PROJETO – TuneCatch

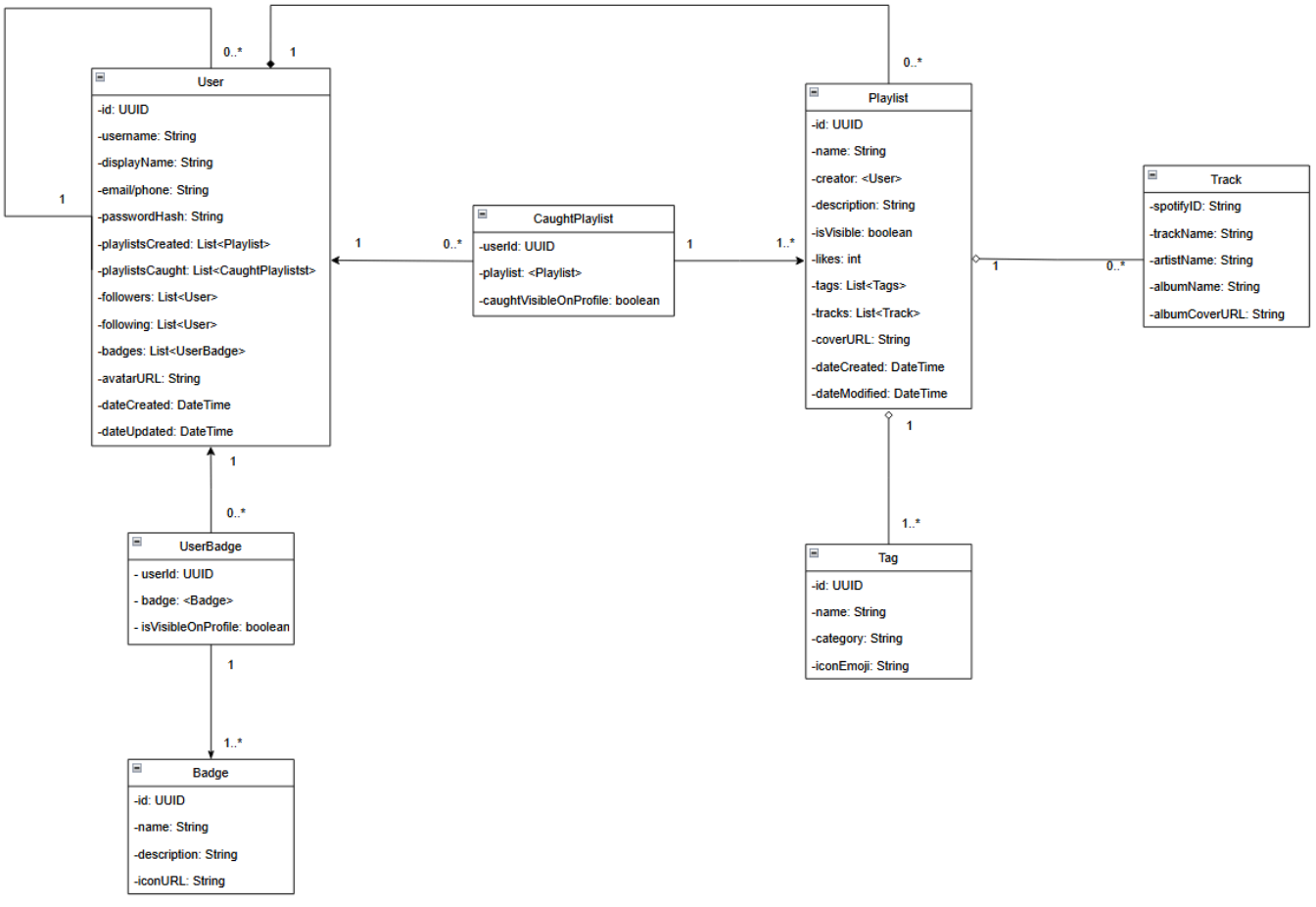
Elementos do TuneCatch –.

O TuneCatch estrutura-se em torno de dois pilares fundamentais: **usuários** e **playlists**.

Os *usuários* são agentes ativos da plataforma – são os perfis que interagem com o conteúdo e entre si. As *playlists*, por sua vez, são as unidades as quais os usuários organizam e compartilham as músicas com base nos contextos emocionais, temporais ou situacionais, atribuindo a elas as *tags*.

No que diz respeito às *músicas*, em vez de armazenar arquivos de áudio ou metadados, cada faixa é referenciada por meio do seu *Spotify Track ID* — um identificador único fornecido pela API do Spotify, o que facilita não apenas o armazenamento, mas também o acesso ao catálogo de músicas disponíveis e sua apresentação para o usuário do sistema.

No entanto, isso não significa que elas serão as únicas classes envolvidas no sistema, haverão classes que as "orbitam" – além da classe `User` e `Playlist`, há as classes `Track`, `Tag` e `Badge` – haverão também as entidades associativas `UserBadge` e `PlaylistCaught`, responsáveis por associar tanto um usuário à uma badge e uma playlist salva à um usuário, permitindo que ele possa modificar seu status de visibilidade para outros perfis.



A partir do Diagrama de Domínio de Entidades, é possível visualizar de forma preliminar como os seus objetos seriam representados dentro sistema da aplicação:

```
User   {
  "id": "u-0001",
  "username": "stargazer",
  "displayName": "marco",
  "email/phone": "marco@dominio.com.br",
  "passwordHash": "$SENHA$455sadFORTE$18$as3aEcXYZ",
  "playlistsCreated": ["p-0000001", "p-0000002"],
  "playlistsCaught":
      [{
        "playlist": {
              "id": "p-0000003",
              "name": "NEW DANCE",
              "creator": "u-0002",
              "description": " ",
              "coverImageURL": "https://br.pinterest.com/pin/alguma_coisa/",
         },
         "caughtVisibleOnProfile": "true"
      },
        "playlist": {
              "id": "p-0000004",
              "name": "just tell me why",
              "creator": "u-0002",
              "description": " ",
              "coverImageURL": "https://br.pinterest.com/pin/alguma_coisa2/",
         },
         "caughtVisibleOnProfile": "false"
      }],
  "followers": ["u-0002", "u-0003"],
  "following": ["u-0002", "u-0003"],
  "userBadges":
      [{
        "badge": {
            "id": "u-0001",
            "name": "Fundador",
            "description": "Fez parte dos 10 primeiros usuários",
            "iconURL": "https://armazenamento/badge/founder-5dsp.jpg"
        },
         "isVisibleOnProfile": "true"
      }],
  "avatarURL": "https://br.pinterest.com/pin/599682506668827088/",
  "dateCreated": 15-02-2025,
  "dataUpdated": 10-04-2025
}
```

```
playlist {
  "id": "p-000001",
  "name": "duskbreaker",
  "creator": "u-0001",
  "description": "i just wanna drive",
  "isVisible": true,
  "catches": 1783,
  "tags": ["t-001", "t-002", “t-003”, “t-004”, “t-005”, “t-006”],
  "tracks":
      [{
        "track”: {
              "spotifyID": "spotify:track:crashing",
              "trackName": "Crashing (with Kalis Uchi)",
              "artistName": "d4vd",
              "albumName": "WITHERED",
              "albumCoverURL": "https://spotify-api/image/d4vd-album2.jpg"
        },
        "track”: {
              "spotifyID": "spotify:track:ilysmih",
              "trackName": "ILYSMIH",
              "artistName": "Kalis Uchi",
              "albumName": "Sincerely,",
              "albumCoverURL": "https://spotify-api/image/kaliuchis-album2.jpg"
        }]
  "coverImageURL": "https://br.pinterest.com/pin/593419688438357287/",
  "dateCreated": 24-05-2025,
  "dateModified": 29-05-2025
}

track {
  "spotifyID": "spotify:track:crashing",
  "trackName": "Crashing (with Kalis Uchi)",
  "artistName": "d4vd",
  "albumName": "WITHERED",
  "albumCoverURL": "https://spotify-api/image/d4vd-album2-cover.jpg"
}

tag {
  "id": "t-001",
  "name": "Late Night Drive",
  "category": "Ocasião",
  "iconEmoji": "🌙🚗"
 }

badge {
  "id": "b-001",
  "name": "Fundador",
  "description": "Fez parte dos 10 primeiros usuários",
  "iconURL": "https://armazenamento/badge/founder-5dsp.jpg"
}
```

Das classes citadas, seriam armazenadas no banco de dados todos os objetos criados a partir de cada uma - com exceção daqueles criados a partir da classe *Track*.

A classe *Track* cria instâncias temporárias das músicas com o objetivo de exibir suas informações na aba de pesquisas e nas playlists – as quais armazenam somente seus identificadores (IDs). A partir desses IDs, a classe Track permite que todos os dados relevantes das músicas sejam apresentados ao usuário, sem a necessidade de armazená-las no banco, já que são obtidas dinamicamente de fontes externas (API do Spotify).

Estas classes irão se relacionar entre si de três maneiras:

1) **Autoassociação**
    a) Classe User associa consigo mesma
        i) Como um usuário pode seguir outros usuários, é necessário armazenar referências tanto daqueles que ele segue quanto dos que o seguem.
2) **Agregação**
    a) Classe Tag é agregada à Playlist:
        i) Um *playlist* deve ter *tags*, no entanto, essas *tags* não dependem do ciclo de vida da *playlist*. Ou seja, mesmo que uma *playlist* seja excluída, as *tags* permanecem no sistema, pois não pertencem exclusivamente a ela – outras playlists ainda podem usá-las.
    b) Classe Track é agregada à Playlist:
        i) As faixas (*tracks*) fazem parte de uma *playlist*, mas não pertencem exclusivamente a ela e não têm seu ciclo de vida controlado por ela – uma mesma *track* pode ser referenciada em múltiplas *playlists*, e continua existindo independentemente da existência ou exclusão de qualquer uma delas.
3) **Composição** –
    a) Classe Playlist é parte do Usuário:
        i) *Playlists* são ligadas diretamente com seu criador – uma entidade da classe usuário. Essa relação indica que cada *playlist* pertence a um único usuário e sua existência está atrelada à existência do usuário e seu ciclo de vida, ou seja, caso ele seja deletado, a playlist atrelada a ele também será.
4) **Entidades Associativas** –
    a) Classe Playlist associada com Classe Usuário:
        i) *Playlists* podem ser curtidas por usuários que não são seu criador, logo, elas se relacionam com esses usuários que curtiram-as por meio de uma entidade associativa

b) Classe Badge associada com Classe Usuário:
    i) Um usuário pode receber *badges*, logo, elas se relacionam com esses usuários por meio de uma entidade associativa

Por fim, por meio do diagrama, é possível verificar quantas instâncias de uma classe se relaciona com outras por meio da cardinalidade dos vínculos entre elas:

- Um único usuário se relaciona com 0 ou mais outros usuários.
    - *Ele pode seguir (ou ser seguido por) nenhum ou muitos usuários.*
- Um único usuário se relaciona com 0 ou mais *playlists*.
    - *Ele pode ter em seu perfil nenhuma (caso ele só queira procurar novas playlists) ou várias playlists (criadas ou curtidas);*
- Um único usuário se relaciona com 0 ou mais *badges*.
    - *Ele pode colocar em seu perfil nenhuma ou várias badges;*
- Uma única playlist se relaciona com 1 ou mais tags.
    - *Uma playlist deve obrigatoriamente ter no mínimo uma tag, mas pode ter inúmeras, à gosto do seu criador;*
- Uma única playlist se relaciona com 0 ou mais faixas musicais.
    - *Uma playlist pode ter nenhuma música (recém-criada) ou várias*.