Projeto de API: BentoTwitter

Introdução

Este projeto será desenvolvido em grupos de 3 pessoas e terá um valor total de 10 pontos. Cada rota que funcionar corretamente e conforme as especificações fornecerá 2 pontos. Logo, para obter a nota máxima, todas as funcionalidades precisam estar implementadas e funcionando.

A nota mínima para aprovação e certificação será de 6,0 pontos.

Após a formação do grupo, os alunos devem escolher um nome para a equipe, pois será esse nome que será chamado durante a apresentação para os gestores do Venturus, que acompanhará o desenvolvimento e resultado na visita à empresa.

Além da apresentação, posteriormente haverá uma code review com o objetivo de identificar pontos de melhoria e sugerir ajustes com feedback para os alunos até o dia 29/11.

Entrega

Até o dia **19/11/2024, às 23h59h**, cada grupo deve entregar um **arquivo TXT** nomeado como "equipe.txt", contendo o nome da equipe e o nome completo de cada integrante e uma **pasta ZIP** nomeada como "codigo_equipe_nome.zip" (onde "nome" corresponde ao nome da equipe).

A pasta ZIP deve conter todo o código desenvolvido, incluindo arquivos de configuração, código fonte e qualquer documentação adicional. Ambos os arquivos devem ser enviados dentro do prazo para avaliação.

Rotas a serem construídas

Usuários

1. Criação de Usuário

   URL: '/usuarios'
   Método: POST
   Corpo da Requisição:

```
{
 "nome": "string",
 "email": "string",
 "senha": "string",
 "nascimento": "YYYY-MM-DD",
 "nick": "string"
}
```

   Respostas Esperadas:

201 Created (sucesso na criação do usuário):

```
{
 "id": "string",
 "nome": "string",
 "email": "string",
 "nick": "string",
 "imagem": "string",
 "nascimento": "YYYY-MM-DD"
}
```

400 Bad Request (*caso algum campo obrigatório esteja ausente, ou caso o e-mail/nick já esteja em uso, ou a idade seja menor que 16 anos*):

```
{ "erro": "Todos os campos são obrigatórios" }
        Ou
{ "erro": "A idade deve ser maior que 16 anos" }
        Ou
{ "erro": "Email já está em uso" }
        Ou
{ "erro": "Nick já está em uso" }
```

500 Internal Server Error (*caso haja um erro ao criar o usuário*):

{ "erro": "Erro ao criar usuário" }

Obs: No campo imagem deve conter uma ULR de uma imagem

2. Listagem de Usuários

URL: '/usuarios'
Método: GET
Parâmetros de Query (opcionais):
search*: Filtro de pesquisa por nome ou nick (busca parcial).*

Exemplo de Requisição:
GET /usuarios?search=joao

Respostas Esperadas:
200 OK (*retorna a lista de usuários que atendem ao critério de busca ou todos se nenhum critério for informado*):

```
[
 {
  "id": "string",
  "nome": "string",
  "email": "string",
  "nick": "string",
  "imagem": "string",
  "nascimento": "YYYY-MM-DD"
 }
]
```

500 Internal Server Error (*caso haja um erro na consulta de usuários*):
{ "erro": "Erro ao buscar usuários" }

3. Detalhes de um Usuário

URL: '/usuarios/:usuario_id'
Método: GET
Parâmetros:
usuario_id: *ID do usuário a ser detalhado*.
Respostas Esperadas:
200 OK (*detalhes do usuário encontrado*):

```
{
 "nome": "string",
 "email": "string",
 "nick": "string",
 "imagem": "string",
```

"nascimento": "YYYY-MM-DD"
}
404 Not Found (*caso o usuário não seja encontrado*):
{ "erro": "Usuário não encontrado" }

4. Atualização de Usuário

URL: '/usuarios/:usuario_id'
Método: PATCH
Corpo da Requisição (*campos opcionais para atualizar*):
{
"nome": "string",
"email": "string",
"nick": "string"
}
Respostas Esperadas:
200 OK (*sucesso na atualização do usuário*):
{
"id": "string",
"nome": "string",
"email": "string",
"nick": "string",
"imagem": "string",
"nascimento": "YYYY-MM-DD"
}
400 Bad Request (*caso nenhum campo seja fornecido para atualização ou caso o e-mail/nick já esteja em uso*):

{ "erro": "Pelo menos um campo deve ser fornecido para atualização" }
Ou
{ "erro": "Email já está em uso" }
Ou
{ "erro": "Nick já está em uso" }
404 Not Found (caso o usuário não seja encontrado):
{ "erro": "Usuário não encontrado" }
500 Internal Server Error (caso haja um erro ao atualizar o usuário):
{ "erro": "Erro ao atualizar usuário" }

Publicações

5. Criação de uma Nova Publicação

URL: ‘/publicacoes’
Método: POST
Corpo da Requisição:

```
{
  "publicacao": "string",
  "usuario_id": "string"
}
```

Regras de Validação:
O campo *publicacao* (conteúdo da publicação) é obrigatório.
O campo *usuario_id* (ID do usuário que está criando a publicação) é obrigatório.
A publicação só será criada se o *usuario_id* corresponder a um usuário existente no banco de dados.

Respostas Esperadas:
201 Created (*sucesso na criação da publicação*):
{ "publicacao_id": "string" }

400 Bad Request (*caso algum campo obrigatório esteja ausente ou o usuário não seja encontrado*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Usuário não encontrado" }

500 Internal Server Error (caso haja um erro ao criar a publicação):
{ "erro": "Erro ao criar publicação" }

6. Listagem de Publicações

URL: /publicacoes
Método: GET
Descrição: Retorna uma lista de todas as publicações, incluindo informações dos usuários que as criaram.
Formato de Resposta:
200 OK (*sucesso na listagem de publicações*):

```
{
  "data": [
    {
      "publicacao_id": "string",
      "publicacao": "string",
```

```
    "usuario_id": "string",
    "nick": "string",
    "imagem": "string",
    "qtd_likes": "number",
    "criado_em": "YYYY-MM-DDTHH:MM:SSZ"
   }
 ],
 "total": "number"
}
```


500 Internal Server Error (*caso haja um erro ao buscar publicações*):

{ "erro": "Erro ao buscar publicações" }

7. Listar Publicações de um Usuário com Comentários e Curtidas

URL: '/publicacoes/de/:usuario_id'
Método: GET
Parâmetros da URL:
usuario_id (obrigatório): ID do usuário cujas publicações serão listadas.
Resposta Esperada:
200 OK:

```
{
 "data": [
  {
   "publicacao_id": "string",
   "publicacao": "string",
   "usuario_id": "string",
   "nick": "string",
   "imagem": "string",
   "qtd_likes": "number",
   "qtd_comentarios": "number",
   "criado_em": "date"
  }
 ],
 "total": "number"
}
```

404 Not Found:

{ "erro": "Usuário não encontrado" }

8. Obter Publicação e Comentários

URL: '/publicacoes/:publicacao_id'
Método: GET
Parâmetros da URL:
publicacao_id (obrigatório): ID da publicação que se deseja obter os detalhes.

Resposta Esperada:
200 OK:

```
{
 "publicacao_id": "string",
 "publicacao": "string",
 "usuario_id": "string",
 "nick": "string",
 "imagem": "string",
 "qtd_likes": "number",
 "criado_em": "date",
 "comentarios": [
  {
   "comentario_id": "string",
   "comentario": "string",
   "usuario_id": "string",
   "nick": "string",
   "imagem": "string",
   "qtd_likes": "number",
   "criado_em": "date"
  }
 ]
}
```

404 Not Found:

```
{ "erro": "Publicação não encontrada" }
```


9. Deletar Publicação e Seus Comentários

URL: '/publicacoes'
Método: DELETE
Corpo da Requisição:

```
{
 "publicacao_id": "string",
 "usuario_id": "string"
}
```

Respostas Esperadas:
200 OK:

```
{ "mensagem": "Publicação deletada com sucesso" }
```


400 Bad Request (caso a publicação ou usuário não sejam encontrados):

```
{ "erro": "Publicação não encontrada" }
```

Ou

```
{ "erro": "Usuário não informado" }
```


403 Forbidden (caso o usuário não tenha permissão para deletar a publicação):

{ "erro": "Usuário não autorizado" }

Comentários

10. Criação de um Novo Comentário em uma Publicação

URL: '/comentarios'
Método: POST
Corpo da Requisição:

```
{
 "publicacao_id": "string",
 "usuario_id": "string",
 "comentario": "string"
}
```

Regras de Validação:
O campo publicacao_id (ID da publicação) é obrigatório.
O campo usuario_id (ID do usuário que está comentando) é obrigatório.
O campo comentario (conteúdo do comentário) é obrigatório.
O comentário só será criado se o usuario_id corresponder a um usuário existente e o publicacao_id corresponder a uma publicação existente no banco de dados.

Respostas Esperadas:
201 Created (sucesso na criação do comentário):
{ "comentario_id": "string" }

400 Bad Request (*caso algum campo obrigatório esteja ausente, o usuário ou a publicação não sejam encontrados*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Usuário não encontrado" }
Ou
{ "erro": "Publicação não encontrada" }

500 Internal Server Error (*caso haja um erro ao criar o comentário*):
{ "erro": "Erro ao criar comentário" }

11. Listagem de Comentários de uma Publicação

URL: '/comentarios'
Método: GET
Parâmetro de Query:
publicacao_id: O ID da publicação cujos comentários serão listados.

Regras de Validação:
O parâmetro publicacao_id é obrigatório.

Formato de Resposta:
200 OK (sucesso na listagem de comentários):

```
{
 "data": [
  {
   "comentario_id": "string",
   "comentario": "string",
   "usuario_id": "string",
   "nick": "string",
   "imagem": "string",
   "criado_em": "YYYY-MM-DDTHH:MM:SSZ"
  }
 ],
 "total": "number"
}
```

400 Bad Request (*caso publicacao_id não seja informado*):
{ "erro": "Publicação não informada" }
500 Internal Server Error (caso haja um erro ao buscar os comentários):
{ "erro": "Erro ao buscar comentários" }

12. Deletar Comentário

URL: '/comentarios'
Método: DELETE
Corpo da Requisição:

```
{
 "comentario_id": "string",
 "usuario_id": "string"
}
```

Respostas Esperadas:
204 No Content (*sucesso na exclusão, sem conteúdo no retorno*):
http (*Sem conteúdo*)
400 Bad Request (*caso o usuário ou o comentário não sejam encontrados*):
{ "erro": "Usuário não encontrado" }
Ou
{ "erro": "Comentário não encontrado" }
403 Forbidden (*caso o usuário não tenha permissão para deletar o comentário*):
{ "erro": "Usuário não autorizado" }

Curtidas

13. Adicionar Curtida a uma Publicação

URL: ‘/curtidas/publicacao’
Método: POST
Corpo da Requisição:

```
{
  "publicacao_id": "string"
}
```

Respostas Esperadas:
200 OK (sucesso ao adicionar a curtida):
{ "qtd_likes": "number" }
400 Bad Request (*caso o publicacao_id não seja informado ou a publicação não seja encontrada*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Publicação não encontrada" }

14. Remover Curtida de uma Publicação

URL: ‘/curtidas/publicacao’
Método: DELETE
Corpo da Requisição:

```
{
  "publicacao_id": "string"
}
```

Respostas Esperadas:
200 OK (sucesso ao remover a curtida):
{ "qtd_likes": "number" }
400 Bad Request (*caso o publicacao_id não seja informado ou a publicação não seja encontrada*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Publicação não encontrada" }

15. Adicionar Curtida a um Comentário

URL: ‘/curtidas/comentario’
Método: POST
Corpo da Requisição:

```
{
  "comentario_id": "string"
}
```

Respostas Esperadas:
200 OK (*sucesso ao adicionar a curtida*):
{ "qtd_likes": "number" }

400 Bad Request (*caso o comentario_id não seja informado ou o comentário não seja encontrado*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Comentário não encontrado" }

16. Remover Curtida de um Comentário

URL: '/curtidas/comentario'
Método: DELETE
Corpo da Requisição:
{
"comentario_id": "string"
}

Respostas Esperadas:
200 OK (*sucesso ao remover a curtida*):
{ "qtd_likes": "number" }

400 Bad Request (*caso o comentario_id não seja informado ou o comentário não seja encontrado*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Comentário não encontrado" }

Seguidores

17. Seguir um Usuário

URL: ‘/seguidores’
Método: POST
Corpo da Requisição:

```
{
  "usuario_id": "string",
  "usuario_a_seguir_id": "string"
}
```

Respostas Esperadas:
201 Created (*sucesso ao seguir o usuário*):
{ "seguidor_id": "string" }

400 Bad Request (*caso algum campo obrigatório esteja ausente, o usuário a ser seguido não seja encontrado ou já esteja sendo seguido*):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Você já segue este usuário" }
Ou
{ "erro": "Você não pode seguir a si mesmo" }
Ou
{ "erro": "Usuário a ser seguido não encontrado" }

18. Deixar de Seguir um Usuário

URL: ‘/seguidores’
Método: DELETE
Corpo da Requisição:

```
{
  "usuario_id": "string",
  "usuario_a_seguir_id": "string"
}
```

Respostas Esperadas:
200 OK (*sucesso ao deixar de seguir o usuário*):
{ "seguidor_id": "string" }
400 Bad Request (caso algum campo obrigatório esteja ausente, o usuário não esteja sendo seguido ou não exista):
{ "erro": "Todos os campos são obrigatórios" }
Ou
{ "erro": "Você não segue este usuário" }

19. Listagem de Seguidores de um Usuário

URL: '/seguidores/:usuario_id'
Método: GET
Parâmetros:
usuario_id: ID do usuário para o qual deseja-se listar os seguidores.
Query Params (*opcionais*):
page: Número da página (padrão: 1).
limit: Número de registros por página (padrão: 10).
Respostas Esperadas:
200 OK (*retorna uma lista paginada dos seguidores*):

```
{
 "data": [
  {
   "seguidor_id": "string",
   "nome": "string",
   "nick": "string",
   "imagem": "string"
  }
 ],
 "total": "number",
 "currentPage": "number",
 "totalPages": "number"
}
```

500 Internal Server Error (*erro na consulta*):
{ "erro": "Erro ao buscar seguidores" }

20. Listagem de Usuários que um Usuário Segue

URL: '/seguidores/seguindo/:usuario_id'
Método: GET
Parâmetros:
usuario_id: ID do usuário para o qual deseja-se listar os usuários seguidos.
Respostas Esperadas:
200 OK (*retorna uma lista dos usuários que o usuário segue*):

```
{
 "data": [
  {
   "usuario_id": "string",
   "nome": "string",
   "nick": "string",
   "imagem": "string"
  }
 ],
```

"total": "number"
}

500 Internal Server Error (*erro na consulta*):
{ "erro": "Erro ao buscar usuários seguidos" }

Links uteis

- Criptografar string : bcryptjs
- Object Relational Mapper (ORM): Sequelize
- Padrões de Projeto: MVC
- Como utilizar o github para trabalho assíncrono
- Anatomia do package.json: Blog Dev