API Spec

2025-10-30

Matchmaking Service API

Overview

Serviço de Matchmaking - Responsável por encontrar a melhor combinação entre mentorados e mentores baseado em tags/áreas de expertise e preferências.

Responsabilidades:
- Receber requisições estruturadas com necessidades do mentorado
- Consultar profile-service para encontrar mentores compatíveis
- Executar algoritmo de pontuação baseado em tags e preferências
- Retornar o mentorId mais adequado com score de compatibilidade
- Registrar logs de decisões de matching (opcional)

IMPORTANTE: Este é um serviço interno, não exposto no API Gateway.
Chamado exclusivamente pelo mentorship-service.

Version

1.0.0

Contact

Mentor-Mentee Platform Team

POST /api/v1/matchmaking

Encontrar mentor compatível.

Endpoint interno chamado pelo mentorship-service para encontrar o mentor mais adequado para um mentorado baseado em suas necessidades e preferências.

Fluxo:
1. Recebe critérios de matching do mentorado
2. Consulta profile-service para obter lista de mentores disponíveis
3. Executa algoritmo de pontuação multi-critério
4. Retorna mentor com maior score de compatibilidade
5. Opcionalmente registra decisão no MongoDB para analytics.

Request Body:

Content: application/json | MatchmakingRequest

```
{
  menteeId: string;  // ID do mentorado que precisa de mentor.
  requiredExpertise: Array<string>;  // Áreas de expertise necessárias (ordenadas por prioridade).
  preferences: MatchingPreferences;
  urgency?: string;  // Urgência para encontrar mentor (afeta o threshold de score).
  menteeProfile?: MenteeProfileSummary;
  excludeMentors?: Array<string>;  // IDs de mentores a excluir (ex. matches anteriores que não
    deram certo).
  weights?: MatchingWeights;
}
```

Example "startupValidation":

```
{
  "menteeId": "550e8400-e29b-41d4-a716-446655440000",
```

```
  "requiredExpertise": [
    "startup-validation",
    "business-model",
    "market-research",
    "mvp-development"
  ],
  "preferences": {
    "availableTimes": [
      {
        "dayOfWeek": 2,
        "startTime": "14:00",
        "endTime": "16:00",
        "timezone": "America/Sao_Paulo"
      },
      {
        "dayOfWeek": 4,
        "startTime": "18:00",
        "endTime": "20:00",
        "timezone": "America/Sao_Paulo"
      }
    ],
    "preferredLanguage": "pt-BR",
    "sessionFormat": "video",
    "minMentorExperience": 5,
    "maxMentorExperience": 0
  },
  "urgency": "high",
  "menteeProfile": {
    "currentRole": "Aspirante a Empreendedor",
    "targetRole": "Founder/CEO Startup",
    "experienceLevel": "entry",
    "location": "São Paulo, BR",
    "background": [
      "Administração",
      "Marketing Digital"
    ]
  }
}
```

Example "fundraising":

```
{
  "menteeId": "550e8400-e29b-41d4-a716-446655440001",
  "requiredExpertise": [
    "fundraising",
    "pitch-development",
    "investor-relations",
    "financial-planning"
  ],
  "preferences": {
    "availableTimes": [
      {
        "dayOfWeek": 3,
        "startTime": "10:00",
        "endTime": "12:00",
        "timezone": "America/Sao_Paulo"
      }
    ],
    "preferredLanguage": "pt-BR",
    "sessionFormat": "in-person",
    "minMentorExperience": 8,
    "mentorGender": "any"
  },
  "urgency": "high",
  "menteeProfile": {
    "currentRole": "Founder Early-Stage Startup",
    "targetRole": "Founder Funded Startup",
    "industry": "Fintech",
    "experienceLevel": "senior"
  }
}
```

Response 200:

Mentor encontrado com sucesso.

Content: application/json | MatchmakingResponse

```
{
    matchId: string;  // ID único desta decisão de matching.
```

mentorId: string; // ID do mentor selecionado.
menteeId: string; // ID do mentorado.
matchScore: number; // Score geral de compatibilidade (0-1).
confidence: string; // Nível de confiança no match.
matchedCriteria: MatchedCriteria;
mentorDetails: MentorMatchDetails;
reasoning: string; // Explicação textual do porquê deste match.
alternativeMentors: Array<AlternativeMentor>; // Outras opções de mentores (top 3).
timestamp: string; // Timestamp da decisão.
}

Example "successfulMatch":


```
{
  "matchId": "550e8400-e29b-41d4-a716-446655440010",
  "mentorId": "550e8400-e29b-41d4-a716-446655440003",
  "menteeId": "550e8400-e29b-41d4-a716-446655440000",
  "matchScore": 0.89,
  "confidence": "high",
  "matchedCriteria": {
    "expertiseMatch": 0.95,
    "availabilityMatch": 0.85,
    "languageMatch": 1,
    "formatMatch": 1,
    "experienceMatch": 0.8
  },
  "mentorDetails": {
    "name": "Carlos Mendes",
    "expertise": [
      "startup-validation",
      "business-model",
      "fundraising",
      "market-strategy"
    ],
    "yearsOfExperience": 12,
    "availability": "high",
    "rating": 4.9,
    "totalMentorships": 67,
    "successRate": 0.94
  },
  "reasoning": "Forte compatibilidade em todas as áreas solicitadas. Mentor tem experiência
comprovada em validação de startups e levantamento de capital.",
  "alternativeMentors": [
    {
      "mentorId": "550e8400-e29b-41d4-a716-446655440004",
      "matchScore": 0.82,
      "reason": "Boa compatibilidade, mas menos experiência em fundraising"
    },
    {
      "mentorId": "550e8400-e29b-41d4-a716-446655440005",
      "matchScore": 0.78,
      "reason": "Excelente em inovação, mas disponibilidade limitada"
    }
  ],
  "timestamp": "2024-10-28T14:30:00Z"
}
```


Response 400:

Response 404:

Nenhum mentor compatível encontrado.

Content: application/json | NoMatchFoundResponse

{
matchId: string; // ID da tentativa de matching.
menteeId: string; // ID do mentorado.
found: boolean; // Se algum match foi encontrado.
reason: string; // Razão de nenhum match ter sido encontrado.
suggestions: Array<string>; // Sugestões para melhorar as chances de match.
checkedMentors: integer; // Número de mentores avaliados.
highestScore: number; // Maior score encontrado.

```
    timestamp: string;
}
```

Response 500:

Response 503:

Profile service indisponível.

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.
}
```

POST /api/v1/matchmaking/batch

Encontrar múltiplos mentores (batch).

Retorna uma lista ordenada de mentores compatíveis, útil quando o mentorship-service precisa de alternativas ou quer apresentar opções ao mentorado.

Request Body:

Content: application/json

All of:

MatchmakingRequest

```
{
    menteeId: string;  // ID do mentorado que precisa de mentor.
    requiredExpertise: Array<string>;  // Áreas de expertise necessárias (ordenadas por
        prioridade).
    preferences: MatchingPreferences;
    urgency?: string;  // Urgência para encontrar mentor (afeta o threshold de score).
    menteeProfile?: MenteeProfileSummary;
    excludeMentors?: Array<string>;  // IDs de mentores a excluir (ex. matches anteriores que não
        deram certo).
    weights?: MatchingWeights;
}
```

and

```
{
    limit: integer;  // Número de mentores a retornar.
    minScore: number;  // Score mínimo aceitável.
}
```

Response 200:

Lista de mentores compatíveis.

Content: application/json | BatchMatchmakingResponse

```
{
    menteeId: string;  // ID do mentorado.
    matches: Array<MatchmakingResponse>;  // Lista de matches ordenada por score (maior
        primeiro).
```

```
    totalEvaluated: integer;  // Total de mentores avaliados.
    timestamp: string;
}
```

Response 400:

Response 404:

Nenhum mentor acima do score mínimo.

Content: application/json | NoMatchFoundResponse

```
{
    matchId: string;  // ID da tentativa de matching.
    menteeId: string;  // ID do mentorado.
    found: boolean;  // Se algum match foi encontrado.
    reason: string;  // Razão de nenhum match ter sido encontrado.
    suggestions: Array<string>;  // Sugestões para melhorar as chances de match.
    checkedMentors: integer;  // Número de mentores avaliados.
    highestScore: number;  // Maior score encontrado.
    timestamp: string;
}
```

Response 500:

POST /api/v1/matchmaking/score

Calcular score de compatibilidade.

Calcula o score de compatibilidade entre um mentorado e um mentor específico. Útil para validar matches ou recalcular após mudanças de perfil.

Request Body:

Content: application/json | ScoreCalculationRequest

```
{
    menteeId: string;  // ID do mentorado.
    mentorId: string;  // ID do mentor.
    requiredExpertise: Array<string>;  // Expertise requerida.
    preferences: MatchingPreferences;
    weights?: MatchingWeights;
}
```

Response 200:

Score calculado com sucesso.

Content: application/json | ScoreCalculationResponse

```
{
    menteeId: string;
    mentorId: string;
    overallScore: number;  // Score geral de compatibilidade.
    criteria: MatchedCriteria;
    recommendation: string;  // Recomendação baseada no score.
    timestamp: string;
}
```

Response 400:

Response 404:

Mentor ou mentorado 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/matchmaking/decisions/{menteeId}

Histórico de decisões de matching.

Retorna o histórico de decisões de matching para um mentorado específico. Útil para analytics e debugging do algoritmo.

Request Parameters:

```
menteeId: string;  // ID do mentorado.
limit: integer;  // Limite de registros.
```

Response 200:

Histórico de decisões.

Content: application/json | MatchingHistoryResponse

```
{
  menteeId: string;
  decisions: Array<MatchingDecision>;
  total: integer;  // Total de decisões no histórico.
}
```

Response 404:

Mentorado não encontrado ou sem histórico.

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 /health

Health check.

Verifica se o serviço está saudável e se pode conectar ao profile-service.

Response 200:

Serviço saudável.

Content: application/json | HealthResponse

```
{
  status: string;  // Status geral do serviço.
  timestamp: string;
  version: string;  // Versão do serviço.
  dependencies: object;
}
```

Response 503:

Serviço não saudável.

Content: application/json | HealthResponse

```
{
  status: string;  // Status geral do serviço.
  timestamp: string;
  version: string;  // Versão do serviço.
  dependencies: object;
}
```

Schemas

MatchmakingRequest

```
{
  menteeId: string;  // ID do mentorado que precisa de mentor.
  requiredExpertise: Array<string>;  // Áreas de expertise necessárias (ordenadas por prioridade).
  preferences: MatchingPreferences;
  urgency?: string;  // Urgência para encontrar mentor (afeta o threshold de score).
  menteeProfile?: MenteeProfileSummary;
  excludeMentors?: Array<string>;  // IDs de mentores a excluir (ex. matches anteriores que não deram
    certo).
  weights?: MatchingWeights;
}
```

MatchingPreferences

```
{
  availableTimes: Array<TimeSlot>;  // Horários disponíveis do mentorado.
  preferredLanguage: string;  // Idioma preferido.
  sessionFormat: string;  // Formato preferido das sessões.
  mentorGender: string;  // Preferência de gênero.
  minMentorExperience: integer;  // Experiência mínima em anos.
  maxMentorExperience: integer;  // Experiência máxima em anos (0 = sem limite).
  mentorLocation: string;  // Preferência de localização (para sessões presenciais).
  minMentorRating: number;  // Avaliação mínima do mentor.
}
```

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.
}
```

MenteeProfileSummary

```
{
  currentRole: string;  // Cargo/função atual.
  targetRole: string;  // Cargo/função desejada.
  experienceLevel: string;  // Nível de experiência.
  industry: string;  // Setor de atuação.
  location: string;  // Localização.
  background: Array<string>;  // Background educacional ou profissional relevante.
}
```

MatchingWeights

```
{
  expertiseWeight: number;  // Peso da compatibilidade de expertise.
```

```
    availabilityWeight: number;  // Peso da compatibilidade de horários.
    experienceWeight: number;  // Peso da experiência do mentor.
    ratingWeight: number;  // Peso da avaliação do mentor.
    languageWeight: number;  // Peso da compatibilidade de idioma.
}
```

MatchmakingResponse

```
{
    matchId: string;  // ID único desta decisão de matching.
    mentorId: string;  // ID do mentor selecionado.
    menteeId: string;  // ID do mentorado.
    matchScore: number;  // Score geral de compatibilidade (0-1).
    confidence: string;  // Nível de confiança no match.
    matchedCriteria: MatchedCriteria;
    mentorDetails: MentorMatchDetails;
    reasoning: string;  // Explicação textual do porquê deste match.
    alternativeMentors: Array<AlternativeMentor>;  // Outras opções de mentores (top 3).
    timestamp: string;  // Timestamp da decisão.
}
```

MatchedCriteria

```
{
    expertiseMatch: number;  // Score de match de expertise (0-1).
    availabilityMatch: number;  // Score de compatibilidade de horários (0-1).
    languageMatch: number;  // Score de compatibilidade de idioma (0-1).
    formatMatch: number;  // Score de compatibilidade de formato (0-1).
    experienceMatch: number;  // Score de compatibilidade de experiência (0-1).
    locationMatch: number;  // Score de compatibilidade de localização (0-1).
    ratingScore: number;  // Score normalizado da avaliação do mentor (0-1).
}
```

MentorMatchDetails

```
{
    name: string;  // Nome do mentor.
    expertise: Array<string>;  // Áreas de expertise do mentor.
    yearsOfExperience: integer;  // Anos de experiência.
    availability: string;  // Disponibilidade atual do mentor.
    rating: number;  // Avaliação média (0-5).
    totalMentorships: integer;  // Total de mentorias realizadas.
    successRate: number;  // Taxa de sucesso das mentorias (0-1).
    languages: Array<string>;  // Idiomas que o mentor domina.
    bio: string;  // Bio curta do mentor.
}
```

AlternativeMentor

```
{
    mentorId: string;  // ID do mentor alternativo.
    matchScore: number;  // Score de compatibilidade.
    reason: string;  // Razão para ser alternativa.
```

```
}
```

BatchMatchmakingResponse

```
{
  menteeId: string;  // ID do mentorado.
  matches: Array<MatchmakingResponse>;  // Lista de matches ordenada por score (maior primeiro).
  totalEvaluated: integer;  // Total de mentores avaliados.
  timestamp: string;
}
```

NoMatchFoundResponse

```
{
  matchId: string;  // ID da tentativa de matching.
  menteeId: string;  // ID do mentorado.
  found: boolean;  // Se algum match foi encontrado.
  reason: string;  // Razão de nenhum match ter sido encontrado.
  suggestions: Array<string>;  // Sugestões para melhorar as chances de match.
  checkedMentors: integer;  // Número de mentores avaliados.
  highestScore: number;  // Maior score encontrado.
  timestamp: string;
}
```

ScoreCalculationRequest

```
{
  menteeId: string;  // ID do mentorado.
  mentorId: string;  // ID do mentor.
  requiredExpertise: Array<string>;  // Expertise requerida.
  preferences: MatchingPreferences;
  weights?: MatchingWeights;
}
```

ScoreCalculationResponse

```
{
  menteeId: string;
  mentorId: string;
  overallScore: number;  // Score geral de compatibilidade.
  criteria: MatchedCriteria;
  recommendation: string;  // Recomendação baseada no score.
  timestamp: string;
}
```

MatchingHistoryResponse

```
{
  menteeId: string;
  decisions: Array<MatchingDecision>;
  total: integer;  // Total de decisões no histórico.
```

```
}
```

MatchingDecision

```
{
  matchId: string;
  mentorId: string;
  matchScore: number;
  requiredExpertise: Array<string>;
  timestamp: string;
  outcome: string;  // Resultado da mentoria (se disponível).
}
```

HealthResponse

```
{
  status: string;  // Status geral do serviço.
  timestamp: string;
  version: string;  // Versão do serviço.
  dependencies: object;
}
```

DependencyHealth

```
{
  status: string;
  responseTime: integer;  // Tempo de resposta em ms.
  lastChecked: string;
}
```

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.
}
```