**1. Login na plataforma**

**História** – Como cidadão ou advogado registrado, quero fazer login para acessar a plataforma com meu perfil.  
**Critérios de aceite**

* Ao informar e‑mail e senha válidos, o sistema gera um token JWT e identifica o papel (usuário ou advogado).
* Com credenciais inválidas, recebo mensagem de erro sem autenticação.  
  **Especificação funcional**
* ~~Formulário solicita e‑mail e senha, valida os campos e envia a requisição para o backend.~~**~~Especificação técnica~~**
* ~~Endpoint POST /login retorna token e role do usuário~~
* ~~Componente de frontend envia email e senha para o endpoint de login~~

**2. Cadastro de usuário (cidadão)**

**História** – Como cidadão interessado em auxílio jurídico, quero registrar uma conta para submeter casos.  
**Critérios de aceite**

* Formulário exige nome, e‑mail válido e senha mínima de 6 caracteres.
* Após registro bem-sucedido, sou redirecionado ao login.  
  **~~Especificação funcional~~**
* ~~Cadastro envia nome, email, senha e define automaticamente o papel USUARIO.~~**~~Especificação técnica~~**
* ~~Endpoint POST /usuarios cria conta de usuário~~
* ~~Formulário de registro chama o endpoint com os dados acima~~

**3. Cadastro de advogado voluntário**

**História** – Como advogado, desejo registrar-me informando OAB para oferecer serviços voluntários.  
**Critérios de aceite**

* Formulário exige nome, e‑mail, senha e número da OAB.
* Após sucesso, sou direcionado para a página de login.  
  **~~Especificação funcional~~**
* ~~Cadastro envia nome, email, senha, oab e define o papel ADVOGADO.~~**~~Especificação técnica~~**
* ~~Endpoint POST /advogados realiza o cadastro do advogado~~
* ~~Formulário de registro envia os dados para esse endpoint~~

**4. Submissão de causa jurídica (usuário)**

**História** – Como usuário, quero cadastrar um caso descrevendo meu problema para receber propostas.  
**Critérios de aceite**

* Campos título e descrição são obrigatórios; status inicial da causa deve ser “ABERTA”.
* Após submeter, recebo confirmação e retorno ao painel.  
  **~~Especificação funcional~~**
* ~~Formulário envia título, descrição e status para o backend com autenticação via token.~~**~~Especificação técnica~~**
* ~~Endpoint POST /causas cria nova causa e exige papel USUARIO~~
* ~~Componente de submissão monta e envia a requisição autenticada~~

**5. Listagem de causas disponíveis (advogado)**

**História** – Como advogado autenticado, quero visualizar todas as causas abertas para avaliar quais posso atender.  
**Critérios de aceite**

* A listagem só é acessível por advogados; usuários não autenticados ou sem permissão são bloqueados.
* É possível filtrar causas por título ou descrição.  
  **~~Especificação funcional~~**
* ~~Tela lista casos com título, descrição, autor e data; inclui busca textual.~~**~~Especificação técnica~~**
* ~~Endpoint GET /causas retorna todas as causas para advogados autenticados~~
* ~~Componente CaseList consome o endpoint, trata autenticação e renderiza resultados filtráveis~~

**6. Consulta ao histórico de causas negociadas (advogado)**

**História** – Como advogado, quero acessar o histórico de casos já negociados ou concluídos.  
**Critérios de aceite**

* Apenas advogados autenticados podem acessar a página de histórico.
* A lista mostra causas anteriormente negociadas com o advogado.  
  **~~Especificação funcional~~**
* ~~Página de histórico reutiliza a listagem com endpoint específico.~~**~~Especificação técnica~~**
* ~~Endpoint GET /causas/historico retorna causas com proposta aceita pelo advogado~~
* ~~Página HistoricoPage consome /causas/historico e aplica verificação de papel~~

**7. Visualização de detalhes da causa (advogado)**

**História** – Como advogado, desejo ver informações completas de uma causa específica para decidir se vou propor um lance.  
**Critérios de aceite**

* Acesso restrito a advogados autenticados.
* Página exibe título, descrição, autor e status da causa.  
  **~~Especificação funcional~~**
* ~~Ao selecionar uma causa, o sistema consulta os detalhes pelo ID.~~**~~Especificação técnica~~**
* ~~Endpoint GET /causas/{id} busca os dados da causa~~
* ~~Página de detalhes realiza a chamada autenticada ao backend e renderiza as informações~~

**8. Envio de lance para causa (advogado)**

**História** – Como advogado, quero enviar um lance com valor e comentário para uma causa a fim de oferecer meus serviços.  
**Critérios de aceite**

* Apenas advogados autenticados podem enviar lance.
* Lance aceita valor numérico e comentário opcional; se enviado, inicia chat com mensagem inicial.  
  **~~Especificação funcional~~**
* ~~Modal de lance coleta valor e comentário e envia a proposta ao backend.~~**~~Especificação técnica~~**
* ~~Endpoint POST /lances cria lance associado à causa e advogado~~
* ~~Modal de frontend envia causaId, valor e comentario na requisição autenticada~~

**9. Gerenciar propostas recebidas (usuário)**

**História** – Como usuário, desejo visualizar e aprovar lances de advogados para iniciar o atendimento.  
**Critérios de aceite**

* Usuário autenticado visualiza todas as propostas pendentes.
* Ao aprovar um lance, o sistema inicia um chat com o advogado e muda o status da causa para “NEGOCIANDO”.  
  **~~Especificação funcional~~**
* ~~Lista de lances pendentes permite aprovar propostas individualmente.~~**~~Especificação técnica~~**
* ~~Endpoint GET /lances/pendentes lista lances para o usuário~~
* ~~Endpoint POST /lances/{id}/aprovar marca a proposta como aceita e altera status da causa~~
* ~~Componente PendingBidsList consome os dois endpoints e abre o chat após aprovação~~

**10. Comunicação por chat entre usuário e advogado**

**História** – Como usuário ou advogado, quero trocar mensagens em tempo real para discutir a causa.  
**Critérios de aceite**

* Apenas participantes do lance aprovado podem visualizar e enviar mensagens no chat.
* Mensagens mostram remetente e são entregues em tempo real via WebSocket.  
  **~~Especificação funcional~~**
* ~~Modal de chat lista mensagens existentes e permite envio de novas.~~**~~Especificação técnica~~**
* ~~Endpoints GET /chats/{chatId}/mensagens e POST /chats/{chatId}/mensagens listam e enviam mensagens~~
* ~~Serviço de chat grava mensagem, identifica remetente pelo papel e publica via WebSocket~~
* ~~Componente ChatModal consome os endpoints e atualiza a interface com as mensagens~~

**11. Painel personalizado conforme papel**

**História** – Como usuário autenticado, desejo ver um painel com funcionalidades específicas ao meu perfil.  
**Critérios de aceite**

* Usuários veem atalhos para submeter casos e listar propostas pendentes.
* Advogados visualizam listagem de casos disponíveis e histórico.  
  **Especificação funcional**
* Layout do dashboard muda dinamicamente de acordo com o papel armazenado na sessão.  
  **Especificação técnica**
* Componente de dashboard renderiza seções distintas para USUARIO e ADVOGADO

QUEBRAR EM 2 historias

**12. Filtro de casos por texto (advogado)**

**História de usuário**  
Como advogado autenticado, quero filtrar casos por título ou descrição para encontrar rapidamente aqueles de interesse.

**Critérios de aceite**

* Campo de busca filtra a lista dinamicamente.
* Limpar filtro restaura a lista original.
* CaseList contém campo Input e botão “Limpar Busca”.
* Filtragem ocorre no lado do cliente sobre dados carregados da API.

**Especificação técnica**

* Não exige chamadas adicionais ao backend.
* Implementado em React com useState e Array.prototype.filter.

Filtrar por usuário que fez