



Relatório da Entrega 4 - 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 dos requisitos 1 a 3 do projeto

Requisito 1 - Com a interface gráfica implementada pelo frontend, o usuário consegue visualizar toda a sua semana, sendo capaz de ver os diferentes compromissos em seu calendário escolhido. Utilizando os métodos das classes de “CalendarDataController”, “EventsDataController”, agora separados em controles diferentes, o usuário é capaz de criar, atualizar e cancelar compromissos, que ficam ainda salvos, mas possuem seu “status” como cancelado. Todo o evento criado ou organizado pelo usuário possui a funcionalidade de convidar pessoas apenas utilizando sua identificação (e-mail). Com os parâmetros “creator” e “organizer”, utilizados pelo próprio GoogleCalendar, é possível ter a diferenciação do papel do usuário no evento, mudando suas permissões e sua capacidade de alteração do evento. Os compromissos do usuário são armazenados localmente no container do MongoDB.

Requisito 2 - Na própria criação do evento, no próprio frontend, é possível adicionar enquete de horário e dia para a ocorrência do evento, consultando os seus convidados. A enquete aceita respostas que fazem parte dos horários livres do criador da enquete. Independentemente do resultado da enquete, o criador/organizador tem poder total da decisão sobre o dia e horário do acontecimento do evento. A lógica do backend da enquete está implementada, mas ainda são necessários ajustes no frontend e nas classes de dados, de forma a facilitar a visualização pelo front.

Requisito 3 - O AGENDUSP utiliza a classe “HomeController” que pega um intervalo escolhido pelo usuário e mostra para o usuário as estatísticas dele, que são mostradas no front. Essas estatísticas são a quantidade de eventos que o usuário participou durante o intervalo, a quantidade de tempo que esses eventos ocuparam e quantos eventos foram cancelados neste intervalo.

Interface gráfica e frontend

O frontend e a interface gráfica continuam utilizando o reactjs. São diversas as novas funcionalidades e arquivos implementados. No [App.js](#), localizado no src, foi implementado uma rota privada para autenticação, que faz uma requisição para o backend, verificando se o usuário está autenticado, e se não, o envia para a página de login. Após o login, o usuário é jogado para a última página que tentou acessar.

O componente geral é o header, que está em todas as pastas, e só tem seu título alterado, e possui botão para voltar para o início, botão de logout, botão de salvar e um botão para abrir as notificações. Também tem-se o menu pro calendário que pode mostrar o mês inteiro ou a semana, permitindo a visualização de todos os eventos. Se pode clicar nos eventos para editá-los ou criar um novo evento. Além disso, agora é possível criar uma enquete e também votar em alguma enquete do evento que o usuário seja um convidado. Agora, segue um panorama do funcionamento e separação de novas classes para o frontend:

Na subpasta de componentes, foram criadas as seguintes pastas, que contém, em geral, um arquivo .css, um arquivo .js, um arquivo de testes e um arquivo index, que exporta o componente.

- CalendarMenu - guarda os arquivos que pegam as informações dos dias e dos meses de forma a completar a grade na tela do usuário
- CalendarPollMenu - pega os intervalos livres do criador da enquete, cria a enquete e envia a enquete para o backend
- DataVisualization - pasta que guarda toda a parte de visualização semanal e dos relatórios dados pela IA
- EventBlock - pasta que guarda a edição visual das cores e componentes do evento
- EventMenu - cria o menu de criação de um evento e de alteração de um evento
- LoginMenu e LoginSucess - cria o menu para o login do usuário, e se tiver sucesso, volta para a página que o usuário estava tentando acessar anteriormente
- NotificationList - cria a lista de notificações
- PrivateRoute - cria as rotas privadas para login e autenticação
- PageHeader - configura o formato da página
- Vote Menu - pega os horários disponíveis do usuário criador através do backend e gera o menu de votação
- WeekView - cria o display do calendário semanal bem como o dos eventos

Padrão de projeto escolhido

O padrão escolhido foi o Observer, um padrão comportamental que descreve um mecanismo de assinatura que serve para notificar diversos objetos sobre algum evento ou mudança de estado que aconteça no objeto observado por eles. No AGENDUSP, o objeto observado é uma enquete de um evento, enquanto os observadores são os convidados desse evento.

O arquivo FormController usa o MessageMapping para enviar a mensagem com a URL para todos os inscritos (que o frontend coloca automaticamente, se você for convidado para o evento). O MessageMapping também possui o “destination” que envia para todos os que estão olhando para um evento específico. Já no outro arquivo, o WebSocketConfig, o “broker” configura os endereços para envio e leitura das mensagens. O destination/notify é o local em que ficam todas as notificações, e o “prefixes” é o local de envio das mensagens. O “endpoints” é o local de inscrição nos eventos pelo websocket, que, conforme dito anteriormente, possui inscrição automática do convidado pelo front.

Esse padrão funciona de forma a adicionar os usuários que participam de uma enquete em uma lista de observers, que, utilizando o websocket (que funciona como uma mensagem), alerta os observers das mudanças de uma enquete. A vantagem de utilizar esse método é que elimina a necessidade do usuário de ficar toda hora consultando a enquete, pois, quando ela for alterada, automaticamente receberá uma notificação sobre essa mudança.

Divisão de tarefas e evolução das contribuições

Há duas semanas, a organização das próximas etapas do projeto tinha maior divisão, no entanto, conforme o avanço do trabalho, percebemos que as partes eram muito mais dependentes que o previsto. Dessa forma, o que cada um realmente fez foi:

- André:
 - criação do PromptBuilder (que cria o prompt que será enviado para o Ollama), que acessa os dados de usuário de forma a fazer a IA pegar os dados do usuário
 - criação do relatório das tarefas dia ou da semana da semana do usuário por meio de IA
- Diego
 - implementação de conexões do frontend
 - finalização do login e autenticação com o backend
 - simplificação do front e alteração do funcionamento do calendário
 - criação do bloco de evento e preparação do front para receber o backend
 - criação da lista de notificações
 - implementação dos “websockets”
 - atualização dos eventos
 - implementação do padrão de projeto Observer
- Gabriel:
 - criação do AI Controller (que controla a funcionalidade de enviar um prompt para a IA)
 - criação do AI Request e AI Response
 - criação do relatório das tarefas dia ou da semana da semana do usuário por meio de IA
- João:
 - implementação do container de IA
 - conserta permissões do mongo
 - cria as funções de envio de enquetes
 - corrigir erros de autenticação no controlador do CalendarList
 - sincronização da base local com o google
 - refatoração do projeto para retirar o uso do EventList
 - implementação do padrão de projeto Observer

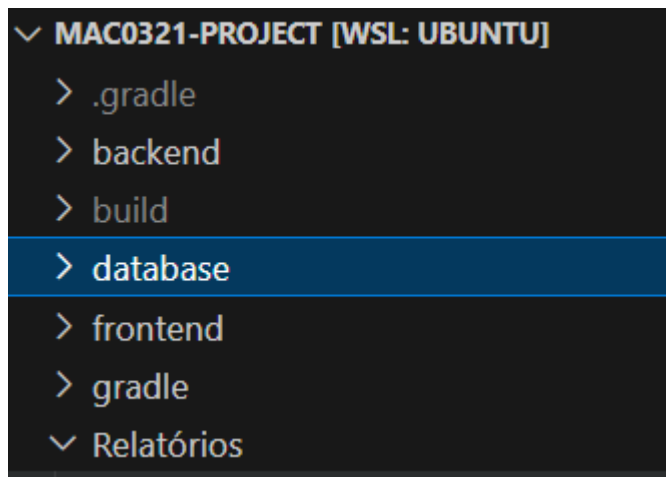
- Matheus:
 - criação a enquete do horário e dia do evento
 - criação de testes para o backend da criação da enquete
 - implementação do envio das enquetes para os usuários inscritos
 - envio dos dados da enquete para o prompt da IA
- Sophia:
 - implementação de testes para os DataController específicos (Events e Calendar)
 - refatoração do PromptBuilder para a IA utilizando o @RestController
 - implementação override nos métodos ToString para utilizar no PromptBuilder
 - criação de testes para as classes de IA, PromptBuilder
 - resolução da autenticação
 - criação do relatório das tarefas dia ou da semana da semana do usuário por meio de IA

Correções

De acordo com a implementação enviada no relatório passada, foram seguidas as recomendações e assim foram retirados os comentários de algumas classes, apagadas as classes duplicadas, e excluídas as classes inúteis. Nas classes em que foi possível reduzir a quantidade de métodos, isso foi feito, como por exemplo o DataController, que foi separado em três classes menores, apenas com métodos específicos. Algumas classes e métodos ainda sem uso não foram deletados para que não fossem esquecidos de implementar, e, além disso, durante a criação dos códigos, quando lembramos de algo, implementamos nessas classes e funções, evitando de ter que escrever em uma parte separada, ou ficar criando e excluindo esses arquivos. Inserimos as instruções para compilação e execução logo abaixo.

Instruções para compilar e executar o programa

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',
npm WARN EBADENGINE   required: { node: '^14 || >=16' },
npm WARN EBADENGINE   current: { node: 'v12.22.9', npm: '8.5.1' }
npm WARN EBADENGINE }
```

```
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`

```
enoch@x1carbon:~/projects/MAC0321-Project$ ./gradlew bootRun
> Task :backend:bootRun
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid potential conflicts
```



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

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

Próximos passos

Alguns dos próximos passos a serem feitos no projeto AGENDUSP são:

- ajustar a implementação da IA no back e no frontend
- criação dos testes da implementação da IA
- enviar os compromissos para o GoogleCalendar, a fim de eles não ficarem salvos apenas localmente
- adicionar mais dados às estatísticas do intervalo de tempo

Bibliografia e referências

<https://refactoring.guru/design-patterns/observer>. Acesso em 22/06/2025.

<https://docs.docker.com/engine/install/>. Acesso em 08/06/25.