CENTRO UNIVERSITÁRIO UNIRUY
WYDEN CAMPUS SALVADOR/BA

Projeto Web/Mobile Extensionista
JAVA

2025
Salvador/BA

CENTRO UNIVERSITÁRIO UNIRUY
WYDEN CAMPUS SALVADOR/BA

Projeto Web/Mobile Extensionista
AGENDA DE COMPROMISSOS

Sinézio da Silva Ramos Junior – 202302375081
Paulo Henrique Ribeiro Chaves – 202303677308
Bruno Santos Oliveira – 202302375138
Gabriel Salazar Araujo Alcântara - 202302375022

Trabalho para obtenção de nota na disciplina de *"Programação Orientada a Objetos em Java"*.

Orientador(a): Heleno Cardoso

2025
Salvador/BA

Sumário

1. Introdução

No dinâmico cenário da vida moderna, onde a gestão do tempo e a organização de tarefas são cruciais para a produtividade pessoal e profissional, ferramentas que auxiliam no agendamento e acompanhamento de compromissos tornam-se indispensáveis. Diante dessa realidade, o presente trabalho descreve o desenvolvimento de uma aplicação de Agenda de Compromissos em Java.

Esta aplicação foi concebida com o propósito de oferecer uma plataforma intuitiva e eficiente para a criação, visualização e gerenciamento simplificado de eventos e tarefas importantes, de forma singular ou coletiva. O projeto visa abordar o problema comum da desorganização e do esquecimento de compromissos, fornecendo aos usuários um meio prático para registrar detalhes de cada evento.

A motivação para o desenvolvimento desta agenda reside na necessidade de uma ferramenta acessível e funcional que permita um controle básico, mas eficaz, dos compromissos diários. Esta aplicação se destaca pela sua implementação focada nas operações essenciais de CRUD e pela flexibilidade de visualização dos compromissos em diferentes formatos (lista, dia, semana, mês, e o próprio modo calendário). A relevância desta aplicação no mundo real é evidente, pois ela oferece uma solução direta para qualquer indivíduo ou profissional que busca otimizar sua rotina, garantindo que prazos e eventos importantes não sejam perdidos e que a gestão do tempo seja feita de forma mais organizada.

2. Objetivos da Aplicação

A principal meta desta aplicação é fornecer aos usuários uma ferramenta digital intuitiva e eficiente para a organização e gerenciamento de seus compromissos diários, semanais e mensais. Ela visa centralizar as informações de eventos e tarefas, facilitando o planejamento e a visualização da rotina.

a) Funcionalidades Principais:

- **Autenticação de Usuários:**
  - Permitir que novos usuários registrem-se na plataforma com um nome de usuário e senha.
  - Possibilitar que usuários existentes realizem login de forma segura, acessando sua agenda pessoal.
  - Manter a sessão do usuário ativa (persistência do login) para uma experiência contínua.
  - Prover funcionalidade de logout para encerrar a sessão de forma segura.

- **Gerenciamento de Compromissos (CRUD):**
  - Criar (Create): Adicionar novos compromissos com informações detalhadas, incluindo título, descrição, data e hora de início, data e hora de término, e local.
  - Visualizar (Read): Exibir os compromissos em diferentes formatos de calendário:

        - Visualização Mensal: Visão geral dos compromissos no mês.
        - Visualização Semanal: Detalhes dos compromissos na semana.
        - Visualização Diária: Detalhes dos compromissos no dia.
        - Visualização em Lista: Listar todos os compromissos ou filtrados por período.
    - Atualizar (Update): Modificar compromissos existentes, permitindo alterar qualquer uma de suas propriedades.
    - Excluir (Delete): Remover compromissos da agenda de forma permanente.

- **Interatividade do Calendário:**
    - Permitir a criação rápida de compromissos clicando diretamente em uma data no calendário.
    - Possibilitar a edição de compromissos através de um clique no evento.
    - Suportar a movimentação de compromissos por meio de arrastar e soltar (drag-and-drop) entre datas/horários.
    - Permitir o redimensionamento da duração de compromissos diretamente no calendário (arrastar bordas do evento).

- **Gerenciamento de Amigos e Colaboração:**
    - Possibilitar que usuários enviem convites de amizade para outros usuários, facilitando o registro e a conexão.
    - Permitir a visualização de convites de amizade enviados e recebidos, com opções para aceitar ou recusar.
    - Exibir uma lista dos usuários que são amigos do usuário logado.
    - Oferece a funcionalidade de remover um usuário da lista de amigos.
    - Permite que, ao criar ou editar um compromisso, o usuário possa convidar amigos de sua lista para fazerem parte desse compromisso.

- **Experiência do Usuário (UX):**
    - Fornecer um **feedback visual** claro sobre o status das operações (carregamento, sucesso, erro).
    - Utilizar um **layout responsivo** para garantir a usabilidade em diferentes tamanhos de tela (desktop e mobile).

3. Funções/Lista de Eventos (Funcionalidades) – RF / RNF

a) Requisitos Funcionais (RF)

Os Requisitos Funcionais descrevem o que o sistema deve fazer para atender às necessidades do usuário.

Número do Requisito Funcional (RF)	Permissão
1	Registro de novos usuários**,** solicitando um nome de usuário e senha.
2	Autenticação de usuários existentes através de nome de usuário e senha.

3	Criação de novos compromissos**,** incluindo título, descrição, data/hora de início, data/hora de fim e local.
4	Visualização de todos os compromissos do usuário logado em um formato de calendário.
5	Atualização de compromissos existentes**,** alterando suas informações (título, descrição, datas, local).
6	Exclusão de compromissos existentes.
7	O criador de um compromisso adicione ou remova participantes (amigos) a este compromisso.
8	Busca por outros usuários pelo nome de usuário.
9	Envio de solicitações de amizade para outros usuários.
10	Aceitação de solicitações de amizade recebidas.
11	Rejeição de solicitações de amizade recebidas.
12	Visualização da lista de amigos do usuário logado.
13	Desfazer uma amizade existente.
14	Visualização de solicitações de amizade enviadas pelo usuário logado.
15	Apenas o criador de um compromisso possa editá-lo ou excluí-lo.

b) Requisitos Não Funcionais (RNF)

Os Requisitos Não Funcionais descrevem **como** o sistema deve se comportar, focando em atributos de qualidade.

Número do Requisito não Funcional (RNF)	Comportamento
1	A aplicação deverá ser segura, utilizando tokens JWT para autenticação e hasheamento de senhas.
2	A aplicação deverá ser responsiva, adaptando sua interface para funcionar adequadamente em diferentes tamanhos de tela (desktop e dispositivos móveis).
3	A comunicação entre o front-end e o back-end deverá ser feita via API RESTful.
4	O tempo de resposta para operações de CRUD em compromissos e amizades deverá ser otimizado, visando uma experiência fluida para o usuário (exemplo: inferior a 2 segundos).
5	O sistema deverá persistir os dados de usuários, compromissos e amizades em um banco de dados relacional (PostgreSQL).
6	A aplicação deverá lidar com erros de forma controlada, exibindo mensagens informativas ao usuário em caso de falha na comunicação ou validação.
7	A autenticação do usuário deverá ser persistente, permitindo que o usuário permaneça logado mesmo após fechar e reabrir o navegador.

8	A aplicação deverá ter uma base de código modular e extensível, utilizando padrões como injeção de dependência (Spring) e gerenciamento de estado (Pinia).

4. Especificação de Programas

Nesta seção, detalhamos a arquitetura interna e as interações do sistema, fornecendo uma visão aprofundada de como a aplicação de agenda de compromissos é construída.

a) Layout da Tela

A interface gráfica do usuário (GUI) da aplicação é projetada para ser intuitiva e responsiva, garantindo uma experiência consistente em diferentes dispositivos.

i. Tela de Login

- Propósito: Permite que usuários existentes acessem a aplicação com suas credenciais.
- Elementos Visuais:
  - Logomarca e Título da Aplicação: "Agenda de Compromissos" e uma logomarca distintiva.
  - Campos de Entrada:
    - Usuário: Campo de texto para o nome de usuário.
    - Senha: Campo de senha (password).
  - Mensagens de Feedback: Áreas para exibir mensagens de erro (ex: "Usuário ou senha inválidos") ou status de carregamento ("Entrando...").
  - Botão: "Entrar" para submeter as credenciais.
  - Link para Registro: Texto "Ainda não é cadastrado?" com um link "Registre-se." para a tela de cadastro.
- Disposição: A tela de login apresenta um layout com dois painéis: um lado com um logo ilustrativo do Java e outro com o formulário de login centralizado, buscando um design moderno e limpo.

ii. Tela de Registro

- Propósito: Permite que novos usuários criem uma conta na aplicação.
- Elementos Visuais:
  - Logomarca e Título da Aplicação: Similar à tela de login.
  - Campos de Entrada:
    - Usuário: Campo de texto para o nome de usuário desejado.
    - Senha: Campo de senha (password).
  - Mensagens de Feedback: Áreas para exibir mensagens de erro (ex: "Nome de usuário já existe") ou sucesso ("Usuário registrado com sucesso!").
  - Botão: "Registrar" para criar a nova conta.
  - Link para Login: Texto "Já possui conta?" com um link "Logar." para a tela de login.
- Disposição: O layout é uma variação da tela de login, adaptada para o processo de registro, com um painel informativo à direita e o formulário à esquerda, proporcionando uma experiência familiar.

iii. Tela da Agenda (Compromissos)

- Propósito: A interface principal para visualizar, criar, editar e excluir compromissos.
- Elementos Visuais:
  - Barra Superior (Cabeçalho):
    - Logomarca e Título da Aplicação: "Agenda de Compromissos".

        - Informações do Usuário: Exibe o nome de usuário logado (ex: "Logado como: Fulano").
        - Botão de Logout: Permite que o usuário encerre sua sessão.
    - Componente de Calendário (FullCalendar):
        - Visualização Mensal: Exibe um mês completo com eventos marcados.
        - Visualização Semanal: Exibe a semana com detalhes de horários.
        - Visualização Diária: Exibe o dia com detalhes de horários.
        - Visualização em Lista: Lista os eventos em formato de lista.
        - Controles de Navegação: Botões para navegar entre meses/semanas/dias ("Anterior", "Próximo", "Hoje").
        - Criação Rápida: Permite clicar em uma data para abrir o modal de criação de compromisso.
        - Interatividade de Eventos: Eventos clicáveis (para edição), arrastáveis (para mudar data/hora) e redimensionáveis (para ajustar duração).
    - Modal de Compromisso:
        - Propósito: Usado tanto para criar novos compromissos quanto para editar existentes.
        - Campos de Formulário: Título, Descrição, Início (data e hora), Fim (data e hora), Local.
        - Botões: "Criar" / "Salvar Alterações", "Excluir" (apenas em modo edição), "Cancelar".
        - Mensagens de Validação/Erro: Exibe feedback sobre dados inválidos (ex: "A data de término deve ser posterior à data de início").
- Disposição: A tela é dividida em uma barra superior fixa para informações do usuário e logout, e o restante da área dominada pelo componente de calendário, maximizando o espaço para a visualização da agenda. A responsividade garante que a visualização seja adaptada a telas menores.

b) Regras de Negócio (RN)

As regras de negócio definem o comportamento esperado da aplicação e garantem a integridade dos dados e a lógica de funcionamento.

- **RN1: Autenticação de Usuário**
    - RN1.1: O nome de usuário deve ser único no sistema. Não é permitido cadastrar dois usuários com o mesmo nome de usuário.
    - RN1.2: A senha do usuário deve ser hashed (criptografada unidirecionalmente) antes de ser armazenada no banco de dados para segurança.
    - RN1.3: Tentativas de login com nome de usuário ou senha inválidos devem resultar em uma mensagem de erro genérica ("Usuário ou senha inválidos") para evitar vazamento de informações.
    - RN1.4: Após o login bem-sucedido, um token JWT será emitido e deve ser utilizado em todas as requisições subsequentes para acesso a recursos protegidos.

- **RN2: Gerenciamento de Compromissos**
    - RN2.1: Um compromisso deve ter um título.
    - RN2.2: Um compromisso deve ter uma data e hora de início e uma data e hora de término.

- RN2.3: A data e hora de término de um compromisso não pode ser anterior ou igual à data e hora de início.
- RN2.4: Somente o usuário proprietário de um compromisso pode visualizá-lo, editá-lo ou excluí-lo.
- RN2.5: Ao criar um compromisso, a data e hora de início e fim devem ser interpretadas como UTC (Coordinated Universal Time) para evitar problemas com fusos horários.
- RN2.6: A movimentação e o redimensionamento de compromissos via arrastar e soltar no calendário devem atualizar as datas/horas correspondentes no banco de dados.

- **RN3: Segurança e Acesso**
  - RN3.1: Apenas usuários autenticados podem acessar as funcionalidades de gerenciamento de compromissos.
  - RN3.2: Requisições a endpoints protegidos sem um token JWT válido (ou com token expirado/inválido) devem resultar em um status 401 (Unauthorized).
  - RN3.3: Requisições a recursos que o usuário não tem permissão de acessar (ex: tentar modificar o compromisso de outro usuário) devem resultar em um status 403 (Forbidden) ou 404 (Not Found, se não houver acesso).

c) Entidades Envolvidas (Classes)

Nesta seção, descrevemos as classes principais que compõem a arquitetura da sua aplicação, suas responsabilidades e os atributos que possuem. Estas classes correspondem diretamente às entidades de negócio e aos componentes de serviço e controle do sistema.

a. Classes do Back-end (Spring Boot)

As classes do back-end seguem uma arquitetura em camadas (Model-View-Controller com serviços e repositórios), onde cada tipo de classe tem uma responsabilidade bem definida.

i. Modelos (Entidades JPA)

Representam as estruturas de dados persistidas no banco de dados.

- ***Usuário.***
  - Responsabilidade: Representa um usuário do sistema, com informações de autenticação e detalhes para amizades e compromissos. Atua também como *UserDetails* para o Spring Security.
  - Atributos:
    - *id*: *Long* (Chave Primária, gerada automaticamente)
    - *username*: *String* (Nome de usuário, único e não nulo)
    - *password*: *String* (Senha hashada, não nulo)
    - *enabled*: *boolean* (Indica se a conta está ativa)
- ***Compromisso.***
  - Responsabilidade: Representa um evento ou compromisso agendado no sistema.
  - Atributos:
    - *id*: *Long* (Chave Primária, gerada automaticamente)
    - *titulo*: *String* (Título do compromisso, não nulo)
    - *descricao*: *String* (Descrição detalhada)
    - *dataHoraInicio*: *Instant* (Data e hora de início, não nulo)

        - *dataHoraFim*: *Instant* (Data e hora de fim, não nulo)
        - *local*: *String* (Local do compromisso)
        - *criador*: *Usuario* (Usuário que criou o compromisso - relacionamento Many-to-One)
        - *participantes*: *Set<Usuario>* (Conjunto de usuários participantes do compromisso - relacionamento Many-to-Many)
- ***Amizade.***
    - Responsabilidade: Gerencia o relacionamento de amizade entre dois usuários, incluindo o status da solicitação.
    - Atributos:
        - *id*: *Long* (Chave Primária, gerada automaticamente)
        - *solicitante*: *Usuario* (Usuário que enviou a solicitação - relacionamento Many-to-One)
        - *solicitado*: *Usuario* (Usuário que recebeu a solicitação - relacionamento Many-to-One)
        - *status*: *StatusAmizade* (Enum: PENDENTE, ACEITO, RECUSADO, BLOQUEADO - não nulo)
        - *dataSolicitacao*: *LocalDateTime* (Data e hora da solicitação, gerada automaticamente na criação)
        - *dataAtualizacaoStatus*: *LocalDateTime* (Data e hora da última atualização do status, gerada automaticamente na atualização)
- ***StatusAmizade.***
    - Responsabilidade: Enumeração para definir os possíveis status de um relacionamento de amizade.
    - Valores: *PENDENTE*, *ACEITO*, *RECUSADO*, *BLOQUEADO*.

ii. Data Transfer Objects (DTOs)

Classes usadas para transportar dados entre as camadas da aplicação, geralmente entre o cliente e o servidor, ou entre serviços.

- ***AuthRequestDTO***
    - Responsabilidade: Receber credenciais de login do front-end.
    - Atributos: *username: String*, *password: String*.
- ***AuthResponseDTO***
    - Responsabilidade: Enviar o token JWT gerado de volta ao front-end após um login bem-sucedido.
    - Atributos: *token: String*, *type: String* (valor padrão "Bearer").
- ***CompromissoCreateDTO***
    - Responsabilidade: Receber os dados para criar ou atualizar um compromisso, incluindo os IDs dos amigos a serem convidados.
    - Atributos:
        - *titulo: String*
        - *descricao: String*
        - *dataHoraInicio: Instant*
        - *dataHoraFim: Instant*
        - *local: String*
        - *amigoIds: List<Long>* (IDs dos amigos convidados)
- ***UsuarioSummaryDTO***
    - Responsabilidade: Representar uma versão resumida do usuário, geralmente para exibição em listas (ex: amigos, participantes), sem expor dados sensíveis.
    - Atributos: *id: Long*, *username: String*.
- ***SolicitacaoAmizadeDTO***

    - Responsabilidade: Representar uma solicitação de amizade na camada de API, incluindo detalhes resumidos dos usuários envolvidos e o status.
    - Atributos:
        - *id: Long* (ID da entidade Amizade)
        - *solicitante: UsuarioSummaryDTO*
        - *solicitado: UsuarioSummaryDTO*
        - *status: StatusAmizade*
        - *dataSolicitacao: LocalDateTime*
- ***RegistroUsuarioDTO*** (Classe interna em *UsuarioController*)
    - Responsabilidade: Usado especificamente para a requisição de registro de um novo usuário.
    - Atributos: *username: String*, *password: String*.

iii. Repositórios (Spring Data JPA)

Interfaces que estendem *JpaRepository*, fornecendo métodos para operações de persistência de dados (CRUD) e consultas personalizadas.

- ***UsuarioRepository***
    - Responsabilidade: Acesso a dados da entidade *Usuario*.
    - Principais Métodos: *findByUsername(String)*, *findByUsernameContainingIgnoreCaseAndIdNot(String, Long)*.
- ***CompromissoRepository***
    - Responsabilidade: Acesso a dados da entidade *Compromisso*.
    - Principais Métodos: *findByParticipantes_Id(Long)*, *findByIdAndCriador_Id(Long, Long)*.
- ***AmizadeRepository***
    - Responsabilidade: Acesso a dados da entidade *Amizade*.
    - Principais Métodos: *findAmizadeEntreUsuarios(Usuario, Usuario)*, *findBySolicitadoAndStatus(Usuario, StatusAmizade)*, *findBySolicitanteAndStatus(Usuario, StatusAmizade)*, *findByIdAndSolicitado(Long, Usuario)*, *findAmizadeAceitaEntreUsuarios(Usuario, Usuario, StatusAmizade)*, *findAllAmizadesByUsuarioAndStatus(Usuario, StatusAmizade)*.

iv. Serviços (Service Layer)

Contêm a lógica de negócio e coordenam as operações entre os repositórios e os controladores.

- ***UsuarioService***
    - Responsabilidade: Gerenciar a lógica de negócio relacionada a usuários, como o registro de novos usuários e a codificação de senhas.
    - Principais Métodos: *registrarNovoUsuario(Usuario)*.
- ***CustomUserDetailsService***
    - Responsabilidade: Componente do Spring Security para carregar os detalhes de um usuário durante o processo de autenticação.
    - Principais Métodos: *loadUserByUsername(String)*.
- ***CompromissoService***
    - Responsabilidade: Gerenciar a lógica de negócio de compromissos, incluindo criação, listagem, atualização e exclusão, com validações de permissão.

    - Principais Métodos: *salvarCompartilhado(CompromissoCreateDTO, Long)*, *listarPorUsuario(Long)*, *atualizar(Long, CompromissoCreateDTO, Long)*, *deletar(Long, Long)*.
- ***AmizadeService***
    - Responsabilidade: Gerenciar a lógica de negócio de amizades, como envio, aceite, rejeição, desfazimento de amizades e listagem de diferentes tipos de relações de amizade.
    - Principais Métodos: *enviarSolicitacaoAmizade(Long, Long)*, *aceitarSolicitacaoAmizade(Long, Long)*, *rejeitarSolicitacaoAmizade(Long, Long)*, *desfazerAmizade(Long, Long)*, *listarSolicitacoesPendentesRecebidas(Long)*, *listarSolicitacoesPendentesEnviadas(Long)*, *listarAmigos(Long)*.

v. Controladores (REST Controllers)

Expondo os endpoints da API REST para o front-end.

- ***UsuarioController***
    - Responsabilidade: Gerenciar requisições relacionadas a usuários (registro, busca).
    - Endpoints: *POST /api/usuarios/registrar*, *GET /api/usuarios/buscar*.
- ***CompromissoController***
    - Responsabilidade: Gerenciar requisições CRUD para compromissos.
    - Endpoints: *GET /api/compromissos*, *GET /api/compromissos/{id}*, *POST /api/compromissos*, *PUT /api/compromissos/{id}*, *DELETE /api/compromissos/{id}*.
- ***AmizadeController***
    - Responsabilidade: Gerenciar requisições relacionadas a operações de amizade.
    - Endpoints: *POST /api/amizades/solicitar/{solicitadoId}*, *POST /api/amizades/aceitar/{amizadeId}*, *POST /api/amizades/rejeitar/{amizadeId}*, *DELETE /api/amizades/desfazer/{amigoId}*, *GET /api/amizades/meus-amigos*, *GET /api/amizades/solicitacoes/recebidas*, *GET /api/amizades/solicitacoes/enviadas*.
- ***AuthController***
    - Responsabilidade: Gerenciar a requisição de login e geração de token JWT.
    - Endpoints: *POST /api/auth/login*.

vi. Componentes de Segurança

- ***JwtUtil***
    - Responsabilidade: Utilitário para geração, extração e validação de tokens JWT.
- ***JwtRequestFilter***
    - Responsabilidade: Filtro do Spring Security que intercepta requisições HTTP para validar tokens JWT e autenticar o usuário no contexto de segurança.
- ***SecurityConfig***
    - Responsabilidade: Configuração principal do Spring Security, definindo regras de acesso, gerenciamento de sessão,

codificador de senhas, CORS e tratamento de exceções de segurança.

b. Classes do Front-end (Vue.js com Pinia)

As classes do front-end são organizadas em componentes, stores (para gerenciamento de estado) e serviços de comunicação.

i. Componentes (Vue Components)

Unidades reutilizáveis da interface de usuário.

- ***App.vue***
  - Responsabilidade: Componente raiz da aplicação, responsável por carregar a *RouterView* e inicializar a autenticação ao montar.
- ***LoginView.vue***
  - Responsabilidade: Interface e lógica para o processo de login do usuário.
- ***RegisterView.vue***
  - Responsabilidade: Interface e lógica para o registro de novos usuários.
- ***CompromissosView.vue***
  - Responsabilidade: Exibir e gerenciar a agenda de compromissos usando o FullCalendar, incluindo abertura de modais para criação/edição.
- ***CompromissoModal.vue***
  - Responsabilidade: Modal genérico para criação e edição de detalhes de um compromisso.
- ***FriendsListView.vue***
  - Responsabilidade: Exibir a lista de amigos do usuário e permitir desfazer amizades.
- ***FriendRequestsView.vue***
  - Responsabilidade: Exibir solicitações de amizade recebidas e enviadas, permitindo aceitar/rejeitar as recebidas.
- ***FindUsersView.vue***
  - Responsabilidade: Permitir a busca por outros usuários e o envio de solicitações de amizade.

ii. Stores (Pinia Stores)

Gerenciamento de estado centralizado para diferentes partes da aplicação.

- ***authStore.ts*** (***useAuthStore***)
  - Responsabilidade: Gerenciar o estado de autenticação (token, username, status de login/logout, mensagens de erro). Persiste o token no *localStorage*.
  - Atributos (State): *token: string | null*, *username: string | null*, *errorMessage: string | null*, *loading: boolean*.
  - Métodos (Actions): *login(credentials)*, *logout()*, *initializeAuth()*.
  -
- ***friendStore.ts*** (***useFriendStore***)
  - Responsabilidade: Gerenciar o estado e a lógica de negócio relacionada a amigos e solicitações de amizade.
  - Atributos (State): *friends: UsuarioSummaryDTO[]*, *incomingRequests: SolicitacaoAmizadeDTO[]*, *outgoingRequests: SolicitacaoAmizadeDTO[]*, *searchedUsers: SearchedUser[]*, *loadingFriends: boolean*, *loadingIncomingRequests: boolean*,

loadingOutgoingRequests: boolean, loadingSearch: boolean, error: string | null, searchError: string | null, actionError: string | null.

- Métodos (Actions): *fetchFriends(), fetchIncomingRequests(), fetchOutgoingRequests(), searchUsers(termo), sendFriendRequest(targetUserId), acceptFriendRequest(amizadeId), rejectFriendRequest(amizadeId), unfriend(friendUserId), clearSearchedUsers()*.

iii. Serviços de Comunicação

- ***api.ts***
  - Responsabilidade: Configurar e exportar uma instância do Axios para comunicação HTTP com o back-end, incluindo interceptores para adicionar o token JWT e tratar erros de autenticação (401).

d) Tabelas do Banco de Dados

Esta seção detalha as tabelas que compõem o esquema do banco de dados PostgreSQL, incluindo seus campos (colunas) e tipos de dados. As informações são extraídas do diagrama EDR e do código JPA.

usuarios			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
id	BIGINT	Chave Primária, Identificador único do usuário	IDENTITY (gerado automaticamente)
enabled	BOOLEAN	Indica se a conta do usuário está ativa	NOT NULL
password	VARCHAR(255)	Senha hashada do usuário	NOT NULL

amizades			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
id	BIGINT	Chave Primária, Identificador único da amizade	IDENTITY (gerado automaticamente)
data_atualizacao_status	TIMESTAMP WITHOUT TIME ZONE	Data e hora da última atualização do status	
data_solicitacao	TIMESTAMP WITHOUT TIME ZONE	Data e hora da solicitação de amizade	NOT NULL
status	VARCHAR(255)	Status da amizade (PENDENTE, ACEITO, RECUSADO...)	NOT NULL
solicitado_id	BIGINT	Chave Estrangeira para o ID do usuário solicitado	NOT NULL, FOREIGN KEY references usuarios(id)

solicitante_id	BIGINT	Chave Estrangeira para o ID do usuário solicitante	NOT NULL, FOREIGN KEY references usuarios(id)

compromissos			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
id	BIGINT	Chave Primária, Identificador único do compromisso	IDENTITY (gerado automaticamente)
data_hora_fim	TIMESTAMP WITHOUT TIME ZONE	Data e hora de término do compromisso	NOT NULL
data_hora_inicio	TIMESTAMP WITHOUT TIME ZONE	Data e hora de início do compromisso	NOT NULL
descricao	VARCHAR(255)	Descrição do compromisso	
local	VARCHAR(255)	Local do compromisso	
titulo	VARCHAR(255)	Título do compromisso	NOT NULL
criador_id	BIGINT	Chave Estrangeira para o ID do usuário criador	NOT NULL, FOREIGN KEY references usuarios(id)

compromissos_participantes			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
compromisso_id	BIGINT	Chave Estrangeira para o ID do compromisso	NOT NULL, FOREIGN KEY references compromissos(id)
usuario_id	BIGINT	Chave Estrangeira para o ID do usuário participante	NOT NULL, FOREIGN KEY references usuarios(id)
(Primary Key)		Chave Primária composta para a tabela de junção	PRIMARY KEY (compromisso_id, usuario_id)

5. DER ou Diagrama de Classe

a) Artefato Gráfico: Diagrama de Entidade-Relacionamento (DER)

O diagrama abaixo, gerado diretamente do PgAdmin4, representa o Diagrama de Entidade-Relacionamento (DER) do banco de dados da aplicação "Agenda de Compromissos". Ele ilustra as entidades (tabelas) e os relacionamentos existentes entre elas.

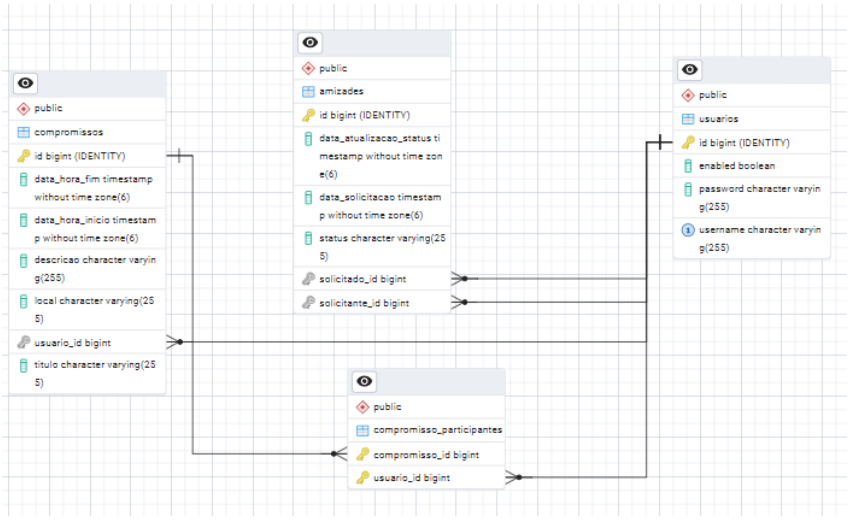

Figura 1: Diagrama Entidade-Relacionamento (DER)

O diagrama apresentado representa um sistema de gerenciamento de usuários, amizades e compromissos. O sistema é composto por quatro entidades principais: "*usuários*", "*amizades"*, "*compromissos"* e "*compromisso_participantes"*.

A entidade "*usuários"* armazena os dados básicos dos usuários do sistema, como "*id"* (identificador único), "*username"* (nome de usuário), "*password"* (senha) e "*enabled"* (indica se o usuário está ativo).

A entidade amizades representa a relação de amizade entre dois usuários. Cada registro contém os campos "*id"*, "*data_solicitacao"* (data em que o pedido de amizade foi feito), "*data_atualizacao_status"* (última atualização do status da amizade) e "*status"* (como "pendente", "aceita", "recusada", etc). Além disso, há dois campos que fazem referência à tabela "*usuários"*: "*solicitante_id"* (quem enviou o pedido) e "*solicitado_id"* (quem recebeu o pedido).

A entidade compromissos representa os eventos criados por um usuário. Cada compromisso possui um "*id"*, "*titulo"*, "*descrição"*, "*local"*, "*data_hora_inicio"*, "*data_hora_fim"* e um campo "*usuario_id"* que indica o usuário criador do compromisso, também referenciando a tabela "*usuários"*.

A entidade "*compromisso_participantes"* é uma tabela associativa que representa os usuários participantes de cada compromisso. Ela possui dois campos: "*compromisso_id"*, que referencia a tabela "*compromissos"*, e "*usuario_id"*, que referencia a tabela "*usuários"*. Essa tabela permite o relacionamento muitos-para-muitos entre usuários e compromissos, ou seja, um compromisso pode ter vários participantes, e um usuário pode participar de vários compromissos.

Portanto, os principais relacionamentos deste modelo são:

- Um usuário pode criar vários compromissos.
- Um compromisso pode ter vários participantes, e um usuário pode participar de vários compromissos.
- Um usuário pode se relacionar com outros por meio de amizades, podendo ser tanto o solicitante quanto o solicitado.

b) Dicionário de Dados (DD):

O Dicionário de Dados detalha cada campo das tabelas do banco de dados, incluindo seu significado, tipo de dado e restrições. Ele fornece uma visão clara da estrutura lógica e física dos dados.

usuarios			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
id	BIGINT	Chave Primária, Identificador único do usuário	IDENTITY (gerado automaticamente)
enabled	BOOLEAN	Indica se a conta do usuário está ativa	NOT NULL
password	VARCHAR(255)	Senha hashada do usuário	NOT NULL
Username	VARCHAR(255)	Nome de usuário hashada	NOT NULL

amizades			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
id	BIGINT	Chave Primária, Identificador único da amizade	IDENTITY (gerado automaticamente)
data_atualizacao_status	TIMESTAMP WITHOUT TIME ZONE	Data e hora da última atualização do status	
data_solicitacao	TIMESTAMP WITHOUT TIME ZONE	Data e hora da solicitação de amizade	NOT NULL
status	VARCHAR(255)	Status da amizade (PENDENTE, ACEITO, RECUSADO...)	NOT NULL
solicitado_id	BIGINT	Chave Estrangeira para o ID do usuário solicitado	NOT NULL, FOREIGN KEY references usuarios(id)
solicitante_id	BIGINT	Chave Estrangeira para o ID do usuário solicitante	NOT NULL, FOREIGN KEY references usuarios(id)

compromissos			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
id	BIGINT	Chave Primária, Identificador único do compromisso	IDENTITY (gerado automaticamente)

data_hora_fim	TIMESTAMP WITHOUT TIME ZONE	Data e hora de término do compromisso	NOT NULL
data_hora_inicio	TIMESTAMP WITHOUT TIME ZONE	Data e hora de início do compromisso	NOT NULL
descricao	VARCHAR(255)	Descrição do compromisso	
local	VARCHAR(255)	Local do compromisso	
titulo	VARCHAR(255)	Título do compromisso	NOT NULL
criador_id	BIGINT	Chave Estrangeira para o ID do usuário criador	NOT NULL, FOREIGN KEY references usuarios(id)

compromissos_participantes			
Nome da Coluna	Tipo de Dado no Banco de Dados	Descrição	Restrições
compromisso_id	BIGINT	Chave Estrangeira para o ID do compromisso	NOT NULL, FOREIGN KEY references compromissos(id)
usuario_id	BIGINT	Chave Estrangeira para o ID do usuário participante	NOT NULL, FOREIGN KEY references usuarios(id)
(Primary Key)		Chave Primária composta para a tabela de junção	PRIMARY KEY (compromisso_id, usuario_id)

6. Aplicação Web/Mobile

Esta seção apresenta a aplicação web desenvolvida, "Agenda de Compromissos", destacando suas funcionalidades principais e a experiência do usuário. A aplicação é construída com Vue.js 3, utilizando Pinia para gerenciamento de estado e Vue Router para navegação, garantindo uma interface reativa e dinâmica que se comunica com o back-end Spring Boot.

a) Menu/Submenu

A aplicação "Agenda de Compromissos" oferece um menu de navegação intuitivo, acessível após a autenticação do usuário. Este menu direciona para as principais funcionalidades, garantindo que o usuário possa transitar facilmente entre as diferentes áreas da aplicação.

Figura 2: Barra de navegação da tela principal. Agenda de Compromissos

Descrição:

- Identificação do Usuário: No cabeçalho, o nome de usuário logado é exibido, proporcionando uma experiência personalizada.

- Links de Navegação Principais:
  - "Meus Amigos": Direciona para a tela onde o usuário pode visualizar sua lista de amigos atuais.
  - "Solicitações": Leva à tela de gerenciamento de solicitações de amizade, tanto as recebidas quanto as enviadas.
  - "Encontrar Amigos": Permite que o usuário pesquise outros usuários da plataforma para enviar novas solicitações de amizade.
- Botão de Ação "LOGOUT": Um botão de fácil acesso para que o usuário possa encerrar sua sessão de forma segura.

b) Telas Funcionais

A aplicação é composta por diversas telas funcionais, cada uma projetada para uma tarefa específica, oferecendo uma experiência de usuário clara e direcionada.

i. Tela de Login

Esta é a porta de entrada para a aplicação, projetada para ser simples e direta, permitindo que o usuário acesse sua agenda.

Descrição: A tela de login é composta por campos para *Usuário* e *Senha*, um botão *Entrar* e um link para o registro, caso o usuário ainda não possua uma conta. A disposição visual busca um equilíbrio entre a identidade visual do projeto e a funcionalidade.

Figura 3: Tela de Login. Agenda de Compromissos

ii. Tela de Registro

Para novos usuários, a tela de registro permite a criação de uma conta na plataforma.

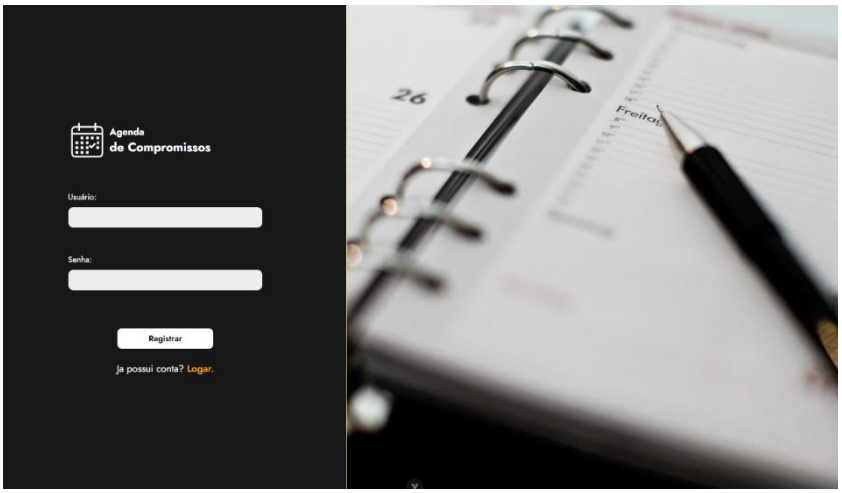

Figura 4: Tela de Registro. Agenda de Compromissos

Similar à tela de login, esta tela oferece campos para *Usuário* e *Senha*. Após o preenchimento, o botão *Registrar* inicia o processo de cadastro. Um link para a tela de login também está disponível para usuários que já possuem uma conta.

iii. Tela Principal - Agenda de Compromissos

A tela principal da aplicação, onde o usuário visualiza e interage com seus compromissos.

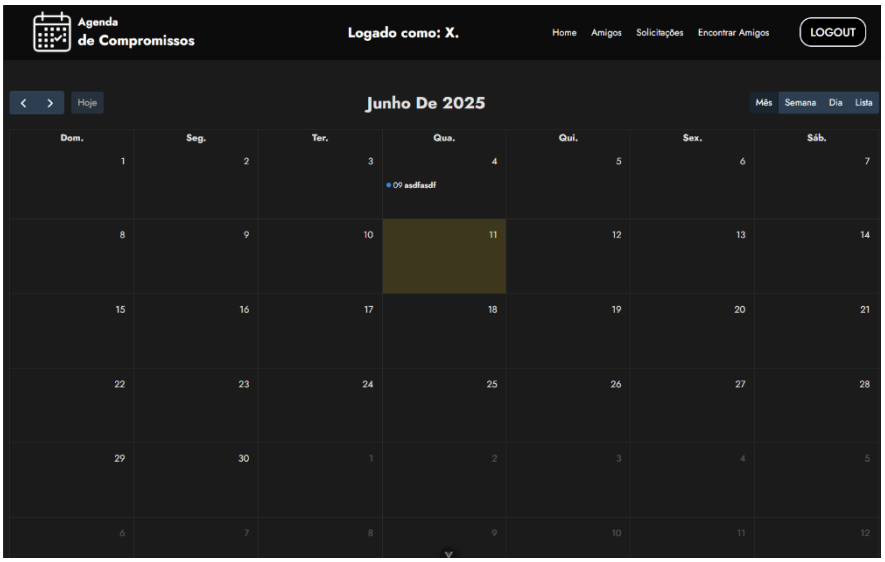

Figura 5: Tela principal. Agenda de Compromissos

Centralizada no calendário interativo FullCalendar, esta tela permite uma visualização diária, semanal ou mensal dos compromissos. Os usuários podem clicar em datas para adicionar novos eventos, e arrastar ou redimensionar compromissos existentes diretamente no calendário, proporcionando uma experiência altamente visual e intuitiva.

iv. Tela de Meus Amigos

Esta tela permite ao usuário gerenciar suas conexões dentro da aplicação.

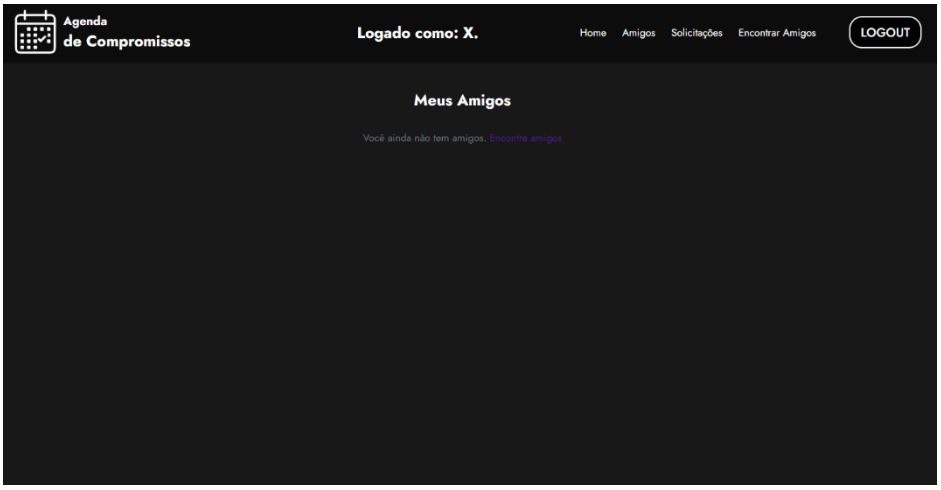

Figura 6: Tela de amigos. Agenda de Compromissos

Descrição: Exibe uma lista clara dos usuários que são amigos do usuário logado. Para cada amigo, há uma opção para "Desfazer Amizade", permitindo a remoção de conexões quando desejado.

v. Tela de Solicitações de Amizade

Gerencia as interações sociais pendentes, dividindo-as em recebidas e enviadas.

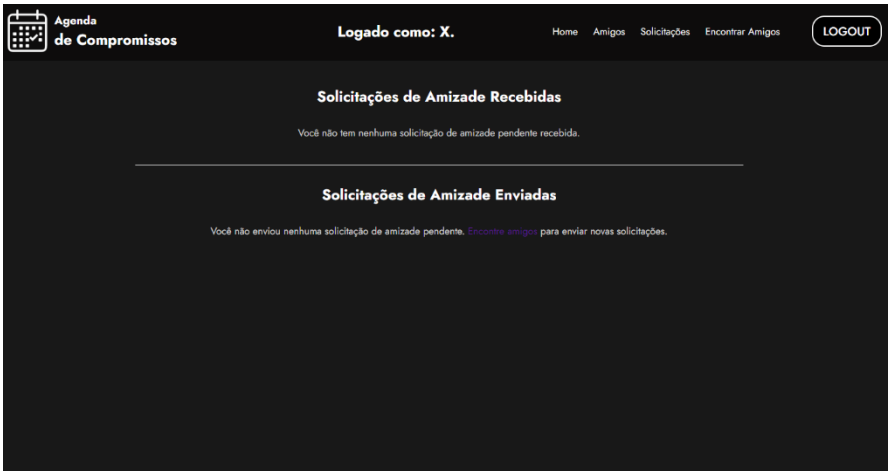

Figura 7: Tela de solicitações. Agenda de Compromissos

Descrição: A tela é dividida em duas seções: "Solicitações de Amizade Recebidas", onde o usuário pode aceitar ou rejeitar pedidos de outros usuários, e "Solicitações de Amizade Enviadas", que mostra o status dos pedidos feitos pelo próprio usuário.

vi. Tela de Encontrar Amigos

Facilita a expansão da rede de contatos do usuário na plataforma.

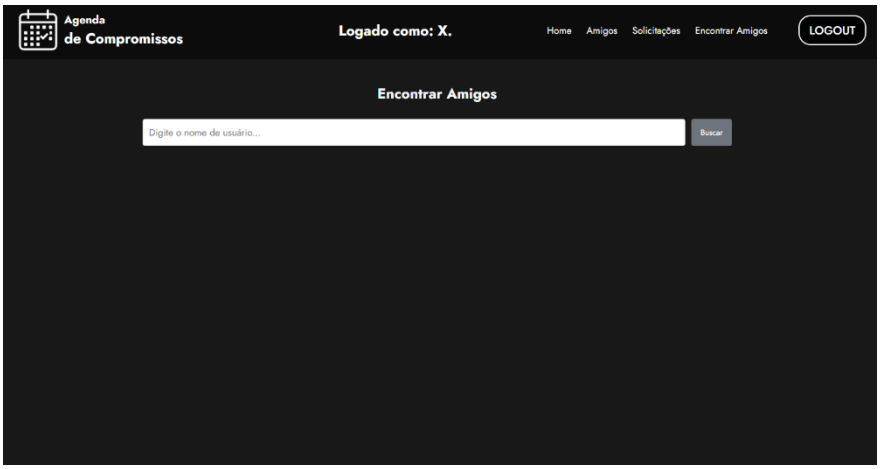

Figura 8: Tela de Encontrar amigos. Agenda de Compromissos

Descrição: Apresenta uma barra de busca onde o usuário pode digitar nomes de usuários para encontrar novas pessoas. Os resultados da busca são exibidos com ações contextuais, como "Adicionar Amigo", "Solicitação Enviada", "Amigos" ou "Responder Solicitação", guiando o usuário na interação.

c) Telas de Diálogo

A aplicação utiliza diálogos e mensagens de feedback para guiar o usuário, confirmar ações e informar sobre o sucesso ou falha das operações.

1. Diálogos no Registro do Usuário.

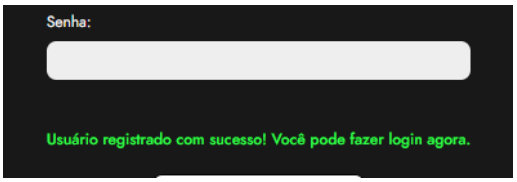

Figura 9: Diag. Sucesso no Registro. Agenda de Compromissos

Figura 10: Diag. Erro de nome do usuário. Agenda de Compromissos

2. Dialogo no Login do Usuário.

Figura 11: Diag. Credenciais inválidas. Agenda de Compromissos

3. **Criação, Atualização e Exclusão de Compromisso.**

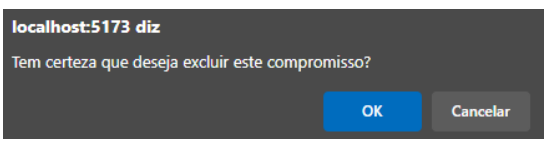

Figura 12: Diag. Confirmação de exclusão do compromisso. Agenda de Compromissos

Figura 13: Diag. Sucesso de exclusão do compromisso. Agenda de Compromissos

Figura 14: Diag. Sucesso de atualização do compromisso. Agenda de Compromissos

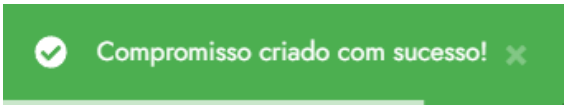

Figura 15: Diag. Sucesso de criação e compartilhamento do compromisso. Agenda de Compromissos

Descrição:

- Mensagens de Erro (***error-message***): Exibidas em vermelho ou cores de alerta, informam o usuário sobre falhas (ex: credenciais incorretas, validação de formulário, erros de conexão). São dinâmicas e mostram a causa específica do problema.
- Mensagens de Sucesso (***success-message***): Apresentadas em verde ou cores que indicam sucesso, confirmam que uma operação foi concluída com êxito (ex: registro de usuário, criação de compromisso).
- Diálogos de Confirmação (***confirm()***): Utilizados antes de ações irreversíveis (ex: "Desfazer Amizade", "Excluir Compromisso") para garantir que o usuário realmente deseja prosseguir.
- Alertas (***alert()***): Usados para mensagens rápidas de feedback após certas operações, complementando as mensagens visuais.
- Estados de Carregamento: Botões e seções da tela exibem um estado de "Carregando..." ou "Processando..." para indicar que uma operação está em andamento, melhorando a percepção de desempenho para o usuário.

d) Layout Relatórios

A natureza da "Agenda de Compromissos" é predominantemente visual e interativa no próprio calendário. Embora não haja uma seção dedicada a "relatórios" tabulares ou gráficos tradicionais, a visualização principal de compromissos atua como o principal "relatório" do usuário sobre sua agenda, especialmente através de sua **função de listagem de compromissos da semana vigente**.

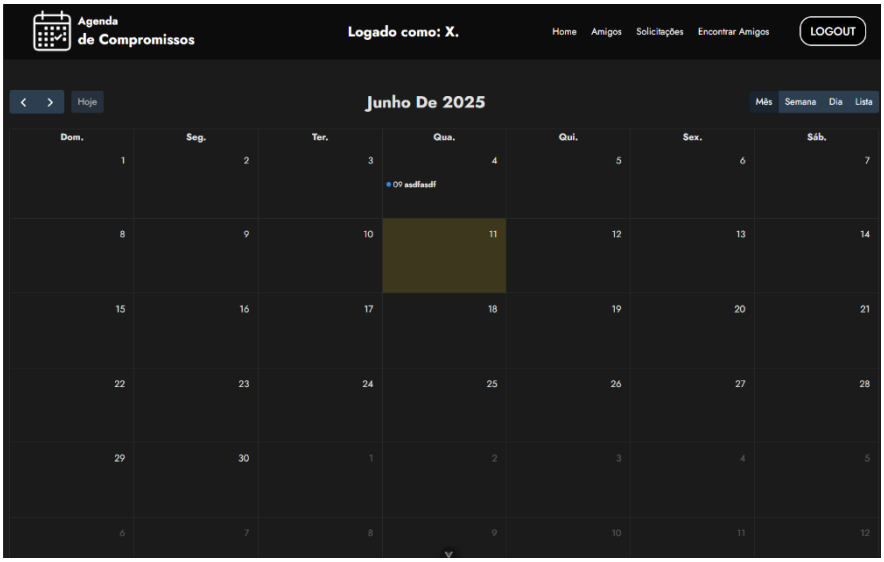

Figura 16: Tela principal. Agenda de Compromissos

Descrição:

- **Visualização de Calendário como Relatório Interativo:** O "FullCalendar", na tela principal, serve como o painel de "relatórios" mais importante. Ele organiza os compromissos por data e hora, permitindo que o usuário visualize rapidamente sua disponibilidade e eventos futuros ou passados de forma interativa.
- **Função de Listagem Semanal (Relatório Requerente):** O calendário oferece uma opção crucial de visualização:
    - **Lista Semanal (`listWeek`):** Esta é a principal funcionalidade de "relatório" que a aplicação oferece. Ela apresenta os compromissos da semana vigente em um formato de lista cronológica clara e concisa. Isso permite ao usuário revisar rapidamente todos os seus compromissos para os próximos dias sem a complexidade visual da grade do calendário, funcionando como um relatório direto de sua agenda semanal.
- **Outras Visualizações Auxiliares:** Além da lista semanal, o calendário também oferece outras opções para uma visão holística:
    - **Mês (`dayGridMonth`):** Para uma visão geral rápida dos compromissos ao longo do mês.
    - **Semana (`timeGridWeek`):** Para uma visão detalhada da semana, com horários específicos.
    - **Dia (`timeGridDay`):** Para o planejamento diário detalhado.
- **Claridade e Intuitividade:** A codificação de cores e a disposição dos eventos no calendário, em todas as visualizações, são projetadas para facilitar a identificação rápida dos compromissos. Isso permite que o usuário utilize a agenda como um "relatório" visual eficiente e fácil de interpretar sobre seus compromissos.

7. Links do repositório GitHub.

- Front-End: https://github.com/GabrielSalazar29/frontAgenda.git
- Back-End: https://github.com/GabrielSalazar29/projectJava.git