Olá Professora, nós decidimos fazer esse texto pra avisar de alguns problemas conhecidos do nosso código que talvez seja relevante citar para evitar de algumas confusões, além disso adicionamos também alguns campos de Post para o teste de API, para lhe evitar o trabalho de ter que descobrir como preenche-los:

1 – Usuário está sem endereço: Esse foi um erro meio bobo, nós adicionamos a relação de endereço para usuário mas esquecemos de enviar para a versão final do código.

2 – Número de vagas em TipoDeConsulta: A ideia era que ao criar um cronograma fosse possível limitar o número de vagas no momento do cadastramento do mesmo, mas atualmente o número de vagas é somente limitado ao alterar o número que está no Enum do TipoDeConsulta, e em razão do tempo nós acabamos não retirando o número de vagas da inserção do cronograma, como você verá no teste de API mais abaixo. Talvez você repare que ao fazer um agendamento o console imprime o número de vagas disponivel, essa impressão só estava lá por motivos de teste e esquecemos de remove-la antes de enviar o código.

3 – Direitos de admin: A parte de usuário ter funções limitadas apenas para ele enquanto o admin tem todos os direitos foi parcialmente implementado, nós planejávamos usar o @PreAuthorize para reconhecer a Enum role do usuário que estaria acessando o método. Mas no fim meio que ficamos indecisos de como isso realmente funcionaria e não conseguimos terminar a lógica geral. Nós também pensamos em delimitar as funções de admin e usuário pelo front, onde baseado no Enum do usuário a autenticação liberaria um botão novo na tela para customizar as funcionalidades de UBS, mas também neste caso ficou meio incompleto

4 – AuthUsuarioController desatualizado: devido a um problema de comunicação acabamos colocando a versão errada de AuthUsuarioControler e por isso a autenticação não está funcionando nesta versão enviada do código, mas a solução para o erro é o tanto quanto simples, basta remover as anotações do Swagger e trocar a chamada do Facade pela chamada direta do AuthUsuarioService, fazendo com que o código final se pareça com isso:

```
@RestController
@CrossOrigin(origins = "*", maxAge = 3600)
@RequestMapping("/auth")

public class AuthUsuarioController {

      @Autowired
      private AuthUsuarioService authUsuarioService;

      @PostMapping("/login")
    public ResponseEntity<Object> autenticarUsuario(@RequestBody AuthRequest
Data) {
        String nome = Data.getNome();
        String senha = Data.getSenha();

        Usuario usuarioAutenticado =
authUsuarioService.autenticarUsuario(nome, senha);

        if (usuarioAutenticado != null) {
             return ResponseEntity.status(HttpStatus.OK).body("Autenticado
com sucesso.");
        } else {
            return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Autenticação falhou.
Nome de usuário ou senha inválidos.");
        }
    }
}
```

Front-end: O Comando para executar o front-end é: npm run dev

Seguindo 3º tópico, a única forma de acessar as telas de cadastro de cronograma e UBS é digitando o URL diretamente, sendo eles: /criar-ubs e /cronograma

Testes de API:

Usuario:

http://localhost:8080/usuario

```
{

    "nome": "Aristoteles",
    "dataDeNascimento": "2004-07-23",
    "cpf": "747118",
    "sus": "4878833",
    "telefone": "8788480",
    "email": "email2@gmail.com",
    "senha": "321",
    "roles": ["USUARIO"]
}
```

UBS:

http://localhost:8080/agenda-facil

```
{

    "imagemUBS": "alguma/foto/imagem.jpg",
    "nomeUBS": "UBS1",
    "endereco": {
        "rua": "Mirabolante",
        "bairro": "Beringela",
        "cidade": "Acapulco",
        "estado": "Ceara"
    }
}
```

Cronograma:

http://localhost:8080/cronograma

```
{

    "ubs": {
        "id": "2383f7c1-1d2a-48ea-a646-47233014f056"
    },
    "diasSemana": ["MONDAY", "MONDAY", "TUESDAY", "WEDNESDAY", "THUSDAY",
"FRIDAY"],
    "tiposConsulta": ["MEDICO", "PEDIATRA", "MEDICO", "PEDIATRA", "DENTISTA",
"MEDICO"],
    "vagas": [10, 10, 10, 10, 10, 10]
}
```

O ID precisa ser atualizado para o novo ID de UBS que foi gerado para você.

Para adicionar mais do que um tipo de consulta para cada dia basta inserir o dia desejado outra vez, como demonstrado com Medico e Pediatra em MONDAY. Os dias da semana precisam ser em inglês pois é essencial para a funcionalidade da biblioteca do DateTime.

Agendamento:

http://localhost:8080/agendamento

```
{

    "usuario": {
        "id": "d4bcd829-390c-4994-8acb-a0b65e5f18c6"
    },
    "ubs": {
        "id": "2383f7c1-1d2a-48ea-a646-47233014f056"
    },
    "dataConsulta": "2024-03-05",
    "tipoConsulta": "MEDICO"
}
```

O ID de usuário e UBS precisam ser atualizados para o ID gerado para você.

O usuário representa o Paciente que marcou aquela consulta e a UBS representa qual UBS ela será realizada.

A inserção de data está no padrão americano ano-mês-dia

Se seguido as instruções do 4º tópico, o teste de autenticação deve funcionar corretamente, e se for o caso, o teste para ele é este:

AuthUsuario:

http://localhost:8080/auth/login

```
{

    "nome": "aristoteles",
    "senha": "321"
}
```

No mais, nós pedimos desculpa por todos estes erros que deixamos passar, além das logicas de negócio para garantir que a inserção de dados estivesse perfeita, mas apesar de todo o tempo disponibilizado para nós, infelizmente ainda não foi o suficiente para nosso grupo conseguir finalizar os nossos objetivos, espero que este documento lhe ajude a nos avaliar melhor de alguma forma.