Relatório de Atividades e Solução de Problemas – Projeto SASBApp

27 jul. 2025

1 Relatório de Bugs e Soluções – Configurações de Ambiente

1.1 Dependência Ausente no Backend

- **Sintoma:** Ao tentar executar o comando `python manage.py migrate` pela primeira vez, o sistema retornou um erro fatal: `ModuleNotFoundError: No module named 'django_extensions'`.
- **Diagnóstico:** O projeto utilizava a biblioteca `django-extensions`, porém, ela não estava declarada no arquivo de dependências `requirements.txt`. Isso impedia que o ambiente virtual fosse configurado corretamente com todos os pacotes necessários.
- **Solução:**
  1. A dependência foi instalada manualmente no ambiente virtual com o comando `pip install django-extensions`.
  2. O arquivo `requirements.txt` foi atualizado com o comando `pip freeze > requirements.txt` para incluir a nova dependência e garantir que futuras instalações do projeto não enfrentem o mesmo problema.

1.2 Bloqueio de Requisição por Política de CORS

- **Sintoma:** A primeira tentativa de conectar o Frontend (React) ao Backend (Django) falhou. O console do navegador exibia um erro de CORS (Cross-Origin Resource Sharing) com a mensagem `CORS Missing Allow Origin`.
- **Diagnóstico:** Por padrão, os navegadores implementam a "Política de Mesma Origem", que impede que uma página web (`http://localhost:5173`) faça requisições para um domínio diferente (`http://localhost:8000`). O

backend não estava configurado para permitir explicitamente essa comunicação.

- **Solução:**
  1. Instalou-se a biblioteca `django-cors-headers` no backend.
  2. O app `corsheaders` foi adicionado a `INSTALLED_APPS` no arquivo `settings.py`.
  3. O middleware `corsheaders.middleware.CorsMiddleware` foi adicionado à lista de `MIDDLEWARE`.
  4. A variável `CORS_ALLOWED_ORIGINS` foi configurada em `settings.py` para autorizar explicitamente o endereço do frontend (`http://localhost:5173`).

1.3 Acesso Negado à API (Erro 403 Forbidden)

- **Sintoma:** Após corrigir o erro de CORS, as requisições ainda falharam, mas agora com um novo erro: `403 Forbidden`.
- **Diagnóstico:** O erro não era mais do navegador, mas sim uma recusa do próprio servidor Django. O Django REST Framework (DRF), por padrão, protege os endpoints com a permissão `IsAuthenticated`, exigindo que o usuário esteja logado. A requisição do frontend estava sendo feita como um usuário anônimo.
- **Solução (Temporária):**
  1. O método `get_permissions` na `ServiceViewSet` (arquivo `api/views.py`) foi modificado.
  2. A lógica foi alterada para retornar a permissão `[permissions.AllowAny()]` para ações de leitura (`list`, `retrieve`), tornando o endpoint de listagem de serviços público e resolvendo o erro 403 para esta operação específica.

2 Relatório de Bugs e Soluções – Integração da Funcionalidade de Serviços

2.1 Sessão do Usuário Expirava Inesperadamente

- **Sintoma:** Após um curto período logado (menos de 5 minutos), as chamadas à API começavam a falhar com um erro `401 Unauthorized`, forçando o usuário a fazer login novamente.
- **Diagnóstico:** O `accessToken` gerado pelo `Simple JWT` tinha um tempo de vida padrão de apenas 5 minutos, e não havia um mecanismo de renovação automática no frontend.

- **Solução:**
  - **Backend:** O tempo de vida do `accessToken` foi aumentado para 60 minutos no arquivo `settings.py`.
  - **Frontend:** Foi implementado um interceptor no `api.ts` para capturar erros `401`, usar o `refreshToken` para obter um novo `accessToken` e refazer a requisição original de forma transparente.

2.2 Mensagens de Erro do Formulário Eram Técnicas e em Inglês

- **Sintoma:** O formulário exibia erros genéricos como `Request failed with status code 400` ou mensagens de validação em inglês.
- **Diagnóstico:**
  1. O frontend não extraía a mensagem de erro específica da resposta da API.
  2. A configuração de idioma (`LANGUAGE_CODE`) no `settings.py` do Django estava definida como `en-us`.
- **Solução:**
  1. **Frontend:** O tratamento de erro em `cadaster-service.tsx` foi refatorado para extrair e exibir a mensagem de erro específica do corpo da resposta da API.
  2. **Backend:** A `LANGUAGE_CODE` no arquivo `settings.py` foi alterada para `'pt-br'`.

2.3 Formulário de Cadastro Aceitava Entradas Parcialmente Inválidas

- **Sintoma:** Os campos "Duração" e "Preço" aceitavam entradas como `110 minutos` ou `120,aa`, interpretando apenas a parte numérica.
- **Diagnóstico:** O comportamento padrão das funções `parseInt()` e `parseFloat()` do JavaScript, que são tolerantes a caracteres não numéricos.
- **Solução:** Foi implementada uma validação com Expressões Regulares (Regex) no frontend (`cadaster-service.tsx`) para garantir que os campos contenham apenas formatos numéricos válidos antes da submissão.

3 Relatório de Bugs e Melhorias – Gerenciamento de Serviços (Editar/Excluir)

3.1 Risco de Inconsistência de Dados na Edição de Serviços Agendados

- **Sintoma:** O sistema permitia a edição de um serviço (ex: mudança de duração) mesmo que ele já estivesse vinculado a agendamentos futuros, o que poderia quebrar a integridade da agenda.
- **Diagnóstico:** A lógica de validação para impedir a atualização de serviços com agendamentos futuros não havia sido implementada no backend.
- **Solução:**
  1. **Backend:** O método `update` na `ServiceViewSet` foi modificado para verificar a existência de agendamentos futuros e retornar um erro `400` se a condição for atendida.
  2. **Frontend:** A API foi ajustada para enviar um campo booleano (`can_edit`). O componente `services-table.tsx` agora usa esse campo para desabilitar visualmente o botão de edição.

3.2 Bug Visual: "Flash" Vermelho ao Carregar a Tela de Edição

- **Sintoma:** Ao abrir a página para editar um serviço, uma caixa de mensagem de erro vermelha com o texto "Carregando..." aparecia rapidamente.
- **Diagnóstico:** O componente reutilizava o mesmo estado para feedback de carregamento e de erro, aplicando o estilo de erro indevidamente.
- **Solução:** Foi adicionado um estado booleano dedicado `isLoading` no componente `edit-service.tsx` para exibir um texto de carregamento simples, eliminando o "flash".

3.3 Bug Visual: Botões Desalinhados na Tela de Edição

- **Sintoma:** O botão "Salvar Alterações" estava mais largo que o botão "Cancelar", quebrando a harmonia visual.
- **Diagnóstico:** O componente reutilizável `SubmitButton` possuía uma classe de largura total (`w-full`) fixa.
- **Solução:** O `<SubmitButton>` foi envolvido em uma `<div>` container no arquivo `edit-service.tsx` para limitar o alcance da classe `w-full` e alinhar os botões corretamente.

3.4 Melhoria de UI: Fundo do Modal de Confirmação

- **Sintoma:** O modal de confirmação para exclusão exibia um fundo preto/cinza semi-opaco, que não era visualmente agradável.
- **Diagnóstico:** As classes de CSS no componente `ConfirmationModal.tsx` aplicavam uma cor de fundo em vez de um efeito de desfoque.
- **Solução:** As classes do Tailwind CSS foram ajustadas para diminuir a opacidade do fundo (`bg-opacity-10`) e adicionar a classe `backdrop-blur-sm`, criando um efeito de "vidro fosco".

4 Outros Bugs Identificados e Solucionados

4.1 Validação na Criação de Agendamento

- **Sintoma:** Erro ao criar agendamento com um "Profissional" devido à validação do modelo.

4.2 Atualização da Navbar

- **Sintoma:** A Navbar não atualizava o nome do usuário logado corretamente.

4.3 Visibilidade do Botão de Cadastro

- **Sintoma:** O botão de cadastrar serviço estava visível para usuários não-administradores.

4.4 Permissões de Profissionais

- **Sintoma:** Profissionais tinham permissão para criar agendamentos e visualizar o botão correspondente, o que não era desejado.
- **Sintoma:** Profissionais conseguiam ver agendamentos de outros profissionais.

4.5 Estilo de Botões

- **Sintoma:** Os botões de editar e excluir em agendamentos e serviços estavam sem a cor de fundo.

4.6 Loop de Login

- **Sintoma:** Ocorria um loop infinito na tela de Login.

4.7 Tabela de Equipe: Exibição de Cargo

- **Sintoma:** O cargo do funcionário não era exibido na tabela de equipe.

4.8 Tabela de Equipe: Botão de Cadastro Indevido

- **Sintoma:** Um botão de "Cadastrar funcionário" aparecia na tela de edição de um funcionário.

4.9 Tabela de Equipe: Desaparecimento de Profissional

- **Sintoma:** Um profissional desaparecia da tabela de equipe após ser editado.

4.10 Tabela de Equipe: Erro na Edição de Usuário

- **Sintoma:** Ocorria o erro `No User matches the given query.` ao tentar editar um usuário.

4.11 Tabela de Equipe: Administrador Visível

- **Sintoma:** O administrador do sistema aparecia indevidamente na tabela de equipe.

4.12 Tabela de Equipe: Estilo e Comportamento dos Botões

- **Sintoma:** Perda de estilo, recarregamento da página ao clicar em "Editar" e desalinhamento dos botões de ação.

4.13 Tabela de Equipe: Barras de Rolagem

- **Sintoma:** As barras de rolagem na tabela de equipe estavam sempre visíveis, mesmo sem necessidade.

4.14 Edição de Agendamento: Redirecionamento e Recarregamento

- **Sintoma:** Ao salvar uma edição, o usuário era redirecionado, impedindo a visualização de mensagens de feedback, e a página recarregava completamente.

4.15 Edição de Agendamento: Botão Não Funcional

- **Sintoma:** O botão "Salvar Alterações" parou de funcionar após uma modificação no tratamento de eventos do formulário.

4.16 Edição de Agendamento: Layout e Mensagens

- **Sintoma:** O layout dos botões era inconsistente com outras telas e as mensagens de erro eram genéricas.

4.17 Edição de Agendamento: Conflitos e Restrições

- **Sintoma:** O backend acusava conflitos de horário incorretamente em edições parciais e permitia que profissionais alterassem o responsável pelo agendamento.

4.18 Edição de Agendamento: Validação e Campos

- **Sintoma:** O formulário era submetido mesmo sem alterações, o backend exigia campos desnecessários em atualizações parciais e o campo `notes` causava erros.

4.19 Backend: Erro de Asserção

- **Sintoma:** O campo `notes` foi declarado no serializador, mas não incluído na opção `fields`, causando um erro de asserção no backend.