API Spec

2025-10-30

Mentorship Service API

Overview

Serviço de Mentorias e Vouchers - Gerencia vouchers, ciclo de vida de mentorias, agendamentos, feedback e orquestração com outros serviços.

Responsabilidades:
- Resgate e validação de vouchers
- Orquestração da criação de mentorias (integrando com matchmaking-service)
- Gerenciamento do ciclo de vida completo das mentorias
- Concessão de vouchers aos mentores baseada em feedback
- Integração com admin-service, profile-service e matchmaking-service.

Version

1.0.0

Contact

Mentor-Mentee Platform Team

POST /api/v1/vouchers/redeem

Resgatar voucher.

Permite que um mentorado resgate um voucher válido.

Request Body:

Content: application/json | VoucherRedeemRequest

```
{
  code: string;  // Código do voucher a ser resgatado.
  menteeId: string;  // ID do mentorado que está resgatando.
}
```

Response 200:

Voucher resgatado com sucesso.

Content: application/json | VoucherRedeemResponse

```
{
  id: string;  // ID único do voucher.
  code: string;  // Código do voucher resgatado.
  menteeId: string;  // ID do mentorado.
  redeemedAt: string;  // Data e hora do resgate.
  expiresAt: string;  // Data de expiração do voucher.
  value: number;  // Valor do voucher em créditos.
  mentorshipEligible: boolean;  // Se o voucher permite iniciar mentoria.
}
```

Response 400:

Response 404:

Response 409:

Response 500:

GET /api/v1/vouchers/validate/{code}

Validar voucher.

Verifica se um código de voucher é válido e ainda não foi utilizado.

Request Parameters:

code: string; // Código do voucher para validação.

Response 200:

Voucher válido.

Content: application/json | VoucherValidationResponse

```
{
  valid: boolean; // Se o voucher é válido.
  code: string; // Código do voucher.
  value: number; // Valor do voucher.
  expiresAt: string; // Data de expiração.
  alreadyUsed: boolean; // Se já foi utilizado.
}
```

Response 404:

Response 410:

Voucher expirado.

Content: application/json | ErrorResponse

```
{
  error: string; // Código do erro.
  message: string; // Mensagem descritiva do erro.
  details: object; // Detalhes adicionais sobre o erro.
  timestamp: string; // Data e hora do erro.
  path: string; // Endpoint onde ocorreu o erro.
  correlationId: string; // ID para rastreamento do erro.
}
```

Response 500:

POST /api/v1/mentorships

Criar mentoria.

Cria uma nova mentoria, orquestrando com matchmaking-service para encontrar mentor adequado baseado no perfil e preferências do mentorado.

Request Body:

Content: application/json | CreateMentorshipRequest

```
{
  menteeId: string; // ID do mentorado.
```

```
    voucherId: string;  // ID do voucher resgatado.
    preferences: MentorshipPreferences;
    urgency?: string;  // Urgência para encontrar mentor.
    notes?: string;  // Notas adicionais sobre a mentoria desejada.
}
```

Response 201:

Mentoria criada com sucesso.

Content: application/json | MentorshipResponse

```
{
    id: string;  // ID único da mentoria.
    menteeId: string;  // ID do mentorado.
    mentorId: string;  // ID do mentor.
    voucherId: string;  // ID do voucher utilizado.
    status: MentorshipStatus;
    preferences: MentorshipPreferences;
    scheduledSession: ScheduledSession;
    createdAt: string;  // Data de criação.
    updatedAt: string;  // Data da última atualização.
    matchingScore: number;  // Score de compatibilidade mentor-mentorado.
    notes: string;  // Notas sobre a mentoria.
}
```

Response 400:

Response 404:

Mentor não encontrado.

Content: application/json | ErrorResponse

```
{
    error: string;  // Código do erro.
    message: string;  // Mensagem descritiva do erro.
    details: object;  // Detalhes adicionais sobre o erro.
    timestamp: string;  // Data e hora do erro.
    path: string;  // Endpoint onde ocorreu o erro.
    correlationId: string;  // ID para rastreamento do erro.
}
```

Response 500:

GET /api/v1/mentorships/mentee

Listar mentorias do mentorado.

Retorna todas as mentorias onde o usuário autenticado é mentorado.

Request Parameters:

```
status: MentorshipStatus;  // Filtrar por status da mentoria.
page: integer;  // Número da página para paginação.
limit: integer;  // Limite de itens por página.
```

Response 200:

Lista de mentorias do mentorado.

Content: application/json | MentorshipListResponse

```
{
  mentorships: Array<MentorshipResponse>;
  pagination: PaginationMeta;
}
```

Response 400:

Response 500:

GET /api/v1/mentorships/mentor

Listar mentorias do mentor.

Retorna todas as mentorias onde o usuário autenticado é mentor.

Request Parameters:

```
status: MentorshipStatus;  // Filtrar por status da mentoria.
page: integer;  // Número da página para paginação.
limit: integer;  // Limite de itens por página.
```

Response 200:

Lista de mentorias do mentor.

Content: application/json | MentorshipListResponse

```
{
  mentorships: Array<MentorshipResponse>;
  pagination: PaginationMeta;
}
```

Response 400:

Response 500:

PUT /api/v1/mentorships/{id}/schedule

Agendar sessão de mentoria (Mentor).

Permite ao MENTOR agendar uma sessão de mentoria informando dia, horário, tipo (presencial/ remoto) e link da reunião.

Chama admin-service para enviar notificações ao mentorado sobre o agendamento.

IMPORTANTE: Apenas o mentor pode criar o agendamento inicial.

Request Parameters:

```
id: string;  // ID da mentoria.
```

Request Body:

Content: application/json | ScheduleMentorshipRequest

```
{
  scheduledAt: string;  // Data e hora para agendar a sessão (ISO 8601).
```

duration: integer; // Duração da sessão em minutos.
format: string; // Formato da sessão: - video: Reunião por vídeo (requer meetingLink)
- audio: Chamada de áudio (requer meetingLink ou telefone)
- chat: Chat online (requer meetingLink)
- in-person: Presencial (requer location).
meetingLink?: string; // Link para a reunião online (OBRIGATÓRIO para formatos remotos: video, audio, chat). Pode ser Google Meet, Zoom, Teams, etc.
location?: string; // Endereço completo para sessões presenciais (OBRIGATÓRIO para formato in-person). Incluir nome do local, endereço e referências.
notes?: string; // Notas do mentor sobre a sessão. Pode incluir preparação necessária, tópicos a serem abordados, materiais para trazer, etc.
timezone: string; // Fuso horário da sessão (IANA timezone). Importante para coordenar horários entre mentor e mentorado.
reminderBefore?: integer; // Quantos minutos antes enviar lembrete (opcional). Padrão: 60 minutos (1 hora).
}

Example "remoteSession":

```
{
  "scheduledAt": "2024-10-30T15:00:00Z",
  "duration": 60,
  "format": "video",
  "meetingLink": "https://meet.google.com/abc-defg-hij",
  "notes": "Primeira sessão - vamos revisar seu modelo de negócios e validar as hipóteses iniciais. Traga seu Business Model Canvas.",
  "timezone": "America/Sao_Paulo"
}
```

Example "inPersonSession":

```
{
  "scheduledAt": "2024-11-02T14:00:00Z",
  "duration": 90,
  "format": "in-person",
  "location": "Hub de Inovação, Av. Faria Lima 3000, São Paulo - Coworking Spaces",
  "notes": "Sessão prática sobre pitch e apresentação para investidores. Traga seu pitch deck.",
  "timezone": "America/Sao_Paulo"
}
```

Response 200:

Sessão agendada com sucesso pelo mentor.

Content: application/json | MentorshipResponse

{
id: string; // ID único da mentoria.
menteeId: string; // ID do mentorado.
mentorId: string; // ID do mentor.
voucherId: string; // ID do voucher utilizado.
status: MentorshipStatus;
preferences: MentorshipPreferences;
scheduledSession: ScheduledSession;
createdAt: string; // Data de criação.
updatedAt: string; // Data da última atualização.
matchingScore: number; // Score de compatibilidade mentor-mentorado.
notes: string; // Notas sobre a mentoria.
}

Response 400:

Response 403:

Apenas o mentor pode agendar sessões.

Content: application/json | ErrorResponse

```
{
  error: string;  // Código do erro.
  message: string;  // Mensagem descritiva do erro.
  details: object;  // Detalhes adicionais sobre o erro.
  timestamp: string;  // Data e hora do erro.
  path: string;  // Endpoint onde ocorreu o erro.
  correlationId: string;  // ID para rastreamento do erro.
}
```

Response 404:

Response 409:

Conflito de horário.

Content: application/json | ErrorResponse

```
{
  error: string;  // Código do erro.
  message: string;  // Mensagem descritiva do erro.
  details: object;  // Detalhes adicionais sobre o erro.
  timestamp: string;  // Data e hora do erro.
  path: string;  // Endpoint onde ocorreu o erro.
  correlationId: string;  // ID para rastreamento do erro.
}
```

Response 500:

POST /api/v1/mentorships/{id}/schedule/confirm

Confirmar agendamento (Mentorado).

Permite ao MENTORADO confirmar ou rejeitar o agendamento proposto pelo mentor.
Após confirmação, notificações são enviadas para ambas as partes.
Se rejeitado, o mentor pode propor novo horário.

Request Parameters:

id: string; // ID da mentoria.

Request Body:

Content: application/json | ConfirmScheduleRequest

```
{
  confirmed: boolean;  // Se o mentorado confirma (true) ou rejeita (false) o agendamento.
  reason?: string;  // Motivo da confirmação ou rejeição (obrigatório se confirmed=false). Exemplos:
    "Horário incompatível", "Prefiro sessão remota", etc.
  alternativeTime?: string;  // Proposta de horário alternativo (opcional, usado quando
    confirmed=false).
}
```

Response 200:

Resposta ao agendamento registrada.

Content: application/json | MentorshipResponse

```
{
  id: string;  // ID único da mentoria.
  menteeId: string;  // ID do mentorado.
```

```
    mentorId: string;  // ID do mentor.
    voucherId: string;  // ID do voucher utilizado.
    status: MentorshipStatus;
    preferences: MentorshipPreferences;
    scheduledSession: ScheduledSession;
    createdAt: string;  // Data de criação.
    updatedAt: string;  // Data da última atualização.
    matchingScore: number;  // Score de compatibilidade mentor-mentorado.
    notes: string;  // Notas sobre a mentoria.
}
```

Response 400:

Response 403:

Apenas o mentorado pode confirmar.

Content: application/json | ErrorResponse

```
{
    error: string;  // Código do erro.
    message: string;  // Mensagem descritiva do erro.
    details: object;  // Detalhes adicionais sobre o erro.
    timestamp: string;  // Data e hora do erro.
    path: string;  // Endpoint onde ocorreu o erro.
    correlationId: string;  // ID para rastreamento do erro.
}
```

Response 404:

Response 500:

POST /api/v1/mentorships/{id}/feedback/mentor

Feedback do mentor.

Permite ao mentor fornecer feedback sobre a sessão. Chama admin-service para analytics e notificações. Executa lógica interna para concessão de voucher.

Request Parameters:

id: string; // ID da mentoria.

Request Body:

Content: application/json | MentorFeedbackRequest

```
{
    rating: integer;  // Avaliação da sessão (1-5).
    sessionCompleted: boolean;  // Se a sessão foi completada.
    feedback?: string;  // Feedback textual sobre a sessão.
    menteeEngagement?: integer;  // Nível de engajamento do mentorado (1-5).
    goalAchievement?: integer;  // Nível de alcance dos objetivos (1-5).
    recommendChanges?: string;  // Sugestões de mudanças para próximas sessões.
    nextSessionNeeded?: boolean;  // Se uma próxima sessão é necessária.
}
```

Response 201:

Feedback registrado com sucesso.

Content: application/json | FeedbackResponse

```
{
  id: string;  // ID do feedback.
  mentorshipId: string;  // ID da mentoria.
  type: string;  // Tipo de feedback.
  createdAt: string;  // Data de criação do feedback.
  voucherGranted: VoucherGranted;
}
```

Response 400:

Response 404:

Response 409:

Feedback já fornecido.

Content: application/json | ErrorResponse

```
{
  error: string;  // Código do erro.
  message: string;  // Mensagem descritiva do erro.
  details: object;  // Detalhes adicionais sobre o erro.
  timestamp: string;  // Data e hora do erro.
  path: string;  // Endpoint onde ocorreu o erro.
  correlationId: string;  // ID para rastreamento do erro.
}
```

Response 500:

POST /api/v1/mentorships/{id}/feedback/mentee

Feedback do mentorado.

Permite ao mentorado fornecer feedback sobre a sessão de mentoria.

Request Parameters:

id: string; // ID da mentoria.

Request Body:

Content: application/json | MenteeFeedbackRequest

```
{
  rating: integer;  // Avaliação geral da sessão (1-5).
  mentorHelpfulness: integer;  // Quão útil foi o mentor (1-5).
  feedback?: string;  // Feedback sobre a sessão.
  goalProgress?: integer;  // Progresso em relação aos objetivos (1-5).
  mentorCommunication?: integer;  // Qualidade da comunicação do mentor (1-5).
  wouldRecommend?: boolean;  // Se recomendaria o mentor para outros.
  additionalNeeds?: string;  // Necessidades adicionais identificadas.
}
```

Response 201:

Feedback registrado com sucesso.

Content: application/json | FeedbackResponse

```
{
```

```
    id: string;  // ID do feedback.
    mentorshipId: string;  // ID da mentoria.
    type: string;  // Tipo de feedback.
    createdAt: string;  // Data de criação do feedback.
    voucherGranted: VoucherGranted;
}
```

Response 400:

Response 404:

Response 409:

Feedback já fornecido.

Content: application/json | ErrorResponse

```
{
    error: string;  // Código do erro.
    message: string;  // Mensagem descritiva do erro.
    details: object;  // Detalhes adicionais sobre o erro.
    timestamp: string;  // Data e hora do erro.
    path: string;  // Endpoint onde ocorreu o erro.
    correlationId: string;  // ID para rastreamento do erro.
}
```

Response 500:

POST /api/v1/mentorships/{id}/request-change

Solicitar mudança.

Permite solicitar mudanças na mentoria (troca de mentor, reagendamento, etc.).

Request Parameters:

id: string; // ID da mentoria.

Request Body:

Content: application/json | RequestChangeRequest

```
{
    changeType: string;  // Tipo de mudança solicitada.
    reason: string;  // Motivo da solicitação.
    preferredNewTime?: string;  // Nova data/hora preferida (para reagendamento).
    newFormat?: string;  // Novo formato desejado (para mudança de formato).
    urgency?: string;  // Urgência da solicitação.
}
```

Response 201:

Solicitação de mudança registrada.

Content: application/json | ChangeRequestResponse

```
{
    id: string;  // ID da solicitação de mudança.
    mentorshipId: string;  // ID da mentoria.
    changeType: string;  // Tipo de mudança solicitada.
    status: string;  // Status da solicitação.
```

```
    reason: string;  // Motivo da solicitação.
    requestedBy: string;  // ID de quem solicitou.
    requestedAt: string;  // Data da solicitação.
    estimatedResolution: string;  // Previsão de resolução.
}
```

Response 400:

Response 404:

Response 500:

PUT /api/v1/mentorships/{id}/status

Atualizar status da mentoria.

Endpoint usado pelo admin-service para atualizar o status das mentorias.

Request Parameters:

```
id: string;  // ID da mentoria.
```

Request Body:

Content: application/json | UpdateStatusRequest

```
{
    status: MentorshipStatus;
    reason?: string;  // Motivo da mudança de status.
    updatedBy?: string;  // ID do admin que atualizou.
    notes?: string;  // Notas adicionais sobre a mudança.
}
```

Response 200:

Status atualizado com sucesso.

Content: application/json | MentorshipResponse

```
{
    id: string;  // ID único da mentoria.
    menteeId: string;  // ID do mentorado.
    mentorId: string;  // ID do mentor.
    voucherId: string;  // ID do voucher utilizado.
    status: MentorshipStatus;
    preferences: MentorshipPreferences;
    scheduledSession: ScheduledSession;
    createdAt: string;  // Data de criação.
    updatedAt: string;  // Data da última atualização.
    matchingScore: number;  // Score de compatibilidade mentor-mentorado.
    notes: string;  // Notas sobre a mentoria.
}
```

Response 400:

Response 404:

Response 500:

Schemas

VoucherRedeemRequest

```
{
    code: string;  // Código do voucher a ser resgatado.
    menteeId: string;  // ID do mentorado que está resgatando.
}
```

VoucherRedeemResponse

```
{
    id: string;  // ID único do voucher.
    code: string;  // Código do voucher resgatado.
    menteeId: string;  // ID do mentorado.
    redeemedAt: string;  // Data e hora do resgate.
    expiresAt: string;  // Data de expiração do voucher.
    value: number;  // Valor do voucher em créditos.
    mentorshipEligible: boolean;  // Se o voucher permite iniciar mentoria.
}
```

VoucherValidationResponse

```
{
    valid: boolean;  // Se o voucher é válido.
    code: string;  // Código do voucher.
    value: number;  // Valor do voucher.
    expiresAt: string;  // Data de expiração.
    alreadyUsed: boolean;  // Se já foi utilizado.
}
```

CreateMentorshipRequest

```
{
    menteeId: string;  // ID do mentorado.
    voucherId: string;  // ID do voucher resgatado.
    preferences: MentorshipPreferences;
    urgency?: string;  // Urgência para encontrar mentor.
    notes?: string;  // Notas adicionais sobre a mentoria desejada.
}
```

MentorshipPreferences

```
{
    expertise: Array<string>;  // Áreas de expertise desejadas.
    availableTimes: Array<TimeSlot>;  // Horários disponíveis para sessões.
    preferredLanguage: string;  // Idioma preferido para as sessões.
    mentorGender: string;  // Preferência de gênero do mentor.
    sessionFormat: string;  // Formato preferido das sessões.
    maxMentorExperience: integer;  // Experiência máxima do mentor em anos (0 = sem limite).
    minMentorExperience: integer;  // Experiência mínima do mentor em anos.
}
```

TimeSlot

```
{
  dayOfWeek: integer;  // Dia da semana (1=Segunda, 7=Domingo).
  startTime: string;  // Horário de início (HH:MM).
  endTime: string;  // Horário de fim (HH:MM).
  timezone?: string;  // Fuso horário.
}
```

MentorshipResponse

```
{
  id: string;  // ID único da mentoria.
  menteeId: string;  // ID do mentorado.
  mentorId: string;  // ID do mentor.
  voucherId: string;  // ID do voucher utilizado.
  status: MentorshipStatus;
  preferences: MentorshipPreferences;
  scheduledSession: ScheduledSession;
  createdAt: string;  // Data de criação.
  updatedAt: string;  // Data da última atualização.
  matchingScore: number;  // Score de compatibilidade mentor-mentorado.
  notes: string;  // Notas sobre a mentoria.
}
```

MentorshipStatus

string
Values: pending_match, matched, scheduled, in_progress, completed, cancelled, change_requested

ScheduledSession

```
{
  id: string;  // ID da sessão agendada.
  scheduledAt: string;  // Data e hora agendada.
  duration: integer;  // Duração em minutos.
  format: string;  // Formato da sessão.
  meetingLink: string;  // Link para a reunião online (para sessões remotas).
  location: string;  // Local para sessões presenciais.
  timezone: string;  // Fuso horário da sessão.
  notes: string;  // Notas do mentor sobre a sessão.
  confirmationStatus: string;  // Status de confirmação da sessão: - pending: Aguardando confirmação do
    mentorado
                              - confirmed: Mentorado confirmou presença
                              - rejected: Mentorado rejeitou o horário
                              - cancelled: Sessão foi cancelada.
  confirmedAt: string;  // Data/hora que o mentorado confirmou.
  confirmedBy: string;  // ID de quem confirmou (mentorado).
  scheduledBy: string;  // ID de quem agendou (mentor).
  notificationsSent: boolean;  // Se as notificações foram enviadas.
  reminderSent: boolean;  // Se o lembrete foi enviado.
  createdAt: string;  // Data de criação do agendamento.
}
```

MentorshipListResponse

```
{
  mentorships: Array<MentorshipResponse>;
  pagination: PaginationMeta;
}
```

PaginationMeta

```
{
  page: integer;  // Página atual.
  limit: integer;  // Itens por página.
  total: integer;  // Total de itens.
  totalPages: integer;  // Total de páginas.
  hasNext: boolean;  // Se há próxima página.
  hasPrev: boolean;  // Se há página anterior.
}
```

ScheduleMentorshipRequest

```
{
  scheduledAt: string;  // Data e hora para agendar a sessão (ISO 8601).
  duration: integer;  // Duração da sessão em minutos.
  format: string;  // Formato da sessão: - video: Reunião por vídeo (requer meetingLink)
                   - audio: Chamada de áudio (requer meetingLink ou telefone)
                   - chat: Chat online (requer meetingLink)
                   - in-person: Presencial (requer location).
  meetingLink?: string;  // Link para a reunião online (OBRIGATÓRIO para formatos remotos: video,
    audio, chat). Pode ser Google Meet, Zoom, Teams, etc.
  location?: string;  // Endereço completo para sessões presenciais (OBRIGATÓRIO para formato in-
    person). Incluir nome do local, endereço e referências.
  notes?: string;  // Notas do mentor sobre a sessão. Pode incluir preparação necessária, tópicos a
    serem abordados, materiais para trazer, etc.
  timezone: string;  // Fuso horário da sessão (IANA timezone). Importante para coordenar horários entre
    mentor e mentorado.
  reminderBefore?: integer;  // Quantos minutos antes enviar lembrete (opcional). Padrão: 60 minutos (1
    hora).
}
```

ConfirmScheduleRequest

```
{
  confirmed: boolean;  // Se o mentorado confirma (true) ou rejeita (false) o agendamento.
  reason?: string;  // Motivo da confirmação ou rejeição (obrigatório se confirmed=false). Exemplos:
    "Horário incompatível", "Prefiro sessão remota", etc.
  alternativeTime?: string;  // Proposta de horário alternativo (opcional, usado quando confirmed=false).
}
```

MentorFeedbackRequest

```
{
  rating: integer;  // Avaliação da sessão (1-5).
  sessionCompleted: boolean;  // Se a sessão foi completada.
```

```
    feedback?: string;  // Feedback textual sobre a sessão.
    menteeEngagement?: integer;  // Nível de engajamento do mentorado (1-5).
    goalAchievement?: integer;  // Nível de alcance dos objetivos (1-5).
    recommendChanges?: string;  // Sugestões de mudanças para próximas sessões.
    nextSessionNeeded?: boolean;  // Se uma próxima sessão é necessária.
}
```

MenteeFeedbackRequest

```
{
    rating: integer;  // Avaliação geral da sessão (1-5).
    mentorHelpfulness: integer;  // Quão útil foi o mentor (1-5).
    feedback?: string;  // Feedback sobre a sessão.
    goalProgress?: integer;  // Progresso em relação aos objetivos (1-5).
    mentorCommunication?: integer;  // Qualidade da comunicação do mentor (1-5).
    wouldRecommend?: boolean;  // Se recomendaria o mentor para outros.
    additionalNeeds?: string;  // Necessidades adicionais identificadas.
}
```

FeedbackResponse

```
{
    id: string;  // ID do feedback.
    mentorshipId: string;  // ID da mentoria.
    type: string;  // Tipo de feedback.
    createdAt: string;  // Data de criação do feedback.
    voucherGranted: VoucherGranted;
}
```

VoucherGranted

```
{
    granted: boolean;  // Se um voucher foi concedido.
    voucherId: string;  // ID do voucher concedido.
    code: string;  // Código do voucher concedido.
    value: number;  // Valor do voucher.
    reason: string;  // Motivo da concessão.
}
```

RequestChangeRequest

```
{
    changeType: string;  // Tipo de mudança solicitada.
    reason: string;  // Motivo da solicitação.
    preferredNewTime?: string;  // Nova data/hora preferida (para reagendamento).
    newFormat?: string;  // Novo formato desejado (para mudança de formato).
    urgency?: string;  // Urgência da solicitação.
}
```

ChangeRequestResponse

```
{
  id: string;  // ID da solicitação de mudança.
  mentorshipId: string;  // ID da mentoria.
  changeType: string;  // Tipo de mudança solicitada.
  status: string;  // Status da solicitação.
  reason: string;  // Motivo da solicitação.
  requestedBy: string;  // ID de quem solicitou.
  requestedAt: string;  // Data da solicitação.
  estimatedResolution: string;  // Previsão de resolução.
}
```

UpdateStatusRequest

```
{
  status: MentorshipStatus;
  reason?: string;  // Motivo da mudança de status.
  updatedBy?: string;  // ID do admin que atualizou.
  notes?: string;  // Notas adicionais sobre a mudança.
}
```

ErrorResponse

```
{
  error: string;  // Código do erro.
  message: string;  // Mensagem descritiva do erro.
  details: object;  // Detalhes adicionais sobre o erro.
  timestamp: string;  // Data e hora do erro.
  path: string;  // Endpoint onde ocorreu o erro.
  correlationId: string;  // ID para rastreamento do erro.
}
```