**NESTE PDF COLOQUEI OS CÓDIGOS DOS DIAGRAMAS UML FEITOS:**

**diagrama de caso de uso:**

```
@startuml
left to right direction

actor Usuário
actor "detentor do acervo" as Detentor
actor "Adm (Administrador)" as Admin

' Tanto o Detentor quanto o Admin "são um" Usuário
Usuário <|-- Detentor
Usuário <|-- Admin

rectangle "Santo Restauro App" {

  ' Casos de Uso (UCs)
  usecase "Fazer login" as UC_Login
  usecase "Verificar senha" as UC_VerificaSenha

  usecase "gerir seu acervo" as UC_GerirProprio
  usecase "exportar acervo para xlsx ou pdf" as UC_Exportar

  usecase "Gerir acervos" as UC_GerirTodos
  usecase "ligar um acervo a um usuário" as UC_LigarAcervo

  usecase "Cadastrar usuários" as UC_CadastrarUser
  usecase "adicionar um acervo a um usuário" as UC_AddAcervoUser

  usecase "Gerir itens de um acervo" as UC_GerirItens
  usecase "selecionar template usado no item" as UC_SelecionarTemplate

  usecase "Gerir templates" as UC_GerirTemplates



  UC_Login ..> UC_VerificaSenha : <<include>>
  UC_GerirItens ..> UC_SelecionarTemplate : <<include>>


  UC_GerirProprio <. UC_Exportar : <<extend>>
  UC_GerirTodos <. UC_Exportar : <<extend>>
  UC_GerirTodos <. UC_LigarAcervo : <<extend>>
  UC_CadastrarUser <. UC_AddAcervoUser : <<extend>>
}


Usuário -- UC_Login
Detentor -- UC_GerirProprio

Admin -- UC_GerirTodos
```

Admin -- UC_CadastrarUser
Admin -- UC_GerirItens
Admin -- UC_GerirTemplates

@enduml9

---

**1. Diagrama de Sequência: Fazer Login**

@startuml
actor Usuário
participant "Sistema Santo Restauro" as Sistema

Usuário -> Sistema: POST /login(email, senha)
activate Sistema

Sistema --> Usuário: 200 OK (Token de sessão)
deactivate Sistema

@enduml

**2. Diagrama de Sequência: Adicionar Item ao Acervo**

@startuml
actor "Detentor do Acervo" as Detentor
participant "Sistema Santo Restauro" as Sistema

Detentor -> Sistema: GET /acervo/{id}/itens/novo
activate Sistema
Sistema --> Detentor: 200 OK (Página de cadastro de item)
deactivate Sistema

Detentor -> Sistema: GET /api/templates
activate Sistema
Sistema --> Detentor: 200 OK (Lista de templates)
deactivate Sistema

Detentor -> Sistema: POST /acervo/{id}/itens (dadosDoItem, templateId)
activate Sistema
Sistema --> Detentor: 201 Created (Item cadastrado)
deactivate Sistema

@enduml

**3. Diagrama de Sequência: Exportar Acervo para PDF**

@startuml
actor "Detentor do Acervo" as Detentor
participant "Sistema Santo Restauro" as Sistema

```
Detentor -> Sistema: GET /acervo/{id}/exportar?formato=pdf
activate Sistema

Sistema --> Detentor: 200 OK (Arquivo PDF para download)
deactivate Sistema

@enduml
```

---

**Nível 1: Diagrama de Contexto (C1)**

```
@startuml
' Carrega a biblioteca C4
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml

Person_Ext(admin, "Administrador", "Gerencia todo o sistema, usuários e templates.")
Person_Ext(detentor, "Detentor do Acervo", "Gerencia seus próprios acervos e itens.")
Person_Ext(visitante, "Usuário (Visitante)", "Consulta acervos públicos.")

System(santo_restauro, "Sistema Santo Restauro", "Plataforma web para gerenciamento de acervos e seus itens.")

Rel(admin, santo_restauro, "Gerencia o sistema", "HTTPS")
Rel(detentor, santo_restauro, "Gerencia seus acervos", "HTTPS")
Rel(visitante, santo_restauro, "Consulta acervos", "HTTPS")

@enduml
```

**Nível 2: Diagrama de Contêineres (C2)**

```
@startuml
' Carrega a biblioteca C4
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/
C4_Container.puml

Person_Ext(admin, "Administrador", "Gerencia todo o sistema.")
Person_Ext(detentor, "Detentor do Acervo", "Gerencia seus próprios acervos.")
Person_Ext(visitante, "Usuário (Visitante)", "Consulta acervos públicos.")

System_Boundary(c1, "Sistema Santo Restauro") {

    Container(spa, "Aplicação Web (Front-end)", "JavaScript, HTML, CSS", "Interface do usuário no navegador.")

    Container(api, "API REST (Back-end)", "python/django", "Contém a lógica de negócio e expõe a API.")

    ContainerDb(db, "Banco de Dados", "PostgreSQL", "Armazena dados de usuários, acervos, itens, etc.")
}
```

```
Rel(admin, spa, "Usa", "HTTPS")
Rel(detentor, spa, "Usa", "HTTPS")
Rel(visitante, spa, "Usa", "HTTPS")

Rel(spa, api, "Faz requisições API", "JSON/HTTPS")

Rel_Right(api, db, "Lê e escreve", "JDBC/SQL")

@enduml
```

**Diagrama de Componentes (Nível C3)**

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/
C4_Component.puml

Container(spa, "Aplicação Web (Front-end)", "JavaScript, React", "Interface do usuário no
navegador.")
ContainerDb(db, "Banco de Dados", "PostgreSQL", "Armazena dados de usuários, acervos, itens,
etc.")

Container_Boundary(api, "API REST (Back-end)") {


    Component(controller, "API Controller (Views/Viewsets)", "Django REST Framework", "Recebe
requisições HTTP do Front-end e as roteia.")

    Component(user_service, "Serviço de Usuário", "Python", "Lógica de autenticação e cadastro de
usuários (auth).")
    Component(acervo_service, "Serviço de Acervo e Itens", "Python", "Lógica para gerenciar
acervos e seus itens.")
    Component(template_service, "Serviço de Templates", "Python", "Lógica para gerenciar
templates de itens.")
    Component(export_service, "Serviço de Exportação", "Python", "Gera arquivos PDF e XLSX.")

    Component(data_access, "Camada de Acesso a Dados (Django ORM)", "Django Models",
"Abstrai a comunicação com o Banco de Dados.")


    Rel(controller, user_service, "Usa")
    Rel(controller, acervo_service, "Usa")
    Rel(controller, template_service, "Usa")
    Rel(controller, export_service, "Usa")

    Rel(user_service, data_access, "Usa")
    Rel(acervo_service, data_access, "Usa")
    Rel(template_service, data_access, "Usa")
    Rel(export_service, data_access, "Usa")
}

Rel(spa, controller, "Faz requisições API", "JSON/HTTPS")
```

```
Rel_Right(data_access, db, "Lê e Escreve", "Django ORM / SQL")

@enduml
```

---

**Diagrama de Implantação**

```
@startuml
actor Usuário

node "Navegador do Usuário" as Navegador {
   artifact "Aplicação Web (React)" as SPA
}

cloud "AWS" {

   node "Servidor Web (ex: S3, Blob Storage)" as ServidorWeb {
      artifact "Aplicação Web (React)" as SPA_Source
   }

   node "EC2" as ServidorApp {
      artifact "API REST (Django)" as API
   }

   database "RDS" as ServidorBD {
      artifact "Banco PostgreSQL" as DB
   }
}

Usuário -- Navegador
Navegador ..> ServidorWeb : "Baixa o SPA React (HTTPS)"
SPA ..> API : "Faz requisições API (JSON/HTTPS)"

' Django ORM usa o driver do Postgres (psycopg2) para falar com o BD
API ..> DB : "Lê e Escreve (Django ORM)"
ServidorWeb <. SPA_Source : "Deploy"

@enduml
```

---

**Diagrama de Classes**

```
@startuml
skinparam classAttributeIconSize 0
hide empty members
skinparam padding 5

' No Django, isso seria o User model
class Usuario {
  + usuario_id: Int (PK)
  + username: CharField
```

```
  + email: EmailField
  + senha: CharField
  + tipo_usuario: CharField
}

class Contratante {
  + contratante_id: Int (PK)
  + nome: CharField
  + logradouro: CharField
  + bairro: CharField
  + numero: CharField
  + ... (outros campos de endereço)
  ' Relação 1-para-1 com o Usuário
  + usuario: OneToOneField(Usuario)
}

class Acervo {
  + acervo_id: Int (PK)
  + nome: CharField
  + cnpj: CharField
  + email: EmailField
  + ... (outros campos de metadata)
  + imagem: ImageField
  + public: BooleanField
  ' Relação 1-para-N com Contratante
  + contratante: ForeignKey(Contratante)
}

class Template {
  + template_id: Int (PK)
  + nome: CharField
  + dados_template: JSONField
}

class Item {
  + item_id: Int (PK)
  + nome: CharField
  + numero_inventario: IntegerField
  + tipo_item: CharField
  + imagem: ImageField
  + dados_item: JSONField
  ' Relação 1-para-N com Acervo
  + acervo: ForeignKey(Acervo)
  ' Relação 1-para-N com Template
  + template: ForeignKey(Template)
}


Usuario "1" -- "1" Contratante : " (usuario)"

Contratante "1" -- "0..*" Acervo : " (contratante)"
```

```
Acervo "1" -- "0..*" Item : " (acervo)"

Template "1" -- "0..*" Item : " (template)"

@enduml
```

---

**Diagramas de Sequência**

```
@startuml
actor "Detentor" as Ator
participant "Aplicação Web (React)" as SPA
participant "API Controller (Viewsets)" as Controller
participant "Serviço de Acervo e Itens" as AcervoService
participant "Serviço de Templates" as TemplateService
participant "Camada de Acesso (ORM)" as ORM
database "Banco (PostgreSQL)" as DB

Ator -> SPA: 1. Preenche e envia formulário de novo item
activate SPA

SPA -> Controller: 2. POST /api/acervo/{id}/itens (dadosDoItem, templateId)
activate Controller

Controller -> AcervoService: 3. adicionar_item(dadosDoItem, templateId)
activate AcervoService

AcervoService -> TemplateService: 4. validar_template(templateId)
activate TemplateService
TemplateService -> ORM: 5. get_template_by_id(templateId)
activate ORM
ORM -> DB: 6. SELECT * FROM template WHERE id = ?
activate DB
DB --> ORM: 7. (Dados do Template)
deactivate DB
ORM --> TemplateService: 8. (Objeto Template)
deactivate ORM
TemplateService --> AcervoService: 9. (Template validado)
deactivate TemplateService

AcervoService -> ORM: 10. create_item(dadosDoItem, acervoId, templateId)
activate ORM
ORM -> DB: 11. INSERT INTO item (...) VALUES (...)
activate DB
DB --> ORM: 12. (Novo Item com ID)
deactivate DB
ORM --> AcervoService: 13. (Objeto Item criado)
deactivate ORM

AcervoService --> Controller: 14. (Item criado)
deactivate AcervoService
```

```
Controller --> SPA: 15. 201 Created (JSON do Item)
deactivate Controller
SPA --> Ator: 16. Exibe "Item salvo com sucesso"
deactivate SPA

@enduml
```

---

**Diagrama de Comunicação**

```
@startuml
actor "Detentor" as Ator
participant "Aplicação Web (React)" as SPA
participant "API Controller (Viewsets)" as Controller
participant "Serviço de Acervo e Itens" as AcervoService
participant "Serviço de Templates" as TemplateService
participant "Camada de Acesso (ORM)" as ORM
database "Banco (PostgreSQL)" as DB

Ator -> SPA : 1. Preenche e envia formulário
SPA -> Controller : 2. POST /api/acervo/{id}/itens
Controller -> AcervoService : 3. adicionar_item(...)

AcervoService -> TemplateService : 4. validar_template(templateId)
TemplateService -> ORM : 5. get_template_by_id(templateId)
ORM -> DB : 6. SELECT * FROM template...
DB --> ORM : 7. (Dados do Template)
ORM --> TemplateService : 8. (Objeto Template)
TemplateService --> AcervoService : 9. (Template validado)

AcervoService -> ORM : 10. create_item(...)
ORM -> DB : 11. INSERT INTO item...
DB --> ORM : 12. (Novo Item com ID)
ORM --> AcervoService : 13. (Objeto Item criado)

AcervoService --> Controller : 14. (Item criado)
Controller --> SPA : 15. 201 Created (JSON)
SPA -> Ator : 16. Exibe "Item salvo com sucesso"

@enduml
```

---

**Diagramas de Estados: ciclo de vida de um objeto acervo**

```
@startuml
title Ciclo de Vida do Acervo (Visibilidade)

state "Privado (public=false)" as Privado
state "Público (public=true)" as Publico
```

```
[*] --> Privado : criarAcervo()

Privado --> Publico : publicar()
Publico --> Privado : despublicar()

Privado --> [*] : excluir()
Publico --> [*] : excluir()

@enduml
```

**modelo de dados**

O modelo de dados não foi feito usando plantuml, por isso, não tenho o seu código.