LINK PARA USO DA FERRAMENTA:
PlantUML Web Server

https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa700003

Codigo do login:

```
@startuml
actor User as "Usuário"
participant "Tela de Login" as LoginScreen
participant "Serviço de Autenticação" as AuthService
participant "Banco de Dados" as Database

User -> LoginScreen: Digita matrícula e senha
LoginScreen -> AuthService: Envia credenciais
AuthService -> Database: Consulta credenciais (matrícula, senha)
Database --> AuthService: Retorna resultado da validação
alt Credenciais válidas
    AuthService --> LoginScreen: Login bem-sucedido
    LoginScreen --> User: Acesso concedido (Aluno ou Professor)
else Credenciais inválidas
    AuthService --> LoginScreen: Erro de autenticação
    LoginScreen --> User: Exibe mensagem de erro
end
@enduml
```

codigo do cadastro:

```
@startuml
actor User as "Usuário"
participant "Tela de Cadastro" as RegisterScreen
participant "Serviço de Cadastro" as RegisterService
participant "Banco de Dados" as Database

User -> RegisterScreen: Preenche os dados (nome, matrícula, senha, etc.)
alt É Professor
    RegisterScreen -> RegisterService: Envia dados com ID de professor
    RegisterService -> Database: Valida ID de professor
    Database --> RegisterService: Resultado da validação
    alt ID de professor válido
        RegisterService -> Database: Salva dados do usuário (Professor)
        RegisterService --> RegisterScreen: Cadastro bem-sucedido
        RegisterScreen --> User: Exibe mensagem de sucesso
    else ID de professor inválido
```

```
        RegisterService --> RegisterScreen: Erro de validação
        RegisterScreen --> User: Exibe mensagem de erro
    end
else É Aluno
    RegisterScreen -> RegisterService: Envia dados de aluno (incluindo matrícula)
    RegisterService -> Database: Valida matrícula de aluno
    Database --> RegisterService: Resultado da validação
    alt Matrícula válida
        RegisterService -> Database: Salva dados do usuário (Aluno)
        RegisterService --> RegisterScreen: Cadastro bem-sucedido
        RegisterScreen --> User: Exibe mensagem de sucesso
    else Matrícula inválida
        RegisterService --> RegisterScreen: Erro de validação
        RegisterScreen --> User: Exibe mensagem de erro
    end
end
@enduml
```

codigo de redefinição de senha:

```
@startuml
actor User as "Usuário"
participant "Tela de Redefinição" as ResetScreen
participant "Serviço de Redefinição" as ResetService
participant "Banco de Dados" as Database
participant "Serviço de Notificação" as NotificationService

User -> ResetScreen: Informa email e telefone
ResetScreen -> ResetService: Envia dados para validação
ResetService -> Database: Verifica se o email e telefone correspondem
Database --> ResetService: Resultado da verificação
alt Dados válidos
    ResetService -> Database: Gera token de redefinição
    ResetService -> NotificationService: Envia token por email ou SMS
    NotificationService --> User: Recebe o token
    User -> ResetScreen: Informa token e nova senha
    ResetScreen -> ResetService: Valida token
    ResetService -> Database: Atualiza senha no banco de dados
    Database --> ResetService: Confirmação de atualização
    ResetService --> ResetScreen: Senha redefinida com sucesso
    ResetScreen --> User: Exibe mensagem de sucesso
else Dados inválidos
    ResetService --> ResetScreen: Erro de validação
    ResetScreen --> User: Exibe mensagem de erro
end
@enduml
```

nova interface aluno:

```
@startuml
actor Aluno
participant "Interface do Aluno" as StudentInterface
participant "Serviço de Análise de Código" as AnalysisService
participant "IA Generativa" as GenerativeAI
participant "Banco de Dados" as Database
participant "Serviço de Notificação" as NotificationService

Aluno -> StudentInterface: Insere script/código com dúvidas
StudentInterface -> AnalysisService: Envia código para análise
AnalysisService -> GenerativeAI: Analisa o código
GenerativeAI --> AnalysisService: Retorna correção, explicação e sugestão
AnalysisService -> StudentInterface: Envia resposta gerada pela IA
StudentInterface --> Aluno: Mostra correção, explicação e pergunta: "Você entendeu?"
loop Até 4 tentativas
    Aluno -> StudentInterface: Responde "Não"
    StudentInterface -> AnalysisService: Solicita nova explicação
    AnalysisService -> GenerativeAI: Reanalisa código para gerar nova resposta
    GenerativeAI --> AnalysisService: Retorna nova explicação
    AnalysisService -> StudentInterface: Envia nova resposta
    StudentInterface --> Aluno: Mostra nova explicação e pergunta novamente
    alt Aluno responde "Sim"
        break
    end
end
alt Aluno respondeu "Não" 4 vezes
    AnalysisService -> Database: Salva ID/Protocolo do código e respostas geradas
    AnalysisService -> NotificationService: Gera link com ID para WhatsApp do professor
    NotificationService --> StudentInterface: Envia link para WhatsApp do professor
    StudentInterface --> Aluno: Mostra link para WhatsApp com ID/protocolo
end
@enduml
```

interface professor:

```
@startuml
actor Professor
participant "Interface do Professor" as TeacherInterface
participant "Serviço de Turmas e Alunos" as ClassService
participant "Serviço de Relatórios" as ReportService
participant "Serviço de Dúvidas" as DoubtService
participant "Banco de Dados" as Database
participant "Serviço de Notificação" as NotificationService
participant "Interface do Aluno" as StudentInterface

== Acessando o Painel de Turmas ==
Professor -> TeacherInterface: Seleciona painel de turmas
TeacherInterface -> ClassService: Solicita turmas e alunos
ClassService -> Database: Consulta dados das turmas e alunos
Database --> ClassService: Retorna dados das turmas e alunos
ClassService --> TeacherInterface: Exibe painel com dados das turmas e alunos
alt Filtrando por turma ou aluno
    Professor -> TeacherInterface: Aplica filtro (por turma ou nome do aluno)
    TeacherInterface -> ClassService: Solicita dados filtrados
    ClassService -> Database: Consulta dados filtrados
    Database --> ClassService: Retorna dados filtrados
    ClassService --> TeacherInterface: Atualiza painel com dados filtrados
end

== Acessando o Relatório de Desempenho ==
Professor -> TeacherInterface: Seleciona painel de relatórios
alt Relatório geral
    TeacherInterface -> ReportService: Solicita relatório geral
    ReportService -> Database: Consulta dados gerais de desempenho
    Database --> ReportService: Retorna dados gerais
    ReportService --> TeacherInterface: Exibe relatório geral
end
alt Relatório por turma
    Professor -> TeacherInterface: Seleciona turma específica
    TeacherInterface -> ReportService: Solicita relatório da turma
    ReportService -> Database: Consulta dados de desempenho da turma
    Database --> ReportService: Retorna dados da turma
    ReportService --> TeacherInterface: Exibe relatório da turma
end
alt Relatório individual
    Professor -> TeacherInterface: Seleciona aluno específico
    TeacherInterface -> ReportService: Solicita relatório do aluno
    ReportService -> Database: Consulta dados de desempenho individual
    Database --> ReportService: Retorna dados individuais
    ReportService --> TeacherInterface: Exibe relatório do aluno
```

```
end

== Acessando o Painel de Dúvidas ==
Professor -> TeacherInterface: Seleciona painel de dúvidas
TeacherInterface -> DoubtService: Solicita dúvidas notificadas
DoubtService -> Database: Consulta dúvidas notificadas
Database --> DoubtService: Retorna lista de dúvidas
DoubtService --> TeacherInterface: Exibe painel com dúvidas notificadas
alt Visualizar dúvida específica
   Professor -> TeacherInterface: Seleciona dúvida por ID/Protocolo
   TeacherInterface -> DoubtService: Solicita detalhes da dúvida
   DoubtService -> Database: Consulta dados da dúvida
   Database --> DoubtService: Retorna detalhes da dúvida
   DoubtService --> TeacherInterface: Exibe detalhes e histórico da dúvida
   alt Enviar mensagem ao aluno
      Professor -> TeacherInterface: Escreve mensagem ou envia material
      TeacherInterface -> DoubtService: Envia mensagem para o aluno
      DoubtService -> NotificationService: Notifica o aluno sobre a mensagem
      NotificationService -> StudentInterface: Notifica aluno e atualiza chat
   end
end
@enduml
```

EXEMPLO DE COMO USAR:

```
@startuml
actor Ator
participant "Participante A" as A
participant "Participante B" as B
participant "Participante C" as C

Ator -> A: Mensagem para A
A -> B: Mensagem para B
B -> C: Mensagem para C
C --> B: Retorno de mensagem
B --> A: Retorno de mensagem
A --> Ator: Retorno final

alt Condição alternativa
   A -> B: Mensagem alternativa
   B --> A: Retorno da alternativa
else Outro caminho
   A -> C: Mensagem de outro caminho
   C --> A: Retorno de outro caminho
end
```

```
loop Repetição
    A -> B: Mensagem em loop
    B --> A: Retorno em loop
end
@enduml
```