Observer Pattern**: Sistema de signals para notificações automáticas
- **Strategy Pattern**: Diferentes backends de email (SMTP vs Console)
- **Factory Pattern**: Criação automática de autores e eventos
- **Command Pattern**: Bulk import com validação e rollback
- **Repository Pattern**: Django ORM como abstração de dados
- **MVC/MVT**: Separação clara de responsabilidades
- **Singleton Pattern**: Gerenciamento de configurações de email

## Principais Funcionalidades:

- **CRUD Completo**: Para todas as entidades (Events, Editions, Articles, Authors)
- **Importação Bulk**: Parser BibTeX com matching automático de PDFs
- **Sistema de Notificações**: Email automático para subscribers
- **Pesquisa Avançada**: Múltiplos filtros com regex e relacionamentos
- **Upload de Arquivos**: Processamento de PDFs com validação

- **Relatórios**: Estatísticas de importação e uso do sistema