Relatório da Entrega 3 - 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

Pastas do Repositório

Relatórios - pasta para adicionar os pdfs dos relatórios.

backend - contém o servidor, o banco de dados e as API's, toda a parte de controle e testes, o login, o processamento, entre outros.

database - contém o Docker para a base de dados local (Mongo) e o configura no arquivo `docker-compose.yaml.`

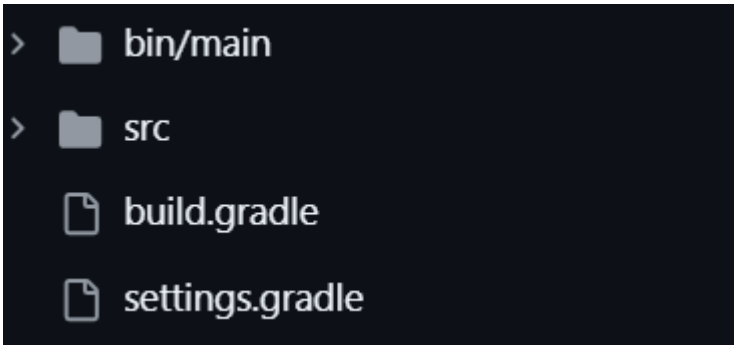
frontend - contém o react-app que é utilizado como interface: arquivos .js com a lógica básica de componentes individuais, arquivos .css para definição de estilos desses componentes, e *assets* que são utilizados para definir uma identidade visual.

gradle/wrapper - é a parte de conexão com a API do Google, que contém a aplicação do gradle e a definição de suas propriedades.

No entanto, neste relatório vamos focar nas pastas backend e frontend, que são as pastas de maior desenvolvimento atual.

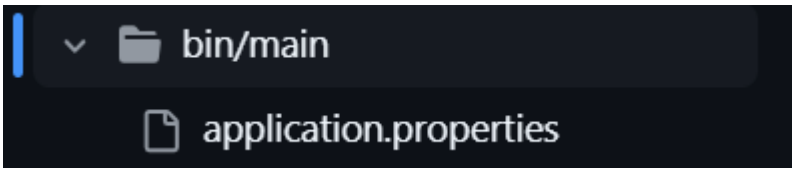
Pasta Backend

Organização da pasta:



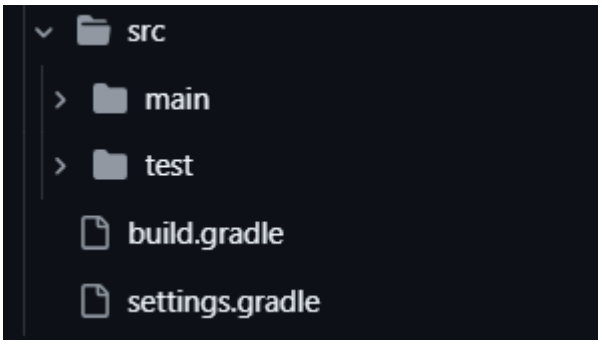
Subpasta bin/main

Essa pasta muda a estrutura do projeto gradle para sustentar uma implementação de múltiplos projetos. O arquivo `application.properties` contém as informações da aplicação como o nome do projeto, porta do servidor e as informações para login com o servidor do google.



Subpasta src

Organização das pastas:



Dentro dessa subpasta, encontra-se dois arquivos em formato gradle e ainda mais duas subpastas. O arquivo `build.gradle` contém os plugins e as dependências usadas no

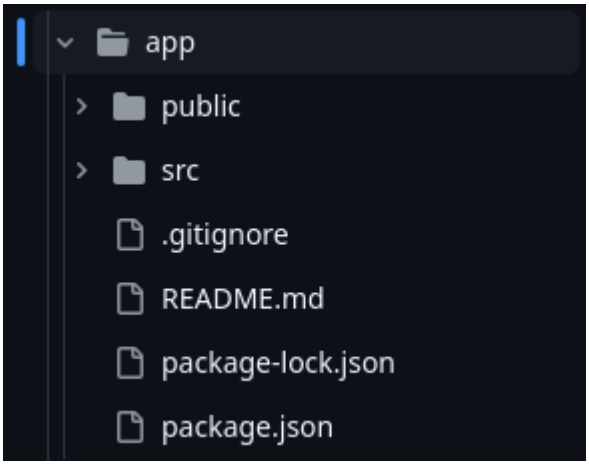
AGENDUSP. Já o arquivo `settings.gradle` serve somente para retornar o nome do projeto raiz como “backend”. As pastas ainda mais internas a essa são:

- Main:
    - java/br/com: o arquivo `GoogleAuthorization.java` contém as dependências de autorização do login do google.
        - agendusp/agendusp:
            - calendar: contém as classes que definem os parâmetros do calendário local e do Google. Possui controladores abstratos e também específicos para os calendários e eventos locais. Seu arquivo que “chama” os métodos a serem usados pelo calendário é o `DataController.java.`
            - config: guarda as diferentes configurações para o programa a depender de seu uso desejado.
            - controller: possui os controladores, que por sua vez definem todos os métodos a serem usados pelo usuário, recebendo as requisições via protocolo http e devolvendo as respostas adequadas.
            - dataobjects: contém os objetos de dados específicos que possuem vários parâmetros diferentes da classe em que seria colocado, e também adiciona os seus getters e setters.
            - documents: guarda os resources que copiam a implementação do Google. Contém todos os parâmetros que o Google Calendar utiliza. Também adiciona getters e setters e provém a informação da relação dos usuários com o calendário.
            - repositories: cria os repositórios de usuário, “Calendar” e “Events” estendendo a aplicação da base de dados local.
            - services: personaliza a `DefaultOAuth2UserService` para a nossa aplicação de projeto.
    - resources: essa pasta contém os arquivos do Spring Boot para alterar o nome e definir parâmetros sobre o Google, e um database para criar uma categoria de alteração da segurança para testes.

- Test: essa pasta contém todos os arquivos de teste do projeto backend. Até agora foram feitos os testes para os repositórios de usuário e calendário, o teste de container do Mongo, a base de dados local escolhida para ser utilizada e os testes para o “Data Controller”. Os testes para o “Data Controller” estão separados em três arquivos: um para o “Calendar”, outro para os “Events” e ainda mais um que testa todas as operações, mas usando o formato MockTest.

Pasta Frontend

Subpasta app



Essa pasta contém mais um arquivo `.gitignore`, para evitar o envio de arquivos desnecessários (no caso do Reactjs, os principais seriam aqueles na pasta “node_modules”) os arquivos `package.json` e `package-lock.json`, que servem para definir quais pacotes npm são usados no subprojeto do front, um `README.md` genérico, gerado na criação da do Reactjs app, e as pastas "public” e "src”:

- public: arquivos de imagem e outros *assets* que são globais em relação à aplicação.
- src: arquivos de configuração e de estilização da aplicação, que se aplicam a quase todas as páginas.
    - assets: arquivos de imagem que são utilizados por páginas individuais.
    - components: reúne todos os componentes criados até o momento, seguindo o paradigma modular que caracteriza o Reactjs. Cada componente contém um arquivo descritivo `README.md` padrão, um arquivo .css para definir o estilo do componente, e 3 arquivos .js que, em conjunto, definem a estrutura e o comportamento do componente, e o exportam para serem usados pela aplicação.

Arquitetura escolhida

A arquitetura escolhida para o projeto é o framework Spring Boot com a build tool Gradle, por várias razões:

- **Gerenciamento eficiente de dependências**: o Gradle permite fixar ou atualizar versões de dependências com facilidade, evitando conflitos entre versões.
- **Arquitetura organizada**: o Spring Boot facilita a manutenção, testes e também reuso de código através do padrão MVC, tornando fácil modificar partes específicas de código sem afetar outras (código modular), componentes que podem ser reutilizados em diferentes partes do código (reusabilidade) e também cada componente pode ser testado independentemente, fazendo o processo de testes mais rápido e eficiente (fácil manutenção).
- **Eficiência do Gradle**: o Gradle utiliza build incremental, o que significa que ele apenas compila o que foi alterado desde a última compilação, o que economiza bastante tempo em aplicações grandes e com muitas dependências.
- **Suporte para testes**: Possui suporte a testes de integração e de aplicação através do MockMVC, que torna possível simular requisições http aos controllers, imitando internamente um ambiente web, o que é ótimo para validação de erros e testes de autenticação.

Divisão de Tarefas

Para esse projeto a metodologia ágil está sendo utilizada, focando principalmente na prática de sprints, que são pequenas entregas rápidas (ou incrementos) de features específicas do projeto, com um ciclo de desenvolvimento de uma semana. Para cada sprint, as tarefas são divididas entre os membros do grupo. As próximas coisas a serem feitas serão:

- Resolver a autenticação e resolver os testes de DataController - Sophia.
- Escrever novos testes para testar o backend - João, Diego, Matheus e Sophia.
- Fazer um container de Inteligência Artificial dentro do docker - João.
- Criar a enquete do horário e dia do evento, que aparecerá para usuários convidados - Diego (frontend) e Matheus (backend).
- Página do relatório que a IA falará sobre os eventos da semana do usuário - André (frontend) e Gabriel (backend).

Dificuldades encontradas

Algumas das dificuldades estão relacionadas à curva de aprendizado do Spring Boot, que contém injeção de dependência e as classes singletons “beans”, que são conceitos difíceis de entender, e com muitas aplicações. A auto-configuração do Spring Boot algumas vezes esconde detalhes importantes que dificultam a depuração do código. Ainda, a sintaxe do Gradle não é tão intuitiva, principalmente ao usar pela primeira vez.

Além disso, a comunicação e autenticação no servidor do Google foi um dos grandes desafios do projeto devido a diversos motivos, como o fato do próprio tutorial do google sobre a API calendar estar desatualizado (deprecated), e a autenticação para a conexão com o Google ser complexa por causa do token de segurança, pois a Rest API está protegida pelo “SecurityConfig”, e ainda, é complicado simular um usuário que passe na autenticação do OAuth2.

Progresso desde o último relatório

Após o último relatório, foi criado um Docker, e esse docker define o serviço MongoDB, que será executado em um container passando a imagem, o critério de reinício, e definições de usuário. Dessa forma, foi possível conectar o Spring Boot ao *container* do Mongo, e esse foi o principal avanço relacionado ao backend. Essa parte foi executada pelo João.

Em seguida, todos os commits feitos nos branches `feature/129-datacontroler` e `feature/107-googlecalendarlistcontroller` foram passados para o branch main. Foram feitas algumas mudanças nas pastas, mas nada muito diferente. Commit enviado pelo João, mas a tarefa foi feita em conjunto com o grupo.

Início do desenvolvimento das partes que utilizam IA, como a pesquisa e criação dos modelos a serem vistos, além do estudo das bibliotecas e classes que deverão ser utilizadas, com interação com a API do Google Calendar. Essa parte foi feita pelo Gabriel.

Foi feito um planejamento e organização sobre os próximos Sprints e tarefas do grupo, incluindo prazos estimados para finalização de tarefas e etapas. Esse passo foi feito pelo Matheus, com uso da plataforma OpenProject.

Foi criada uma identidade visual para o grupo, com logo e desenho, e também criado um email “AGENDUSP”, para servir de usuário teste. Além disso, foi criada uma página que muda o Token de segurança para uma configuração “test”, para ser usada nos testes, de forma a não dar problemas na autenticação, que não funcionou totalmente. Essa etapa foi feita pela Sophia.

Para a aplicação do gradle, foi necessário separar os application-properties de cada repositório, definindo o nome e as permissões passadas para cada caso específico, como a criação de Tasks. Além disso, estudo sobre as próximas etapas da aplicação da IA. Esses passos foram feitos pelo André.

Já no frontend, já havia sido criada uma aplicação com Reactjs, sem nada além de um componente de exemplo. Entretanto, desde a entrega do último relatório, foram desenvolvidos outros aspectos, como a base da identidade visual, e alguns componentes que definem diferentes páginas, como o calendário em si, o *login*, e um menu de edição de um evento. Essa implementação foi feita pelo Diego.