gama academy

API com Node.js e Express

Desafio | **Nível Básico**

E-mail: contato@gama.academy

Website: gama.academy

API com Node.js e Express

Autoria: Hendy Almeida, Vinicius Oliveira

Editora: Lauren da Silveira Francke

Consultor de Acessibilidade: Eduardo Zangrossi Lino

Consultora de Design: Juliana Moraes

Equipe Multidisciplinar:
Natália Gomes

Sumário

Clique nos links para ir ao tópico correspondente:

Objetivos de Aprendizagem

Coloque em prática seus conhecimentos de Back-end, criando uma API do zero utilizando Node.js, Express e Banco de dados MySQL.

Esperamos que ao final do desafio você consiga:

- Explicar o que é uma API e como utilizá-la;
- Construir uma API utilizando boas práticas em Node.js e Express;
- Criar um banco de dados do zero, desde a estrutura DER a criação de tabelas com SQL;
- Conectar e manipular banco de dados através do Sequelize;

Instruções gerais

Chegou o momento de colocar em prática tudo que aprendeu sobre Back-end, bora?

Um grupo de psicólogos se juntaram e criaram a clínica **La Vie - Saúde Mental** que oferece diversos tipos de terapia.

Para ajudar nos atendimentos, eles precisam de uma API que permita criar registros de psicólogos, pacientes e prontuários. Em uma conversa com os Front-end e os PO foram decididos alguns grupos de endpoints que devem ser criados.

Instruções gerais

BANCO DE DADOS

Nossa equipe também ficará responsável por criar o banco de dados que inclui: Criação do DER e do script SQL que gera o banco.

Deve ser analisado os endpoints para seja montado de acordo com a necessidade, observando os dados que constituem a 3 entidades do sistema:

- Pacientes
- Psicólogos
- Atendimentos

Se atentar corretamente à criação das relações das entidades.

Funcionalidades Obrigatórias

1. LOGIN

O sistema deve permitir autenticação dos psicólogos para acessar a plataforma. Os dados de autenticação devem ser um **e-mail válido** e uma **senha** (mínimo 6 caracteres). As informações de autenticação devem ser criadas no mesmo momento que é feito o cadastro do psicólogo.

E para formar esse sistema deve existir os seguintes endpoints:

POST /login

Esse endpoint irá receber no body dois parâmetros que devem ser obrigatórios: email e senha. E devem ser validados de acordo com as informações do banco de dados.

Caso o email não exista, ou a senha não esteja correta para esse e-mail deverá ser exibida como resposta a seguinte mensagem com o status 401:

“E-mail ou senha inválido, verifique e tente novamente”

Caso as informações estejam corretas, deve ser gerado um token JWT que contenha o id, email e nome do psicólogo que fez o login dentro do seu conteúdo. Esse token deve ser enviado como resposta com o status 200.

Funcionalidades Obrigatórias

2. CRUD PSICÓLOGOS

GET /psicologos
Deve ser listado todos os psicólogos cadastrados no banco de dados, exibindo todos os atributos da entidade.
Caso não exista nenhum psicólogo, basta enviar um array vazio como resposta.
Em todos os casos deve ser retornado o status 200

GET /psicologos/:id
Deve devolver um objeto com todas as informações do psicólogo do id informado na url, com exceção da senha. O status deve ser 200 para resposta.
Caso não exista um psicólogo com o id informado deve retornar a seguinte menssagem de erro acompanhada do status 404:

“Id não encontrado”

Funcionalidades Obrigatórias

2. CRUD PSICÓLOGOS

POST /psicologos
Deve receber no body da requisição um objeto com as seguintes propriedades:

- **nome** (Campo string) Ex: Fabricio Oliveira
- **email** (Campo string) Ex: fabricio.psicologo@email.com
- **senha** (Campo string) Ex: 123456
- **apresentacao** (Campo String) Ex: Sou um psicólogo incrível e muito bom!

Todas essas informações são obrigatórias e caso não sejam enviadas devem exibir um status 400 indicando que há um erro na requisição.

Se o cadastro ocorrer corretamente deve devolver a resposta com o status 201 e com as informações que foram criadas no banco.

Funcionalidades Obrigatórias

2. CRUD PSICÓLOGOS

PUT /psicologos/:id
Você irá receber pelo params, o id do psicólogo que será atualizado.

Deve receber no body da requisição um objeto com as seguintes propriedades:

- **nome** (Campo String) Ex: Fabricio Oliveira
- **email** (Campo String) Ex: fabricio.psicologo@email.com
- **senha** (Campo String) Ex: 123456
- **apresentacao** (Campo String) Ex: Sou um psicólogo incrível e muito bom!

Todas essas informações são obrigatórias e caso não sejam enviadas devem exibir um status 400 indicando que há um erro na requisição.

Se a atualização ocorrer corretamente deve devolver a resposta com o status 200 e com as informações que foram atualizadas no banco.

Funcionalidades Obrigatórias

2. CRUD PSICÓLOGOS

DELETE /psicologos/:id

Você irá receber pelo params, o id do psicólogo que será deletado.

Se o id existir deve ser deletado do banco de dados o psicólogo informado e devolve como resposta o status 204

Caso não exista um psicólogo com o id informado deve retornar a seguinte menssagem de erro acompanhada do status 404:

“Id não encontrado”

Funcionalidades Obrigatórias

3. CRUD PACIENTES

GET /pacientes
Deve ser listado todos os pacientes cadastrados no banco de dados, exibindo todos os atributos da entidade.

Caso não exista nenhum psicólogo, basta enviar um array vazio como resposta.

Em todos os casos deve ser retornado o status 200

GET /pacientes/:id
Deve devolver um objeto com todas as informações do paciente do id informado na url. O status deve ser 200 para resposta.

Caso não exista um psicólogo com o id informado deve retornar a seguinte menssagem de erro acompanhada do status 404:

“Id não encontrado”

Funcionalidades Obrigatórias

3. CRUD PACIENTES

POST /pacientes
Deve receber no body da requisição um objeto com as seguintes propriedades:

- **nome** (Campo string) Ex: Fabricio Oliveira
- **email** (Campo string) Ex: fabricio.psicologo@email.com
- **idade** (Campo Date) Ex: 06/12/1997

Todas essas informações são obrigatórias e caso não sejam enviadas devem exibir um status 400 indicando que há um erro na requisição.

Se o cadastro ocorrer corretamente deve devolver a resposta com o status 201 e com as informações que foram criadas no banco.

Funcionalidades Obrigatórias

3. CRUD PACIENTES

PUT /pacientes/:id
Você irá receber pelo params, o id do paciente que será atualizado.

Deve receber no body da requisição um objeto com as seguintes propriedades:

- **nome** (Campo string) Ex: Fabricio Oliveira
- **email** (Campo string) Ex: fabricio.psicologo@email.com
- **idade** (Campo Date) Ex: 06/12/1997

Todas essas informações são obrigatórias e caso não sejam enviadas devem exibir um status 400 indicando que há um erro na requisição.

Se a atualização ocorrer corretamente deve devolver a resposta com o status 200 e com as informações que foram atualizadas no banco.

Funcionalidades Obrigatórias

3. CRUD PACIENTES

DELETE /pacientes/:id
Você irá receber pelo params, o id do paciente que será deletado.

Se o id existir deve ser deletado do banco de dados o paciente informado e devolve como resposta o status 204

Caso não exista um paciente com o id informado deve retornar a seguinte menssagem de erro acompanhada do status 404:

“Id não encontrado”

Funcionalidades Obrigatórias

4. CRUD ATENDIMENTOS

GET /atendimentos
Deve ser listado todos os atendimentos realizados por todos os psicólogos cadastrados no banco de dados, exibindo todos os atributos da entidade.

Caso não exista nenhum atendimento, basta enviar um array vazio como resposta.

Em todos os casos deve ser retornado o status 200

GET /atendimentos/:id
Deve devolver um objeto com todas as informações do atendimento do id informado na url. O status deve ser 200 para resposta.

Caso não exista o atendimento com o id informado deve retornar a seguinte menssagem de erro acompanhada do status 404:

“Id não encontrado”

Funcionalidades Obrigatórias

4. CRUD ATENDIMENTOS

POST /atendimentos

Deve receber no body da requisição um objeto com as seguintes propriedades:

- **paciente_id** (Campo inteiro) Ex:: 1
- **data_atendimento** (Campo Data) Ex: 2020-01-01T10:10:00z
- **observação** (Campo String) - Ex: Descrição do atendimento, pode ser um campo longo.

Além das informações recebidas no body, é preciso pegar o id do psicólogo que está logado para associá-lo a esse atendimento. Lembre-se que essas informações ficam presentes dentro do Token JWT.

Todas essas informações são obrigatórias e caso não sejam enviadas devem exibir um status 400 indicando que há um erro na requisição.

Se o cadastro ocorrer corretamente deve devolver a resposta com o status 201 e com as informações que foram criadas no banco.

Funcionalidades Obrigatórias

5. DOCUMENTAÇÃO DA API

Deve ser gerado uma documentação da api, colocando os endpoints existentes assim como os dados que devem ser passados na requisição e o que será devolvido pelo servidor como resposta.

Para isso pode ser usado o **Insomnia** junto com o plugin **Export HTML Documentation** (Ensinado do material assíncrono), ou o postman.

Funcionalidade Opcional

DASHBOARD

Deverá ser criados um grupo de endpoints a partir da rota **/dashboard** para cada tipo de informação presente nesta lista:

- Número de pacientes
- Número de atendimentos
- Número de psicólogos
- Média de atendimentos por psicólogos

Ficando por exemplo: **/dashboard/numero-paciente**. Os dados a serem retornando podem ser apenas os números dos resultados em si!

Entregáveis

A entrega do desafio é feita através de um formulário na plataforma Gama. Antes de enviar, verifique se o projeto atende aos critérios de avaliação.

Critérios de Avaliação

- Boas práticas em relação ao uso de JS
- Boa organização do projeto usando os princípios do MVC
- Validação dos dados que entram na api
- Feedback de erros para os usuários
- Divisão de tarefas entre os membros da equipe seguindo os princípios da Metodologia Ágil Scrum
- Utilizar boas práticas de versionamento de código com Git

Quais os entregáveis desse desafio?

- Código disponibilizado no Github;
- Páginas publicadas no Github Pages;

Material complementar

Caso você queira aprofundar seu conhecimento, abaixo seguem algumas dicas de materiais complementares:

- Documentação Express

gama.
academy