Relatório da Entrega 5 - AGENDUSP

Projeto de Laboratório de Orientação a Objetos

André Akira Horigoshi Maximino

Diego Fontes de Avila

Gabriel Lopes Prodossimo

João Victor Costa Teodoro

Matheus Davi Leão

Sophia Soares Mariano

Professor Denis Deratani Mauá

MAC 0321

2025

Implementação do requisito 4 do projeto

Requisito 4 - O AGENDUSP utiliza a inteligência artificial Ollama, integrada ao backend. Assim, o app é capaz de ler os compromissos semanais do usuário, fornecer um resumo sobre os eventos semanais e suas descrições, incluindo cancelamentos e as enquetes de escolha de horário das reuniões. Depois de capturar essas informações, um texto contendo esse resumo é mostrado para o usuário no front-end através dos componentes *AIResponseVisualization* e *WeekView*.

Interface gráfica e frontend

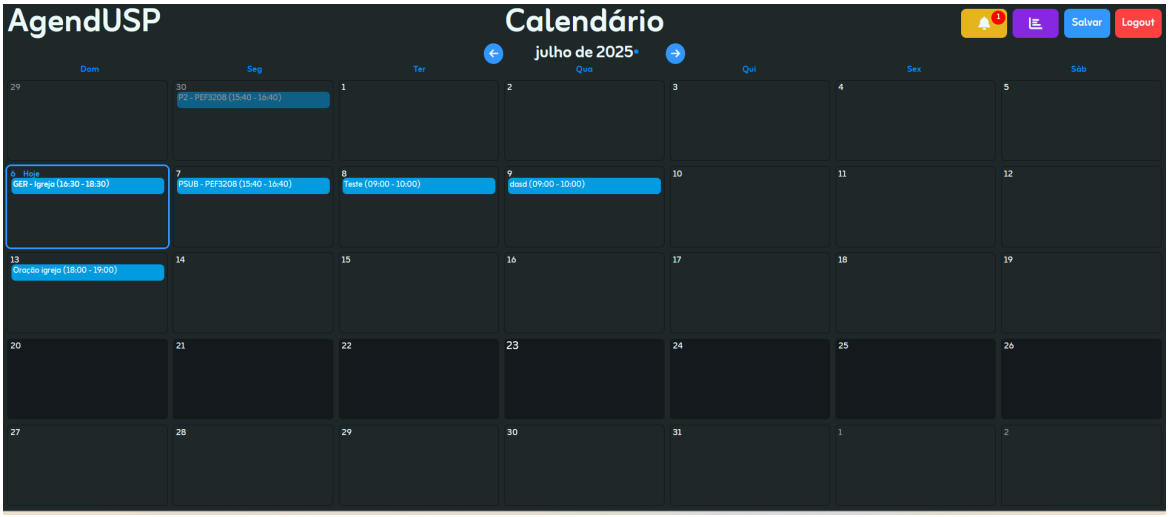
As implementações refatoradas são os arquivos .js responsáveis pela resposta da IA, a página principal do calendário, a página de visualização semanal e todos os arquivos responsáveis pela implementação e visualização das enquetes. Para a implementação da resposta da IA, são utilizados os seguintes arquivos:

- *AIResponseVisualization*: Localizado na pasta **components-> AIResponseVisualization**, pega o calendarId e firstDay vindo da navegação através do useLocation do React Router. Então, uma requisição GET é feita para o backend da IA, que retorna o relatório gerado, o qual é convertido para texto e apresentado da forma adequada na página html.
- Index: Localizado na pasta **components->AIResponseVisualization**, reexporta o componente *AIResponseVisualization* para importar o componente utilizando apenas o diretório.

Para a implementação das enquetes, são utilizados os arquivos:

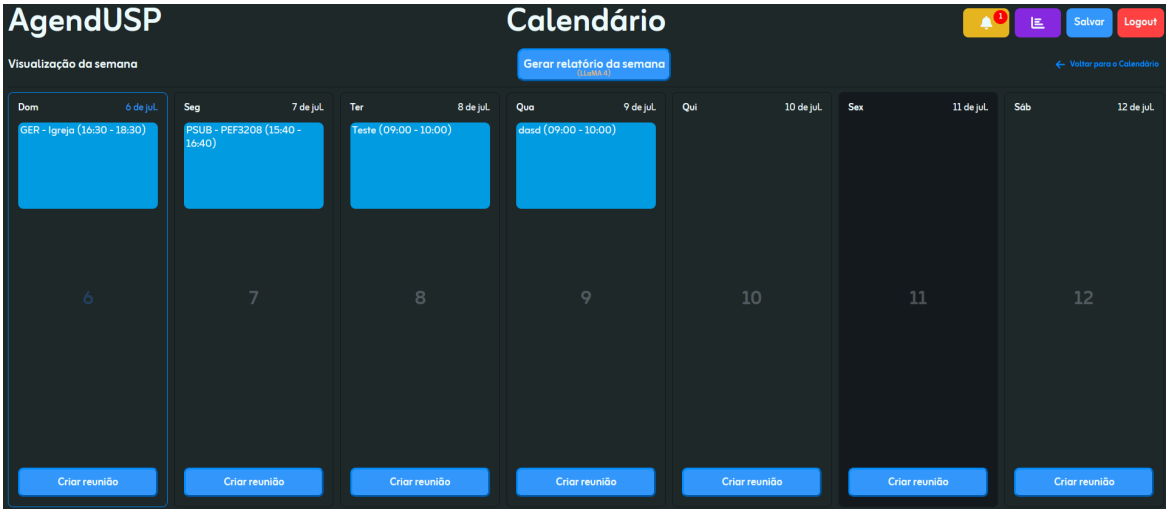
- *VoteMenu*: Localizado na pasta **components->VoteMenu**, pega o calendarId e eventId através do useLocation do React Router. Então, faz duas requisições GET: uma para buscar a enquete e outra para buscar os detalhes do evento. Logo após isso, se não houver a enquete, mostra uma mensagem informando que não há. Caso contrário, se houver, exibe uma lista de horários disponíveis, mostrando o horário e a quantidade de votos de cada opção. Na etapa de seleção e votação, o usuário pode selecionar horários clicando nos itens da lista. Feito isso, o componente envia um voto para cada horário selecionado usando a API /poll/vote.

Quanto à interface gráfica que o usuário tem acesso, apresenta essa visualização:



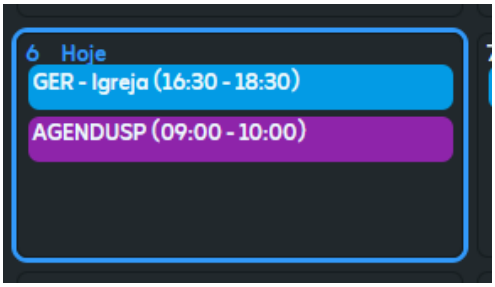
O quadrado em destaque representa o dia de hoje. Esses quadrados em azul são os eventos, mas eles podem assumir qualquer cor que o usuário selecionar.

Ao clicar no texto **AgendUSP** do cabeçalho, ele volta para a página inicial do calendário.

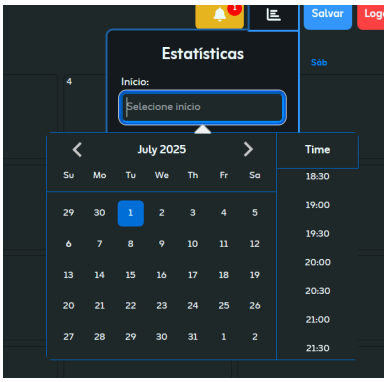


Ao clicar em um dia qualquer, a interface passa a mostrar a semana que contém aquele dia. Cada dia possui o botão de “criar reunião”.

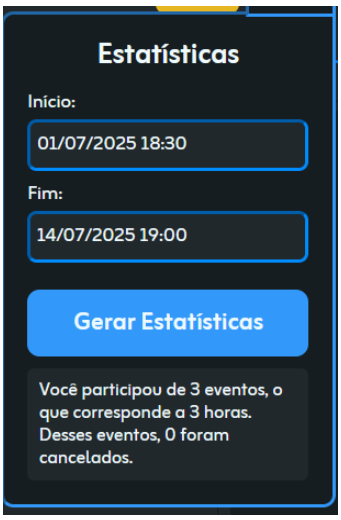
O usuário pode escolher o nome do evento, a data e horário que ele acontecerá, a cor que ele terá na tela do AGENDUSP, o local, e ainda adicionar e remover os convidados que possuem conta no calendário. Caso seja de desejo do usuário convidar outro usuário, este deve estar conectado ao AGENDUSP no próprio computador de uso (abrir em duas abas). Dessa forma, é necessário utilizar dois emails diferentes (veja instruções de compilação para melhor compreensão).



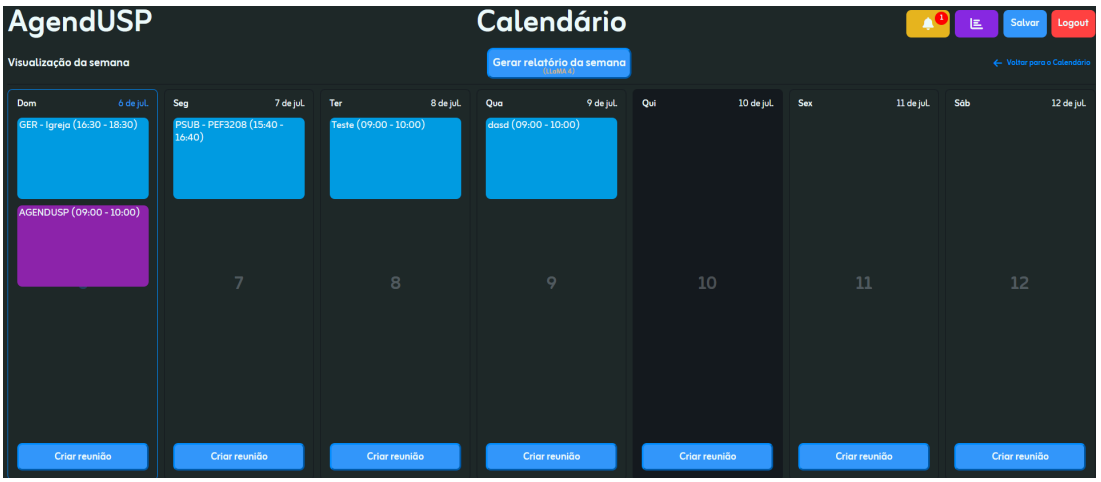
Após a confirmação, se pode ver que o evento aparece na página principal do calendário.

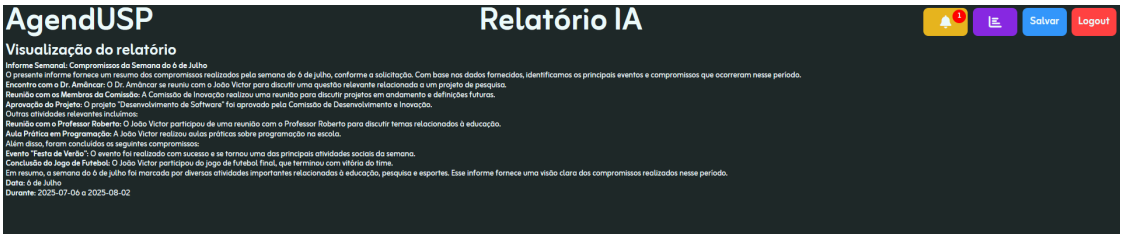


No botão roxo, selecionando as datas de início e fim do intervalo desejado, é possível ver as estatísticas de reuniões do usuário.

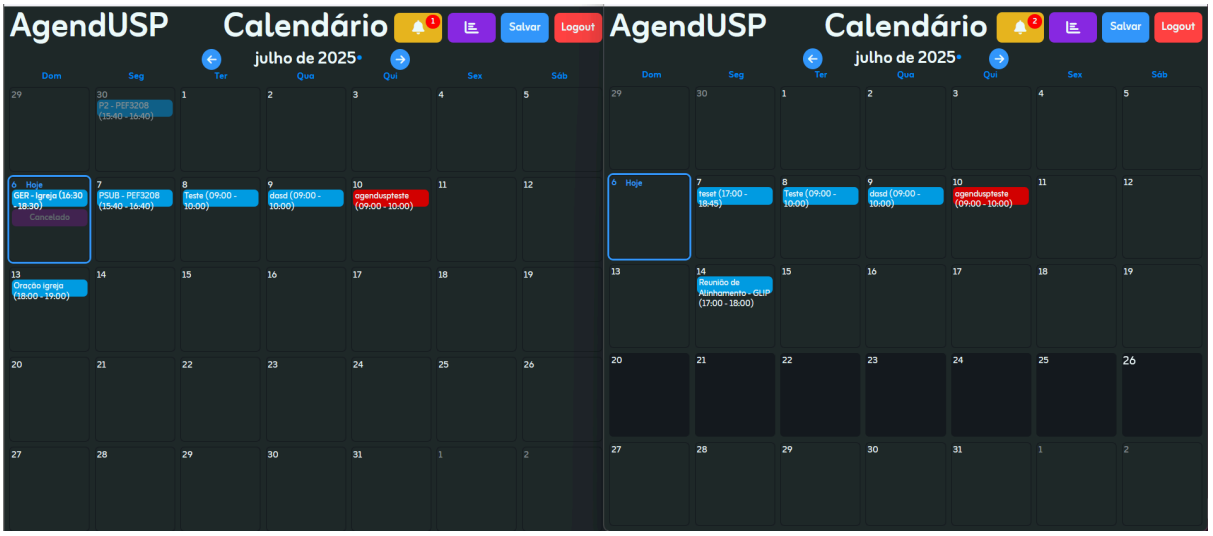


As estatísticas mostram a quantidade de eventos que o usuário possui nesse intervalo de tempo, quanto tempo elas demandam, e quantos eventos foram cancelados.

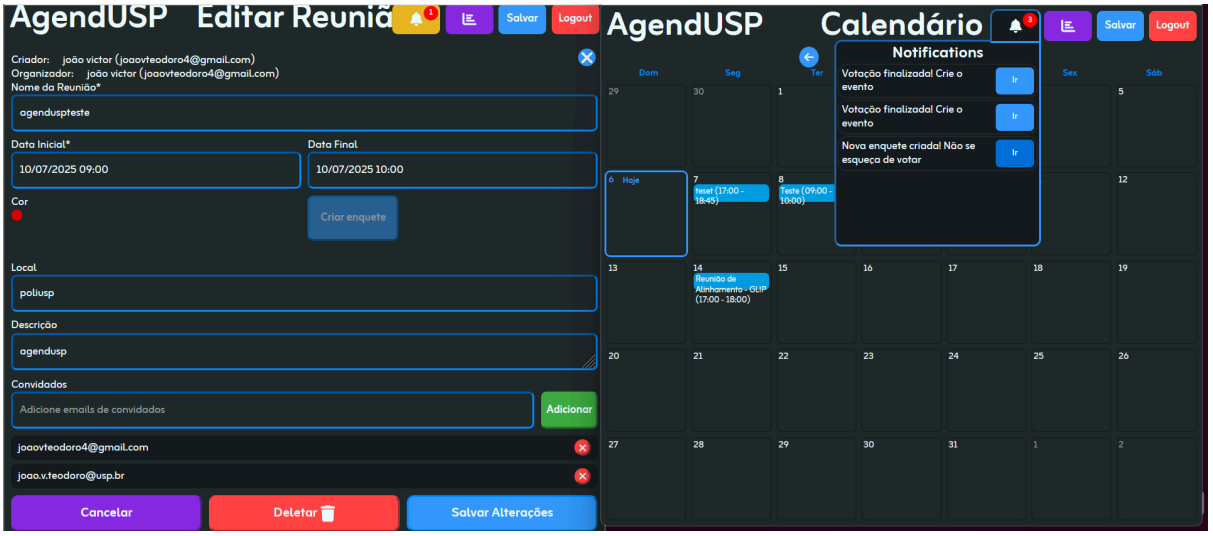




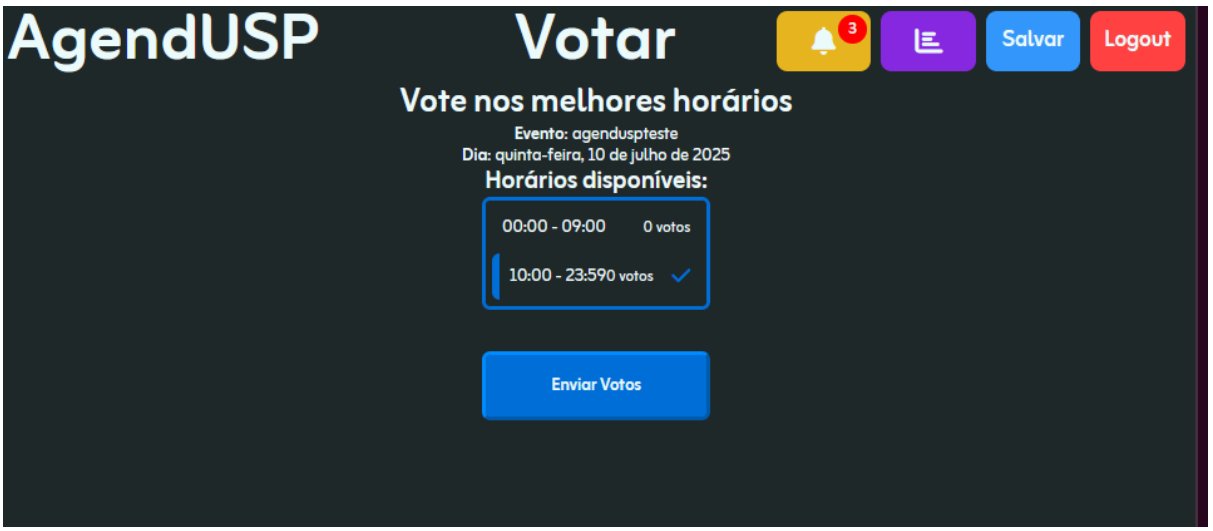
Na aba semanal, é possível gerar o relatório, dessa vez com o uso da inteligência artificial Ollama. O tempo para a geração do relatório depende do hardware do seu computador, mas pode levar até alguns minutos.



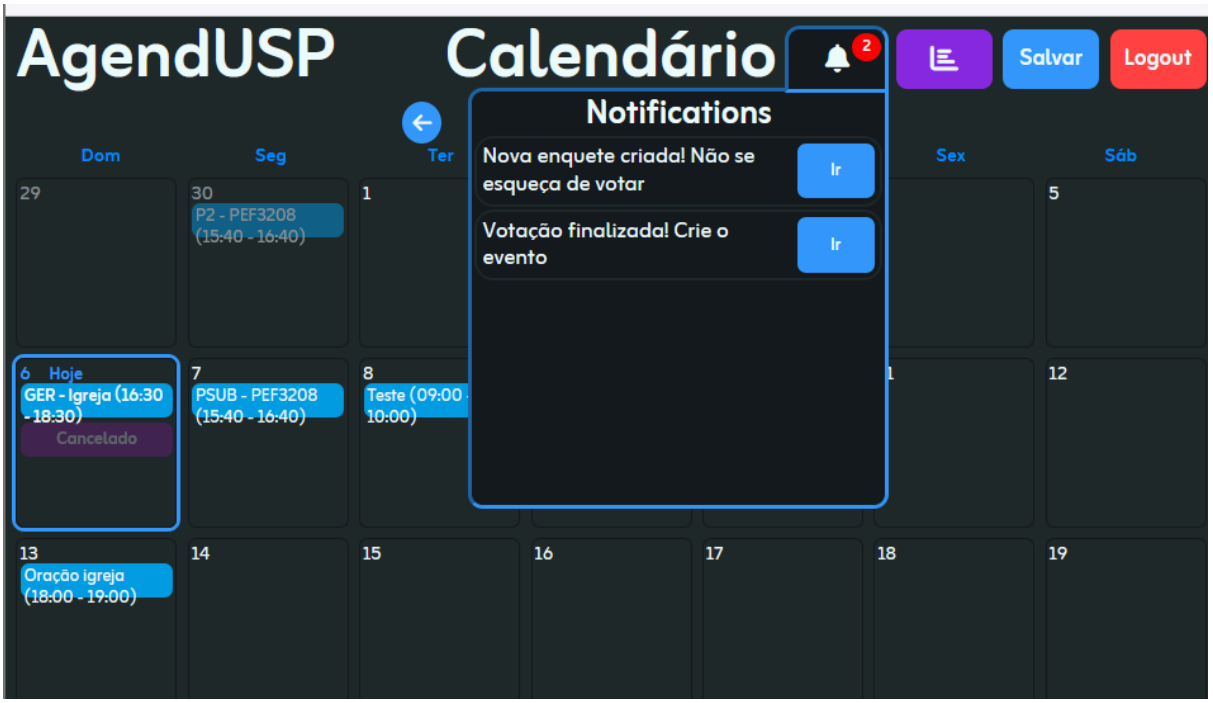
Aqui estão dois usuários diferentes rodando localmente, e o usuário da esquerda convidou o usuário da direita para o evento de cor vermelha. Dessa forma, esse evento aparece nos dois calendários.

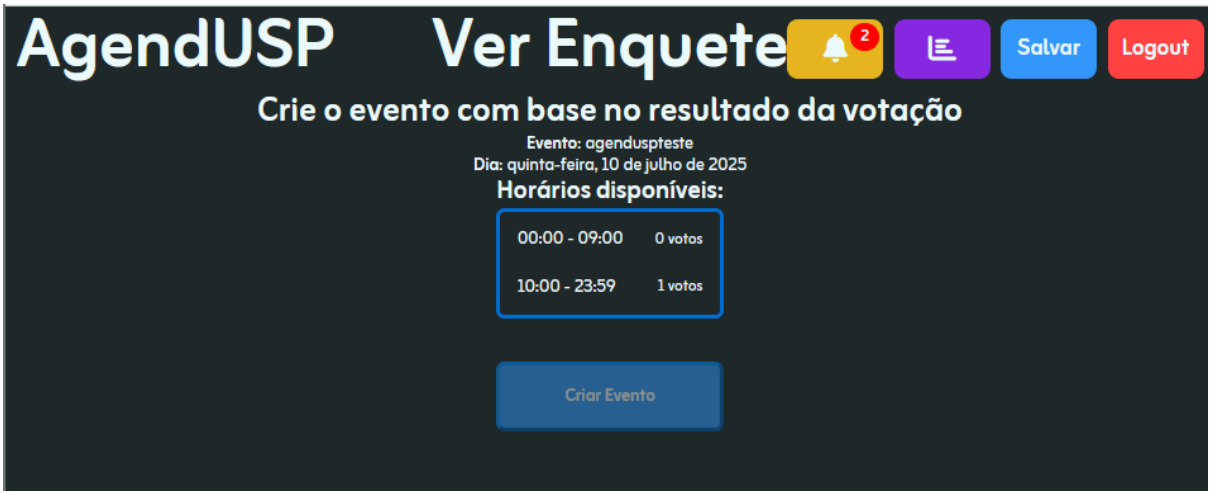


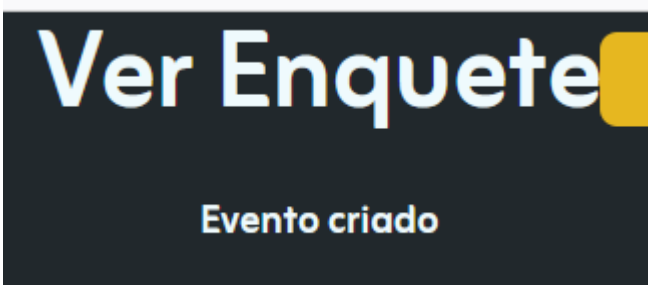
Agora, gerando uma enquete de horário para esse mesmo evento, teremos que o usuário da direita receberá uma notificação quando a enquete for criada, para lembrá-lo de votar nela. O botão da enquete do lado direito quando selecionado fica mais escuro, e do outro lado, a última notificação de cima para baixo é a mais recente.



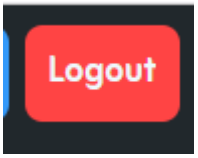
Ao clicar na notificação para ir votar, o usuário cai na aba de horários livres disponíveis do criador da enquete, e pode selecionar uma ou mais opções que se adequarem aos seus horários.







Quando todos os usuários convidados terminarem de votar, o usuário que criou a enquete recebe uma notificação de finalização da enquete, e pode visualizar os votos, e ter poder total da decisão do dia ou horário do evento.



Por fim, quando o usuário estiver satisfeito com o uso das funcionalidades do AGENDUSP, ele pode clicar no botão Logout e se desconectar sem maiores problemas.

Backend e estrutura

As implementações novas são as classes de enquetes e notificações, além das classes para IA. Na criação das enquetes, e das notificações são utilizadas as classes:

- *EventPoll*: localizado na pasta **dataobjects-> eventObjects**, implementa os métodos de encontrar o horário livre do usuário que propõe a enquete, votação, os métodos de pegar, colocar os convidados, saber quando a enquete deve ser finalizada (após o voto de todos os usuários). Além disso, existem os

métodos de pegar a id da enquete, e o id do criador da enquete e o id do evento.

- *EventPollRepository*: está na pasta **repositories**, e estende a classe do MongoDB e adiciona novos métodos
- *EventPollDataController*: esse arquivo está na pasta **controller**, implementa os métodos de pegar todas as enquetes, pegar pelo ID, criação da enquete, votação.
- *PollNotification*: está alocada na pasta **dataobjects,** e estende a classe das notificações, criando também métodos de instanciar e pegar o id da enquete.
- *EventPollNotification*: dentro da pasta **events**, essa classe implementa os métodos de pegar e editar a mensagem da notificação, além dos ids, da enquete e dos eventos
- *EventPollDoneEvent*: essa classe se encontra na pasta **events**, e é utilizada quando a enquete para o evento é finalizada.
- *EventPollDoneEventListener*: na pasta **services**, fica escutando pra ver se a classe "*EventPollDoneEvent*" avisa que a enquete foi finalizada, e gera notificação para o criador da enquete
- *InvitePollListener*: também na pasta **services**, assim que surge uma enquete, essa classe notifica os usuários convidados ao evento para que eles votem na enquete
- *EventPollDataControllerTest*: pasta **tests**, testa as implementações das enquetes

Para a implementação da IA, geradora dos relatórios semanais, foram criadas as seguintes classes:

- AIController: localizada na pasta **google**, essa classe utiliza os requests para conectar o frontend com o backend e gerar os relatórios da semana, pegando também os dias de começo e de fim do intervalo de tempo que o relatório cobre.
- AIRequest: está na **dataobjects -> aiObjects**, possui um construtor de prompt para a IA, set model, em geral, métodos de criação e construção de um prompt para o Ollama, gerador de relatórios

Divisão de tarefas e evolução das contribuições

O AGENDUSP teve que ser dividido em várias partes para que ficasse funcional, compartimentalizado e fácil de entender. Dessa forma, houve maior integração entre o front e o backend, além de testes serem feitos rodando o projeto no computador, além dos testes de arquivo. Também, devido a criação das enquetes, o sistema de notificações foi atualizado, pois houve a necessidade de adicionar o sistema de notificação automática na mudança dos eventos e enquetes, fazendo com que os usuários envolvidos recebam a mensagem em sua interface gráfica. Para o caso de uma notificação de enquete, essa some apenas se o usuário responder à enquete de sugestão de horários para um evento. Ademais, com a implementação finalmente funcional da IA, utilizando-a para gerar os relatórios, foi necessário criar várias novas classes, além de refatorar as antigas.

- André:
    - Execução que pega os dados das classes e transforma em prompt para o Ollama gerar os relatórios
    - Comentários de explicação do código
    - Separação das responsabilidades de classes que faziam a mesma coisa
    - Mudanças nas classes das enquetes
- Diego:
    - Criação dos botões pra enquete no front e lógica do aparecimento ou desaparecimento da notificações e capacidade de gerar voto e de definir horário novo do evento com base na votação
    - Alteração da capacidade de gerar enquete pelos usuários que não são o “owner” ou “organizer” do evento
    - Implementação da exclusão de eventos, e não só o cancelamento no frontend
    - Correção do relatório da IA
- Gabriel:
    - Correção da importação dos dados para o front
    - Ajuste do controlador da IA
    - Atualização da nuvem para receber os dados atualizados dos eventos do usuário
    - Integração do backend com o frontend da IA
    - Exclusão e refatoração dos códigos não utilizados
- João:

    - Implementação da notificação no backend
    - Refatoração e exclusão de classes obsoletas
    - Implementação do cancelamento de eventos
    - Implementação das enquetes no backend
    - Correção de injeção de dependências nos controladores locais
- Matheus:
    - Implementação do componente no front que mostra no calendário, nas respectivas datas, os feriados nacionais
    - Implementação no frontend dos botões que faltavam
    - Correções visuais e estéticas
    - Criação de testes para controladores locais
    - Criação da lista de notificações do usuário

- Sophia:
    - Implementação das classes de notificação
    - Conexão das notificações no back e o frontend
    - Inclusão das notificações no front e especificação delas para o usuário
    - Geração de testes para as enquetes
    - Refatoração de classes que geram o relatório dos compromissos do usuário

Correções

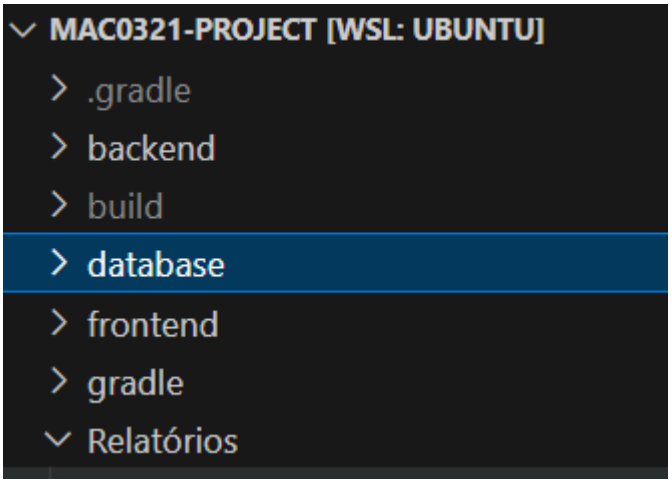
Algumas classes que implementam as enquetes estavam usando incorretamente a terminologia “pool” ao invés de “poll”, e agora estão corretas. Além disso, a implementação das estatísticas semanais, como horas de reunião e quantidades de eventos do usuário foram refatoradas para melhor resumir e exibir os dados. No frontend, alguns botões estavam levando para links https errados devido a um uso errôneo de “navigate” em javascript. Ainda no frontend, botões que antes estavam mal posicionados em algumas resoluções agora estão corretamente posicionados.

Instruções para compilar e executar o programa

Antes de rodar o programa, para ser possível convidar o usuário, é necessário que essa conta esteja registrada em nosso auth para enviar requisição para o Google, dessa forma, se

deseja testar essa funcionalidade no programa, nos envie um email diferente para que tenha acesso em duas contas ao mesmo tempo, a fim de testar essa funcionalidade.

Para conseguir executar o AGENDUSP em seu dispositivo, é necessário ter o docker, que pode ser encontrado nesse link, juntamente às suas instruções de instalação: https://docs.docker.com/engine/install/. Versões requeridas: Node (18.19.1) e npm (9.2.0). Em seguida, acesse a pasta database do agendusp:




```
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project$ cd database
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project/database$
```

Agora, ative os containers com o comando `docker compose up -d`

```
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project/database$ docker compose up -d
[+] Running 2/2
 ✔Container mongodb  Running
 ✔Container ollama   Running
```

Volte para a pasta mãe e entre na pasta frontend, e em seguida, na pasta app

```
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project/database$ cd ..
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project$ cd frontend/app
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project/frontend/app$
```

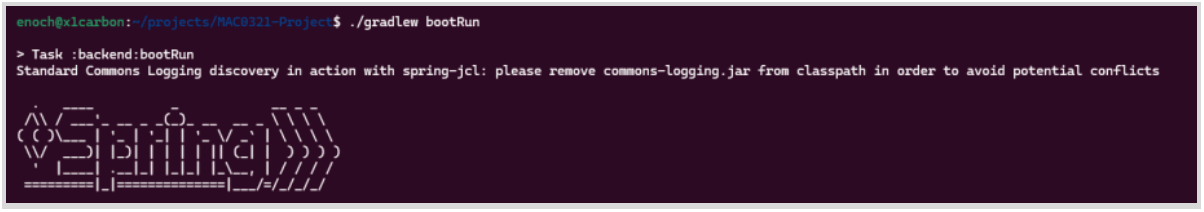
Instale o npm e o inicialize usando `npm install` e `npm start`

```
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project/frontend/app$ npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@csstools/postcss-trigonometric-functions@1.0.2',
npm WARN EBADENGINE   required: { node: '^14 || >=16' },
npm WARN EBADENGINE   current: { node: 'v12.22.9', npm: '8.5.1' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@csstools/selector-specificity@2.2.0',
```

```
sophia@Lenovo:~/windows-files/documents/mac0321/MAC0321-Project/frontend/app$ npm start

> app@0.1.0 start
> react-scripts start
```

Em seguida, volte para a pasta mãe e rode o comando `./gradlew bootRun`



Para finalizar o programa que está rodando, escreva o comando Control + C no terminal