

DESENVOLVIMENTO WEB



SIMULADOR DE VESTIBULAR ONLINE



Integração Full-Stack (React + FastAPI) com API Externa

Link do Video

<https://drive.google.com/file/d/1joM2INSkX3-yrrig4ECOxgTMPPHgpcpz/view?usp=sharing>

Autores:

Aluno 1: Julio Cesar Farias

Aluno 2: Michael Pariz Pereira

Aluno 3: Gabriel Takeshi

Professor:

*Luiz Antonio
Lima Rodrigues*

Objetivos

- Substituir o conteúdo estático por questões geradas dinamicamente
 - Integrar nosso frontend React com um backend FastAPI que consome uma API externa de IA





Frontend (React):

- O usuário define as opções do simulado no componente “SelecaoSimulado”
- A página detecta que a API deve ser usada
- Ela chama a função apiPost

Backend(FastApi):

- Recebe a requisição no endpoint
- Valida automaticamente os dados recebidos
- Oculta a complexidade e a chave da API externa. O frontend nunca fala diretamente com a IA.

Integração Externa

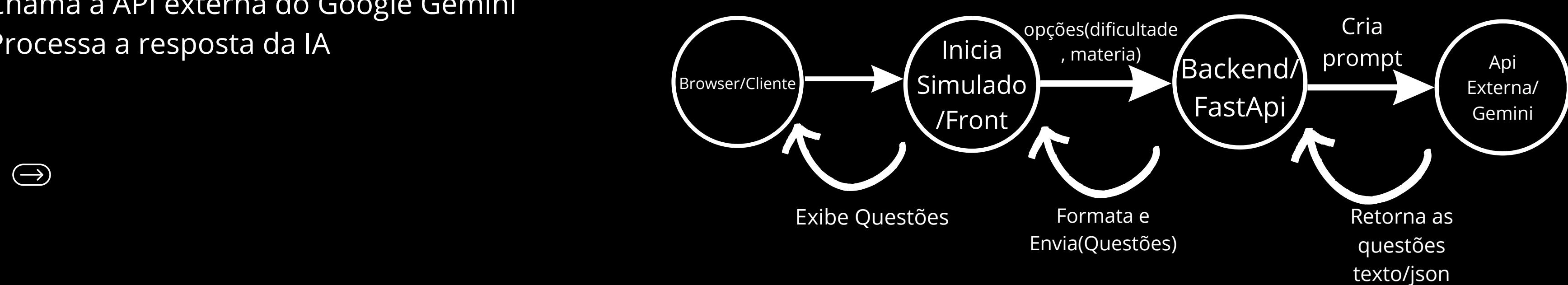
- O backend constrói um prompt detalhado para a IA
- Chama a API externa do Google Gemini
- Processa a resposta da IA

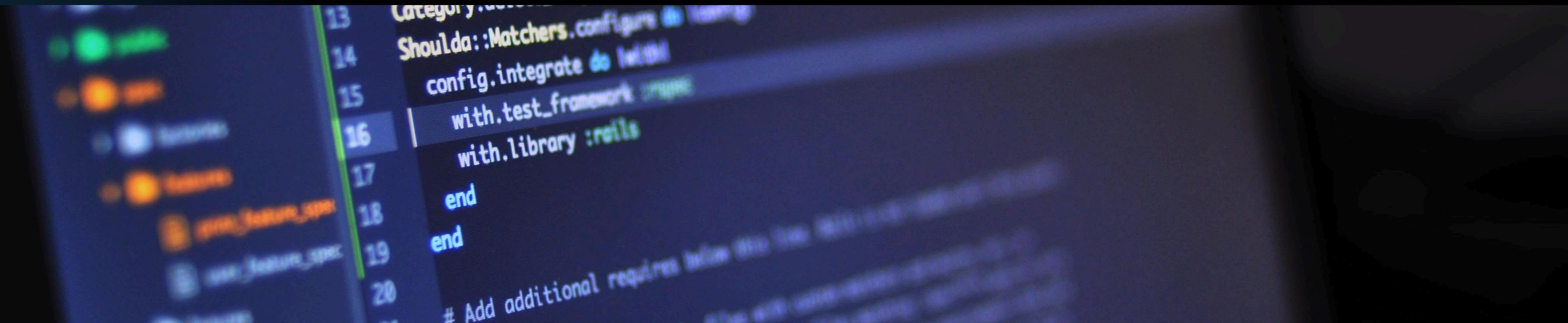
```
import { apiPost } from '../services/apiClient';

const usarAPI = location.state?.usarAPI === true;

const questoesDaApi = await apiPost(
```

```
@app.post("/gerar-simulado/{vestibular_id}", response_model=list[QuestaoResponse])
```





Integração de Segurança (Autenticação e Autorização)



FLUXO DE LOGIN COMPLETO QUE CONECTA O FRONTEND (REACT) E O BACKEND (FASTAPI).

- Backend (/token): Valida o usuário/senha e gera um Token JWT (a "identidade" do usuário).
- Frontend (AuthContext.jsx): Criei um "cérebro" global que salva o Token JWT no localStorage.
- Frontend (ProtectedRoute.jsx): Criei uma "tranca" que redireciona o usuário para /login se ele tentar acessar páginas protegidas sem um token.

SISTEMA DE PERMISSÕES (ROLES) QUE PROTEGE A API.

- Backend (/gerar-simulado): A rota agora exige um Token JWT válido (usando Depends do FastAPI).
- Lógica da Regra: O backend lê a role ("free" ou "premium") de dentro do token.
- Resultado: Se um usuário free pedir mais de 5 questões, a API bloqueia com um Erro 403 (Proibido).



Gerenciamento de Estado da Interface (UX)

ESTADO GLOBAL DA UI

Um segundo estado global, o ThemeContext.jsx, foi implementado.

COMPONENTES DINÂMICOS

- 1. Header Dinâmico (Header.jsx):
- O cabeçalho lê o AuthContext para verificar se o usuário está autenticado.
- O componente renderiza condicionalmente, mostrando "Login" (para visitantes) ou "Usuário" (para autenticados).
- 2. Página de Perfil (UsuarioPage.jsx):
- Uma nova rota protegida (/usuario) foi adicionada.
- A página lê o AuthContext para exibir dados da conta (username, role) e a data do "Último Login".
- Inclui a funcionalidade de Logout.



ENDPOINTS

- Integração com API
- Autenticação
- Autorização

```
# --- Endpoint de Registro (Autenticação) ---
@app.post("/users/", response_model=User)
async def create_user(user: UserCreate):
    """
    Registra um novo usuário.
    """
```

```
# --- Endpoint de Login (Autenticação) ---
@app.post("/token", response_model=Token)
async def login_for_access_token(form_data: Annotated[OAuth2PasswordRequestForm, Depends()]):
    """
    Endpoint de login. Recebe 'username' e 'password' de um formulário.
    Verifica as credenciais e retorna um Token JWT.
    """
```

```
# --- O Endpoint Principal (Autorização) ---
```

```
@app.post("/gerar-simulado/{vestibular_id}", response_model=list[QuestaoResponse])
async def gerar_simulado(
    vestibular_id: str,
    request_data: SimuladoRequest,
    # --- MODIFICAÇÃO CHAVE ---
    # Ao adicionar esta dependência, o FastAPI vai AUTOMATICAMENTE
    # exigir um token válido para este endpoint.
    # Se o token for inválido, ele retorna 401 (Não Autenticado).
    current_user: Annotated[User, Depends(get_current_user)]
):
    """
    Este endpoint recebe os detalhes do simulado, chama a IA
    e retorna uma lista de questões respondidas.
    """
```

