



Gustavo Mourato, Leonardo Gutzeit,
Paulo Rosado, Thomaz Lima & Vinícius de Andrade



Veo



O QUE É A FORGEFIT?

- Uma plataforma integrada que **unifica a gestão da academia & a jornada do aluno**, com uma abordagem focada em três pilares estratégicos:
 - **1. Gestão da Rotina Fitness:**
 - Organização inteligente de Aulas, espaços & Professores;
 - Criação & acompanhamento de Treinos personalizados.
 - **2. Acompanhamento da Evolução Individual:**
 - Monitoramento de progresso físico através de Bioimpedância & medidas;
 - Ciclo de feedback contínuo com Avaliação de Professores.
 - **3. Estímulo ao Engajamento e Comunidade:**
 - Mecanismos de gamificação com Rankings e Pontuações;
 - Criação de Guildas para fortalecer o senso de comunidade & a motivação em grupo.



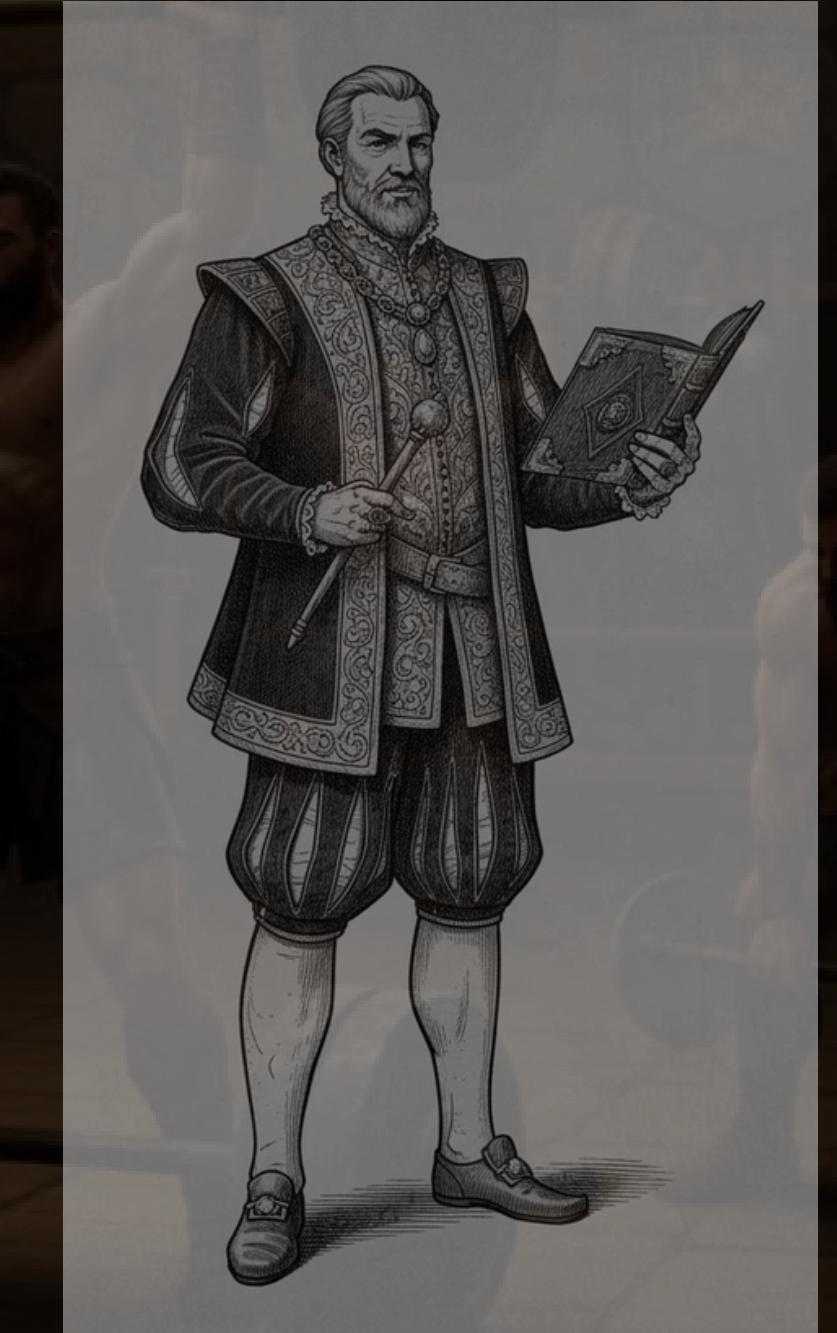
PERSONAS



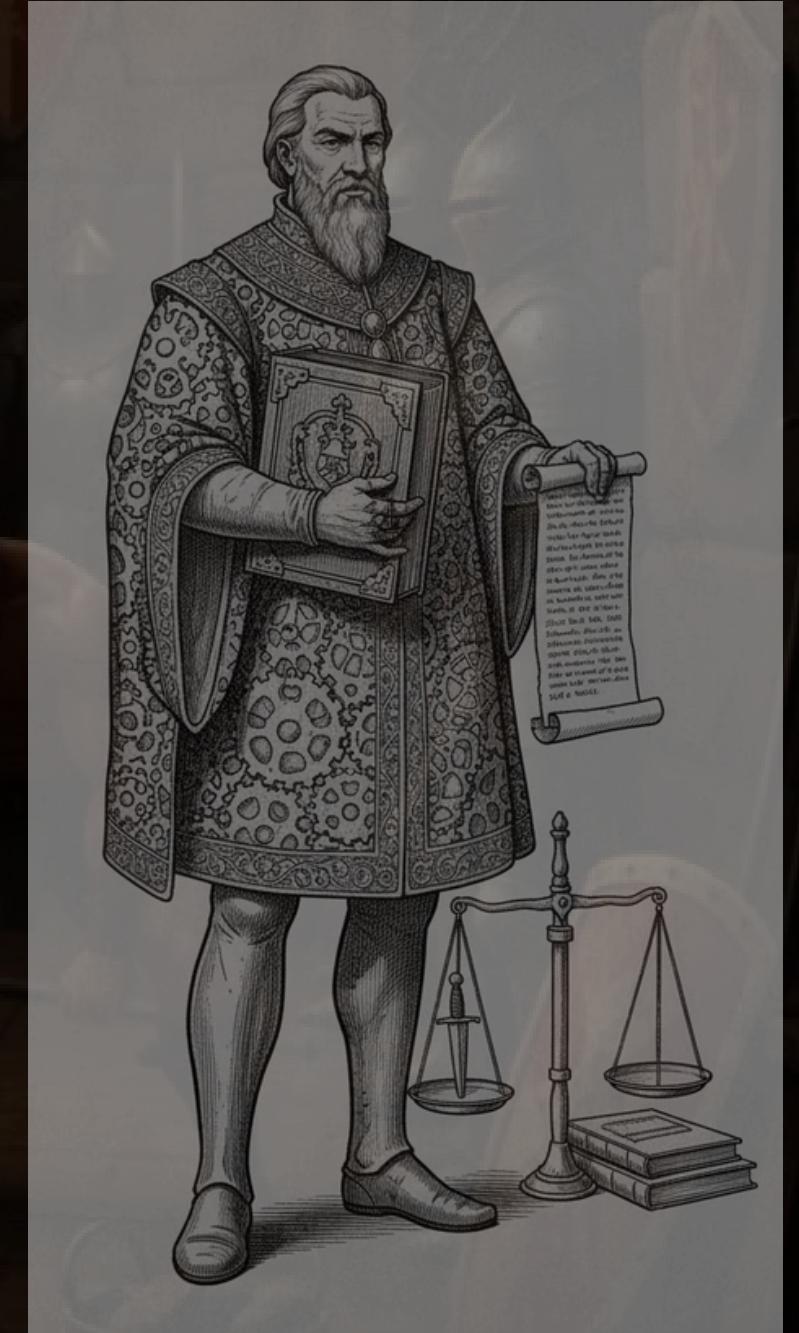
ALUNO



PROFESSOR



GERENTE



SISTEMA

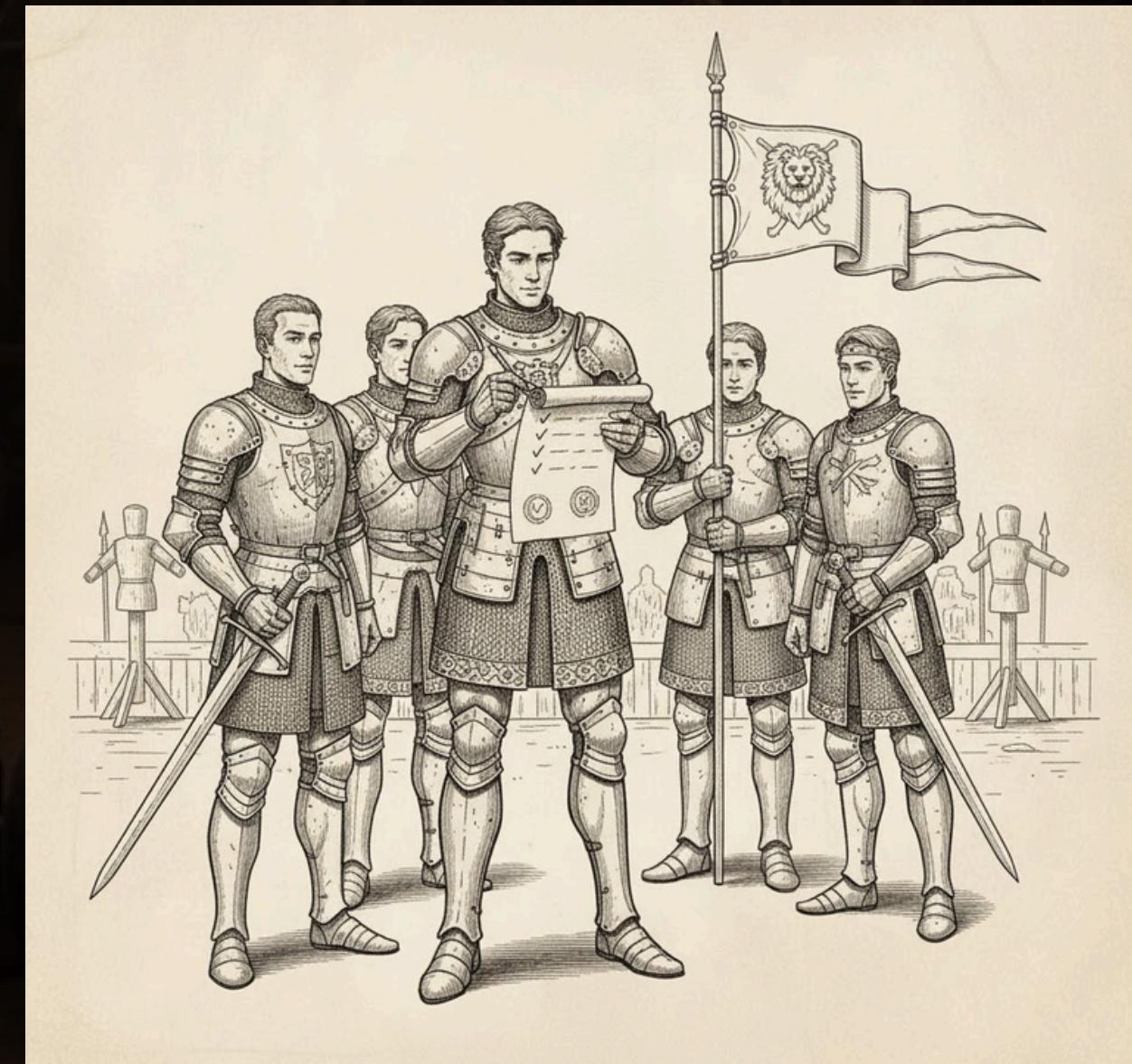


PERSONAS - ALUNO

- **Jornada:** Participar de competições com amigos
- **Objetivo:** Engajar-se com outros Alunos por meio de desafios em grupo.

- **Passos:**

- Criar uma Guilda;
- Alterar dados de uma Guilda;
- Excluir uma Guilda;
- Analisar o Rank de Pontuação;
- Entrar em uma Guilda com Código de Convite;
- Realizar Check-in diário nos Treinos;
- Acompanhar o Ranking da academia.



PERSONAS - ALUNO

- Jornada: Participar de competições com amigos

The wireframe illustrates the student's journey through the application:

- TELA DE GUILDA**: Shows a sidebar with 'Aula', 'Treino', 'Guilda' (highlighted), 'Torneio', 'Ranking', and 'Evolução'. The main area says "Participe de competições com seus amigos" with buttons for "Entrar em uma guilda" and "Criar guilda". It contains placeholder text aboutLorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur at dolor lorem. Nullam non ex sagittis.
- Entrar / Criar uma guilda**: Shows the same sidebar. The main area says "Entrar em uma guilda" with a "Inserir o código:" input field, an "ENTRAR" button, and a "CANCELAR" button.
- Página inicial da guilda com histórico de check-ins**: Shows the guilda's name "#COD3" and a history of check-ins:
 - Gustavo Mourato concluiu o treino de Peito "VAMOOOO, O DE HOJE TA PAGO" 2 dias atrás
 - Leonardo participou da aula de pilates "GALERIA MUITO FODA" 3 horas atrásA large red 'X' is overlaid on the bottom right of the history section.
- RANKING DOS ALUNOS**: Shows the sidebar. The main area displays a ranking table:

Rank	Aluno	Pontos
#1	Erico Chen	9600 pts
#2	João Marcelo	670pts
#3	Matheus Domingos	500pts
#4	José Braz	- 120pts
#5	Júlia Sales	- 100pts
#6	Demétrius	- 50pts

A large red 'X' is overlaid on the bottom right of the ranking table.
- Crie a sua guilda**: Shows the sidebar. The main area says "Crie a sua guilda" with a "Nome da guilda:" input field, a "Detalhes:" input field, a "CRIAR" button, and a "CANCELAR" button.

- Histórias:

- Listar a Pontuação individual dos Alunos da academia;
- Criar uma Guilda;
- Visualizar Código de Convite;
- Entrar em uma Guilda com o Código de Convite.

PERSONAS - ALUNO

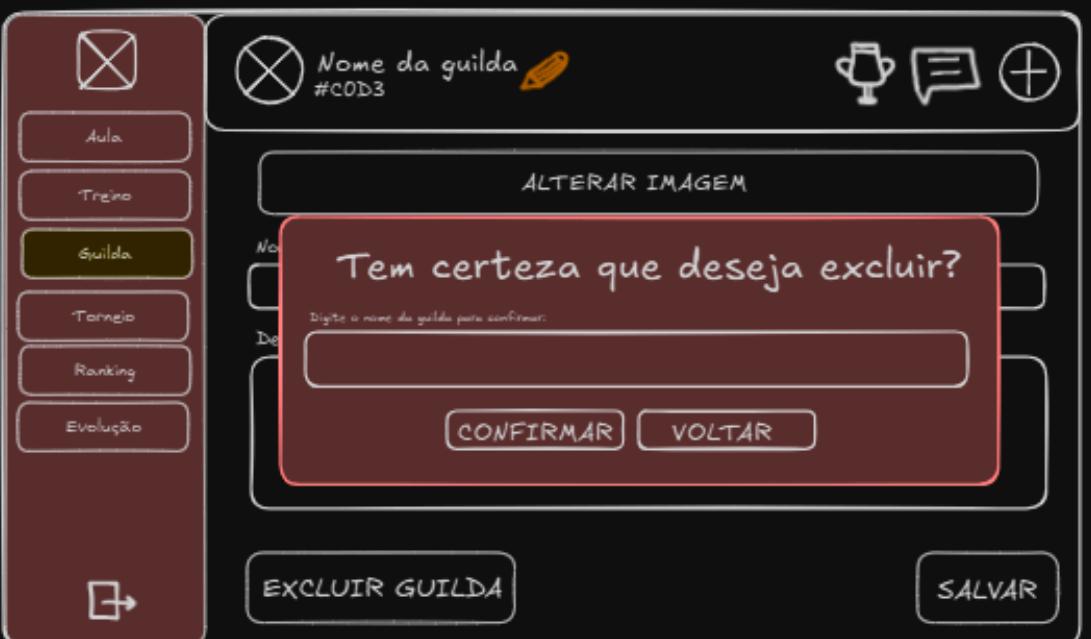
- **Jornada:** Participar de competições com amigos

The image displays three mobile application screens for managing a guild:

- Ranking de membros:** Shows a ranking of guild members. At the top, there are filters for "Este semana", "Este mês", and "Sempre". The ranking table includes columns for rank, member name, and points. Top members shown are Gustavo Mourato (1200pts), Paulo Rosado (350pts), and Vinícius de Andrade (200pts). Buttons for "ALTERAR IMAGEM", "Nome da guilda", and "Descrição" are at the bottom.
- Realizar check-in de treino:** A form to log in to a training session. It asks to "Selecionar o treino" and "Data" (today, 30/09/2025). There are fields for "Escreva uma mensagem" and "ANEXAR IMAGEM". A "ENVIAR" button is at the bottom.
- Editar informações da guilda:** A screen to edit guild details. It has fields for "Nome da guilda" and "Descrição", along with "ALTERAR IMAGEM" and "EXCLUIR GUILDA" buttons. A "SALVAR" button is located on the right.

- **Histórias:**

- Alterar dados de uma Guilda;
- Excluir uma Guilda criada;
- Listar Pontuação de membros da Guilda por semana, mês e sempre;
- Realizar Check-in de Treino.





PERSONAS - ALUNO

- **Jornada:** Realizar atividades físicas
- **Objetivo:** Acompanhar suas Aulas, desempenho & evolução corporal.
- **Passos:**
 - Inscrever-se em Aulas;
 - Cancelar Inscrição em uma Aula;
 - Avaliar Professor;
 - Acompanhar Lista de Espera;
 - Acompanhar evolução física.



PERSONAS - ALUNO

- Jornada: Realizar atividades físicas

The screenshots illustrate the following features:

- LISTAR AULA (Top Left):** Shows a list of classes with icons and details:
 - Boxe - Seg 9:00-10:30
 - Judô - Ter 9:00-10:30
 - Dança - Qua 9:00-10:30
 - Pilates - Qui 9:00-10:30
 - Karatê - Seg 9:00-10:30
 - Yoga - Seg 9:00-10:30
- RESERVAR AULA (Top Middle):** Shows a reservation screen for Boxe class by Professor Vínius de Andrade at 9:00-10:30. It includes 'Lista de presença' and 'Lista de espera'.
- CANCELAR AULA (Top Right):** A confirmation dialog asking "Você Realmente Deseja Cancelar a Aula?" (Do you really want to cancel the class?). It shows class details: Boxe, Professor Vínius de Andrade, and Horário 9:00-10:30.
- Histórias (Bottom Left):** A list of stories:
 - Listar Aulas de diferentes Modalidades;
 - Inscrever-se em Aulas;
 - Notificar e calcular reembolso;
- CANCELAR AULA (Bottom Middle):** Another confirmation dialog asking "GOSTARIA MESMO DE CANCELAR AULA?" (Would you like to cancel the class?). It shows the same class details as the previous cancel screen.

PERSONAS - ALUNO

- Jornada: Realizar atividades físicas

The image shows a sequence of five mobile application screens illustrating the user flow for evaluating a teacher:

- AVALIAR PROFESSOR**: A navigation bar with icons for Aula (green), Treino (blue), Squilda (orange), Torneio (yellow), Ranking (purple), and Evolução (red). Below the bar is a large red square button with a white square icon.
- Nome do Professor**: A circular placeholder for the teacher's name. Below it are four rating scales: Didática (Didactic), Atenção (Attention), Pontualidade (Punctuality), and Comentários (Comments). Each scale has a row of five diamond icons, some filled with yellow and some empty. A text input field labeled "Comentários" is at the bottom, and a red "Enviar" (Send) button is at the bottom right.
- Nome do Professor**: Similar to the previous screen, but the diamond icons in the rating scales are now all filled with yellow, indicating a higher rating. The "Enviar" button is still present.
- Nome do Professor**: The same layout as the second screen, but the rating scales show a mix of filled and empty diamonds. A text message "Foi muito atencioso em relação à minha forma." (Was very attentive regarding my form.) is displayed in a box at the bottom. The "Enviar" button is still present.
- Nome do Professor**: The same layout as the third screen, but the rating scales show all filled yellow diamonds. A confirmation message "Avaliação enviada com sucesso." (Evaluation sent successfully.) is displayed in a box, along with a "Confirmar" (Confirm) button. The "Enviar" button is still present.

- Histórias:
 - Avaliar Professor;
 - Listar a pontuação por atributo.

PERSONAS - ALUNO

- Jornada: Realizar atividades físicas

The screenshots illustrate the student's interaction with the app:

- Screenshot 1: ACEITAR PARTICIPAÇÃO EM AULA**
Shows the main menu with "Aula" selected. Below it, the "Boxe" section displays:
 - Professor: Vinicius de Andrade
 - 9:00-10:30
 - Lista de presença: 1. Thomas Lima, 2. Pedro, 3. Joabe, 4. Marcos
 - Lista de espera: 1.
 - Buttons: "Quero entrar na lista de espera" (blue) and "Quero sair da lista de espera" (pink).
- Screenshot 2: Boxe**
Shows the "Boxe" section after accepting participation:
 - Professor: Vinicius de Andrade
 - 9:00-10:30
 - Lista de presença: 1. Thomas Lima, 2. Paulo Rosado, 3. Domingos
 - Lista de espera: 1. Evaldo, 2. Saulo
 - Buttons: "Quero sair da lista de espera" (pink).
- Screenshot 3: ACOMPANHAMENTO DE BIOIMPEDANCIA**
Shows the "ACOMPANHAMENTO DE BIOIMPEDANCIA" section for Demetrius:
 - Demetrius
 - Massa muscular: 33kg
 - % de gordura: 20%
 - Comprimento do braço direito: 61cm
 - Comprimento do braço esquerdo: 60cm
 - Massa magra: 55kg
 - Comprimento da cintura: 80cm
 - Comprimento da perna direita: 92cm
 - Comprimento da perna esquerda: 92cm
- Screenshot 4: Boxe**
Shows the "Boxe" section after confirming presence:
 - Professor: Vinicius de Andrade
 - 9:00-10:30
 - Lista de presença: 1. Thomas Lima, 2. Pedro, 3. Evaldo
 - Lista de espera: 1. Saulo
 - Buttons: "Quero sair da lista de espera" (pink).A confirmation message at the top says: "Uma vaga foi liberada! Deseja confirmar sua presença?" with "Sim" (green) and "Não" (pink) buttons.



PERSONAS



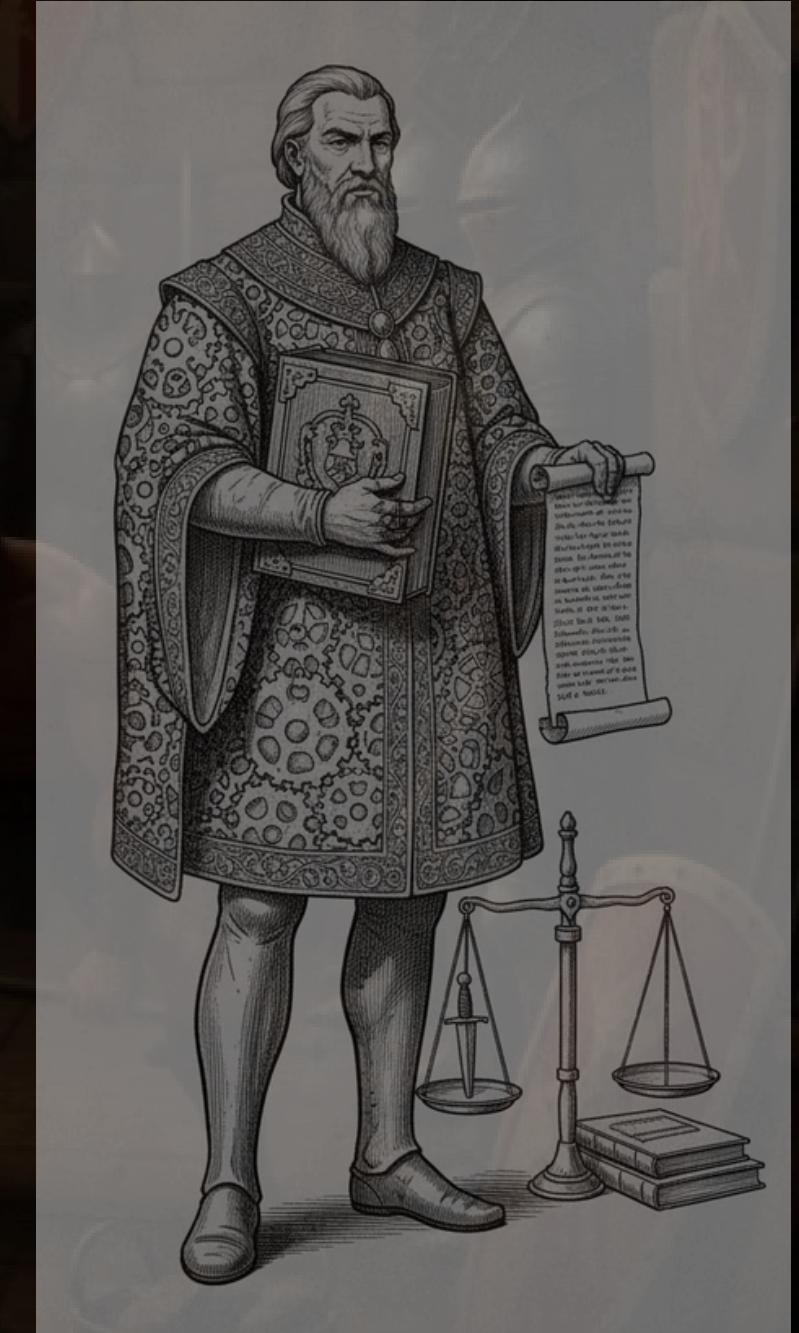
ALUNO



PROFESSOR



GERENTE



SISTEMA

PERSONAS – PROFESSOR

- **Jornada:** Treinar Alunos
- **Objetivo:** Organizar Aulas, Treinos & acompanhar o progresso dos Alunos.
- **Passos:**
 - Criar Aula;
 - Alterar data/hora da Aula;
 - Cancelar Aula;
 - Personalizar Treino de Aluno;
 - Avaliar Aluno;
 - Realizar Bioimpedância.



PERSONAS - PROFESSOR

- Jornada: Treinar Alunos

The image displays three wireframe prototypes for a mobile application interface, likely for a professor's digital dashboard. The prototypes are arranged vertically.

- Top Prototype (Left): CRIAR AULA (Create Class)**
 - Left sidebar: Buttons for 'Aula' (selected), 'Treino', and 'Evolução'.
 - Main area: A large circular '+' button labeled 'criar aula'.
 - Preview section: Shows a class named 'Boxe - Seg 9:00-10:30' with a red diamond icon.
- Top Prototype (Right): EXCLUIR AULA (Delete Class)**
 - Left sidebar: Buttons for 'Aula' (selected), 'Treino', and 'Evolução'.
 - Main area: A preview of a class named 'Boxe' with a red diamond icon.
 - Confirmation dialog: 'GOSTARIA MESMO DE CANCELAR A AULA?' with 'NÃO' (No) and 'CANCELAR AULA' (Cancel Class) buttons.
- Bottom Prototype: EDITAR AULA (Edit Class)**
 - Left sidebar: Buttons for 'Aula' (selected), 'Treino', and 'Evolução'.
 - Main area: A preview of a class named 'aula de boxe' with a red diamond icon.
 - Form fields:
 - name: 'aula de boxe'
 - horario: '00:00'
 - data: 'xx/xx/xxx'
 - descrição: Text input field
 - matriculados: List of students: ALAN PATRICK, NEYMAR JUNIOR, LUCAS LIMA, DERIK LACERDA, RAFAEL THYERE. Each student has a small red diamond icon next to their name.
 - Switch: 'TORNAR RECORRENTE?' (Make Recurrent?) with 'SIM' (Yes) and 'NÃO' (No) options.

- Histórias:

- Criar Aula / Aulas recorrentes;
- Editar data/hora das Aulas;
- Cancelar Aula;
- Encerrar recorrência de Aulas.

PERSONAS - PROFESSOR

- Jornada: Treinar Alunos

The interface consists of four main panels:

- Panel 1 (Top Left):** Shows a list of registered students ("matriculados") with edit icons. The list includes ALAN PATRICK, NEYMAR JUNIOR, LUCAS LIMA, DERIK LACERDA, and RAFAEL THYERE. A search bar at the bottom is labeled "pesquise".
- Panel 2 (Top Middle):** Shows the "Acompanhamento de Evolução" (Monitoring of Progress) for student ALAN PATRICK. It displays various physical measurements and body composition data. Buttons for "Enviar" (Send) and "Confirmar" (Confirm) are present.
- Panel 3 (Top Right):** Shows the "Acompanhamento de Evolução" for student ALAN PATRICK after data has been sent. A message box indicates "Dados de acompanhamento enviados com sucesso" (Monitoring data sent successfully). Buttons for "Enviar" (Send) and "Confirmar" (Confirm) are present.
- Panel 4 (Bottom):** Shows the "Treino" (Training) screen for student ALAN PATRICK. It includes buttons for exercises A, B, C, and a plus sign. Below are lists for "perna" (leg), "costas" (back), "biceps" (biceps), and "inferiores" (inferior). On the right, a list of five exercises (exercicio 1 to 5) is shown with a "SALVAR" (Save) button.

- Histórias:
 - Criar Treino de Aluno;
 - Atualizar Treino de Aluno;
 - Salvar dados corporais de um Aluno.



PERSONAS



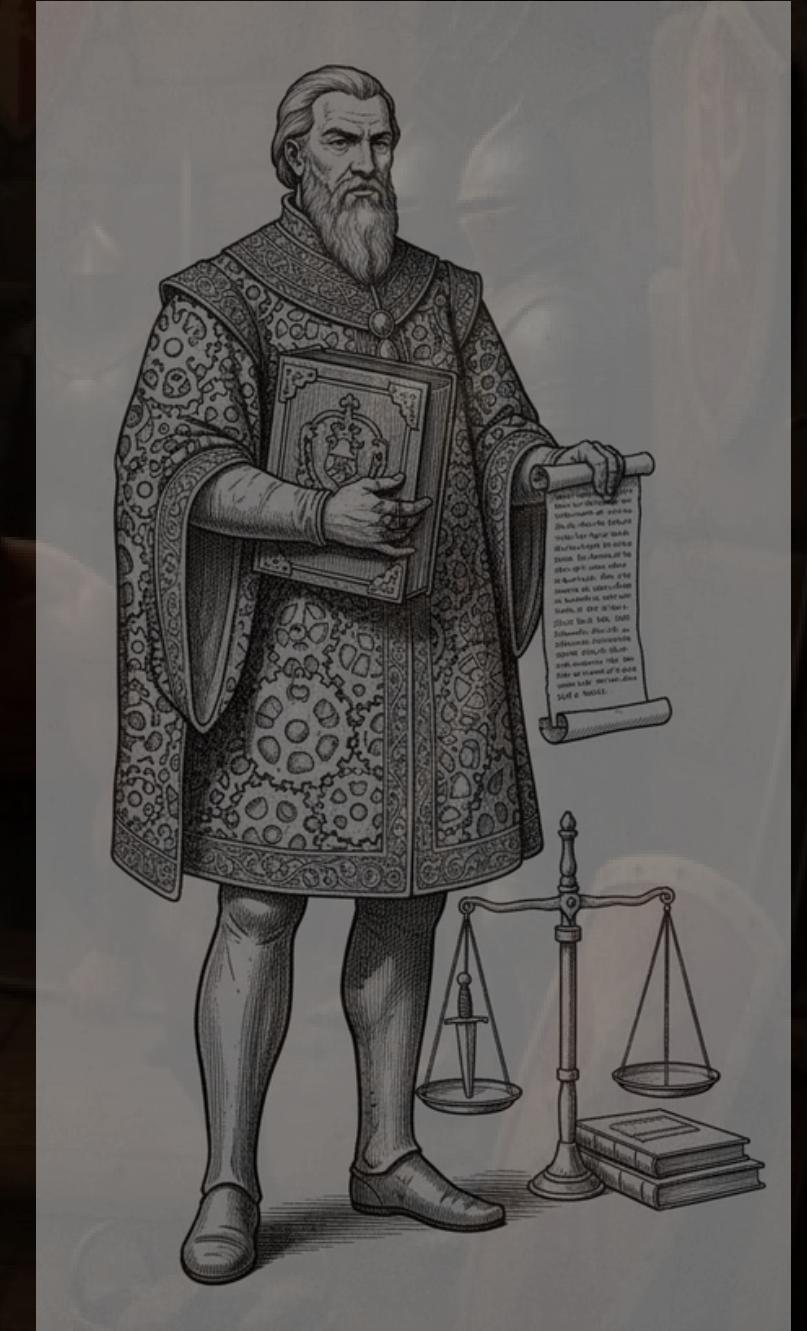
ALUNO



PROFESSOR



GERENTE



SISTEMA



PERSONAS - GERENTE

- **Jornada:** Proporcionar competições que incentivem os Alunos
- **Objetivo:** Criar competições & Torneios entre Guildas para aumentar o engajamento.
- **Passos:**
 - Iniciar um Torneio entre Guildas;
 - Alterar dados de um Torneio;
 - Cancelar um Torneio.





PERSONAS - GERENTE

- **Jornada:** Proporcionar competições que incentivem os Alunos

- **Histórias:**

- Iniciar um Torneio;
- Definir os Prêmios do Torneio;
- Alterar dados de um Torneio;
- Cancelar um Torneio.

The image displays six wireframe prototypes for a tournament management application, arranged in a 2x3 grid. Each prototype includes a sidebar with navigation buttons: Aula, Treino, Guilda, **Torneio**, Ranking, and Evolução.

- Criar torneio:** Shows a large button labeled "Criar torneio" and a "Visualizar anteriores" button.
- Definir premiações:** Shows a tournament titled "Torneio de São João" (Ending in 12 days) with a top prize of "#1 Kit 1kg Whey + 600g Creatina". It includes a pencil icon for editing and a trash icon for deleting.
- Alterar Torneio:** Shows the same tournament details with fields for "Nome do torneio" (Torneio de São João), "Data de inicio" (22/10/25), and "Data de Fim" (29/10/25). Buttons for "SALVAR ALTERAÇÕES" and "CANCELAR TORNEIO" are present.
- Iniciar torneio:** Shows fields for "Nome do torneio", "Data de inicio", and "Data de Fim", along with "CRIAR" and "CANCELAR" buttons.
- Premiação do 2º lugar:** A modal dialog for defining second-place prizes, showing fields for "Prêmio" and "URL da Imagem", with "CONFIRMAR" and "VOLTAR" buttons.
- Torneio de São João:** A confirmation dialog asking "Tem certeza que deseja cancelar?", with "CONFIRMAR" and "VOLTAR" buttons.



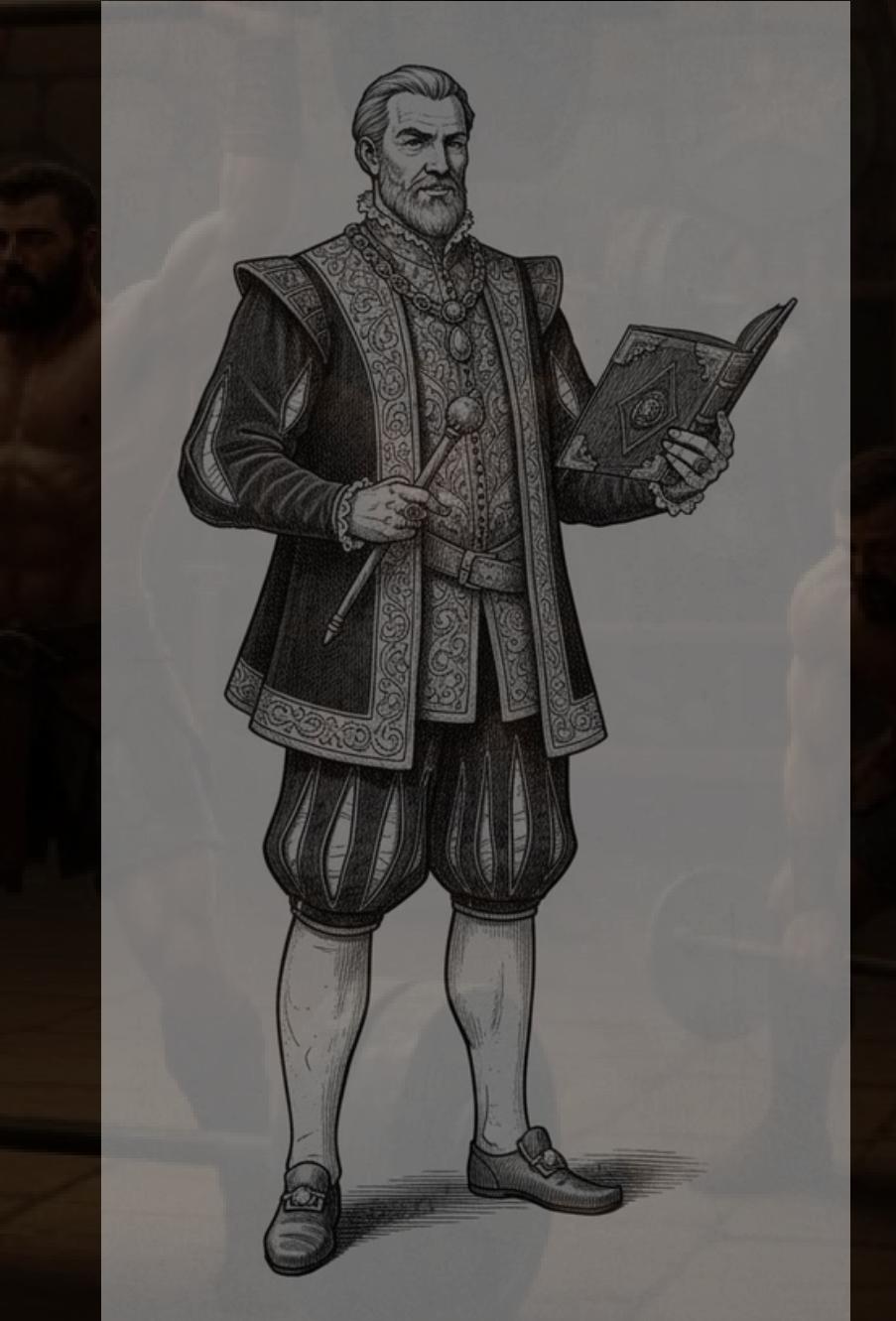
PERSONAS



ALUNO



PROFESSOR



GERENTE



SISTEMA

PERSONAS – SISTEMA

- **Jornada:** Automação & Controle
- **Objetivo:** Garantir o funcionamento automatizado & eficiente do ambiente.

- **Passos:**

- Maximizar a ocupação de uma Aula;
- Controlar Frequência dos Alunos;
- Promover automaticamente Alunos da Lista de Espera;
- Bloquear novas Inscrições por excesso de Faltas;
- Desbloquear Aluno após período determinado;
- Integrar com sistema de catraca inteligente.



PERSONAS – SISTEMA

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Cenários de Teste:**
 - Cancelamento com reembolso total processado;
 - Tentativa de cancelamento após prazo de reembolso total.

```
cancelamento_de_reserva.feature features ×  
projeto > dominio > src > test > resources > features > cancelamento_de_reserva.feature  
1 Feature: Cancelamento de reserva com política de reembolso  
2  
3 # Regra 1 – Cancelamento com antecedência suficiente (reembolso total em até 15 dias)  
4 Scenario: Cancelamento com reembolso total processado  
5 Given existe uma reserva confirmada para o dia "10/09/2025" às "10:00" com duração de "60 minutos"  
6 When o aluno solicita o cancelamento em "20/08/2025"  
7 Then o sistema informa "cancelamento aprovado, reembolso integral em processamento"  
8  
9 Scenario: Tentativa de cancelamento após prazo de reembolso total  
10 Given existe uma reserva confirmada para o dia "10/09/2025" às "10:00" com duração de "60 minutos"  
11 When o aluno solicita o cancelamento em "01/09/2025"  
12 Then o sistema informa "cancelamento aprovado, reembolso parcial em processamento"  
13
```

PERSONAS - SISTEMA

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Automação dos testes BDD com Cucumber:**
 - CancelamentoDeReservaFuncionalidade - @Given

```
● ● ●  
1  @Given("existe uma reserva confirmada para o dia {string} às {string} com duração de {string}")  
2  public void existe_uma_reserva_confirmada_para_o_dia(String dataStr, String horario, String duracao) {  
3      cpfAluno = new Cpf("12345678900");  
4      Aluno aluno = new Aluno(cpfAluno);  
5      aluno.setNome("Aluno Teste");  
6      contexto.repositorio.salvar(aluno);  
7  
8      LocalDate data = LocalDate.parse(dataStr, dateFormatter);  
9      String[] partesHorario = horario.split(":");  
10     int hora = Integer.parseInt(partesHorario[0]);  
11     int minuto = Integer.parseInt(partesHorario[1]);  
12  
13     int duracaoMinutos = Integer.parseInt(duracao.replace(" minutos", "").trim());  
14  
15     LocalDateTime inicio = LocalDateTime.of(data, java.time.LocalTime.of(hora, minuto));  
16     LocalDateTime fim = inicio.plusMinutes(duracaoMinutos);  
17  
18     aulaCriada = contexto.aulaService.criarAulaUnica(  
19         Modalidade.YOGA,  
20         Espaco.ESTUDIO_PILATES,  
21         20,  
22         inicio,  
23         fim  
24     );  
25  
26     reservaCriada = contexto.reservaService.reservarVaga(cpfAluno, aulaCriada.getId());  
27 }
```

PERSONAS – SISTEMA

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Automação dos testes BDD com Cucumber:**
 - CancelamentoDeReservaFuncionalidade - @When

```
● ● ●
1 @When("o aluno solicita o cancelamento em {string}")
2 public void o_aluno_solicita_o_cancelamento_em(String dataStr) {
3     dataCancelamento = LocalDate.parse(dataStr, dateFormatter);
4     LocalDateTime momentoCancelamento = dataCancelamento.atStartOfDay();
5
6     try {
7         double credito = contexto.reembolsoService.calcularCreditoDeReembolso(
8             aulaCriada.getId(),
9             aulaCriada,
10            momentoCancelamento
11        );
12
13     contexto.reservaService.cancelarReserva(cpfAluno, aulaCriada.getId(), momentoCancelamento);
14
15     double valorBase = 20.0;
16     if (credito ≥ valorBase * 0.99) {
17         mensagemResposta = "cancelamento aprovado, reembolso integral em processamento";
18     } else if (credito ≥ valorBase * 0.49) {
19         mensagemResposta = "cancelamento aprovado, reembolso parcial em processamento";
20     } else {
21         mensagemResposta = "cancelamento realizado sem direito a reembolso";
22     }
23 } catch (IllegalArgumentException e) {
24     contexto.excecao = e;
25     mensagemResposta = "nenhuma reserva encontrada para cancelamento";
26 } catch (Exception e) {
27     contexto.excecao = e;
28     mensagemResposta = e.getMessage();
29 }
30 }
```



PERSONAS - SISTEMA

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Automação dos testes BDD com Cucumber:**
 - CancelamentoDeReservaFuncionalidade - @Then

```
● ● ●  
1 @Then("o sistema informa {string}")  
2 public void o_sistema_informa(String mensagemEsperada) {  
3     assertNotNull(mensagemResposta);  
4     assertEquals(mensagemEsperada, mensagemResposta);  
5 }
```



PERSONAS – SISTEMA

```
cpfAluno = new Cpf("12345678900");
Aluno aluno = new Aluno(cpfAluno);
aluno.setNome("Aluno Teste");
contexto.repositorio.salvar(aluno);
```

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Código necessário para os testes serem bem sucedidos:**
 - Cpf, Aluno, AcademiaFuncionalidade

```
6 import java.util.Objects;
7
8 public class Cpf {
9     private final String numero;
10
11    public Cpf(String numero) {
12        notNull(numero, "O CPF não pode ser nulo.");
13        notBlank(numero, "O CPF não pode estar em branco.");
14
15        if (!numero.matches("\\d{11}")) {
16            throw new IllegalArgumentException("Formato de CPF inválido. Deve ter 11 dígitos numéricos.");
17        }
18
19        this.numero = numero;
20    }
21
22    public String getNumero() {
23        return numero;
24    }
25
26    @Override
27    public boolean equals(Object obj) {
28        if (obj != null && obj instanceof Cpf) {
29            Cpf other = (Cpf) obj;
30            return numero.equals(other.numero);
31        }
32        return false;
33    }
34
35    @Override
36    public int hashCode() {
37        return Objects.hash(numero);
38    }
39
40    @Override
41    public String toString() {
42        return numero;
43    }
44 }
```

```
12
13    public class Aluno {
14        private final Cpf cpf;
15        private String nome;
16        private Matricula matricula;
17        private LocalDate dataNascimento;
18        private int pontuacaoTotal;
19        private double creditos;
20        private StatusAluno status;
21        private LocalDate bloqueioAte;
22        private PlanoDeTreino planoAtivo;
23        private GuildaId guildaId;
24
25        public Aluno(Cpf cpf) {
26            notNull(cpf, "O CPF não pode ser nulo");
27            this.cpf = cpf;
28            this.status = StatusAluno.ATIVO;
29            this.pontuacaoTotal = 0;
30            this.creditos = 0.0;
31        }
32
33        public Aluno(Cpf cpf, StatusAluno status) {
34            notNull(cpf, "O CPF não pode ser nulo");
35            this.cpf = cpf;
36            this.status = status;
37            this.pontuacaoTotal = 0;
38            this.creditos = 0.0;
39        }
40
41        public Cpf getCpf() {
42            return cpf;
43        }
44
45        public String getNome() {
46            return nome;
47        }
48
49        public void setNome(String nome) {
```

```
10
11        public class AcademiaFuncionalidade implements EventoBarramento {
12
13            public final Repositorio repositorio;
14            public final GuildaService guildaService;
15            public final CheckinService checkinService;
16            public final TorneioService torneioService;
17            public final AulaService aulaService;
18            public final ReservaService reservaService;
19            public final RankingService rankingService;
20            public final AvaliacaoService avaliacaoService;
21            public final ReembolsoService reembolsoService;
22            public final TreinoService treinoService;
23            public final FrequenciaService frequenciaService;
24
25            public List<Object> eventos;
26            public Exception excecao;
27
28            public AcademiaFuncionalidade() {
29                this.repositorio = new Repositorio();
30                this.eventos = new ArrayList<>();
31
32                // Inicializa serviços implementados
33                this.guildaService = new GuildaService(this.repositorio);
34                this.checkinService = new CheckinService(this.repositorio, this.repositorio, this.repositorio);
35                this.torneioService = new TorneioService(this.repositorio);
36                thisaulaService = new AulaService(this.repositorio);
37                this.reembolsoService = new ReembolsoService();
38                this.reservaService = new ReservaService(this.repositorio, this.repositorio, this.reembolsoService);
39                this.rankingService = new RankingService(this.repositorio);
40                this.avaliacaoService = new AvaliacaoService(this.repositorio);
41                this.treinoService = new TreinoService(this.repositorio);
42                this.frequenciaService = new FrequenciaService(this.repositorio, this.repositorio, this.repositorio);
43            }
44
45            @Override
46            public void postar(Object evento) {
47                if (evento == null) {
48                    throw new IllegalArgumentException("Evento não pode ser nulo");
49                }
50                eventos.add(evento);
51            }
52
53            public void resetarContexto() {
54                this.excecao = null;
55                this.eventos.clear();
56            }
57
58        }
```



PERSONAS – SISTEMA

```
cpfAluno = new Cpf("12345678900");
Aluno aluno = new Aluno(cpfAluno);
aluno.setNome("Aluno Teste");
contexto.repositorio.salvar(aluno);
```

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Código necessário para os testes serem bem sucedidos:**
 - Repositorio, AlunoRepositorio

```
● ● ●
1 public class Repositorio implements AlunoRepositorio, RepositorioGeral, GuildaRepositorio,
2   CheckinRepositorio, TorneioRepositorio, AulaRepositorio,
3   RankingRepositorio, AvaliacaoRepositorio, TreinoRepositorio,
4   FrequenciaRepositorio {
5
6     private Map<Cpf, Aluno> alunos = new HashMap<>();
7
8     @Override
9     public void salvar(Aluno aluno) {
10       notNull(aluno, "O aluno não pode ser nulo");
11       alunos.put(aluno.getCpf(), aluno);
12     }
13
14     @Override
15     public Optional<Aluno> obterPorCpf(Cpf cpf) {
16       notNull(cpf, "O CPF não pode ser nulo");
17       return Optional.ofNullable(alunos.get(cpf));
18     }
19
20     private Map<GuildaId, Guilda> guildas = new HashMap<>();
21
22     @Override
23     public void salvar(Guilda guilda) {
24       notNull(guilda, "A guilda não pode ser nula").
```

```
● ● ●
1 package br.com.forgefit.dominio.aluno;
2
3 import java.util.Optional;
4
5 public interface AlunoRepositorio {
6   void salvar(Aluno aluno);
7   Optional<Aluno> obterPorCpf(Cpf cpf);
8 }
```



PERSONAS - SISTEMA

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Código necessário para os testes serem bem sucedidos:**
 - AulaService, AulaRepositorio, criarAulaUnica(), verificarConflitoHorario(), temConflito()

```
public class AulaService {  
    private final AulaRepositorio aulaRepositorio;  
    private final AtomicInteger contadorId = new AtomicInteger(1);  
  
    public AulaService(AulaRepositorio aulaRepositorio) {  
        notNull(aulaRepositorio, "O repositório de aulas não pode ser nulo");  
        this.aulaRepositorio = aulaRepositorio;  
    }  
  
    public Aula criarAulaUnica(Modalidade modalidade, Espaco espaco, int capacidade,  
                               LocalDateTime inicio, LocalDateTime fim) {  
  
        verificarConflitoHorario(espaco, inicio, fim);  
  
        AulaId id = new AulaId(contadorId.getAndIncrement());  
        Aula aula = new Aula(id, modalidade, espaco, capacidade, inicio, fim);  
        aulaRepositorio.salvar(aula);  
        return aula;  
    }  
}
```

```
public interface AulaRepositorio {  
    void salvar(Aula aula);  
    Optional<Aula> obterPorId(AulaId aulaId);  
    List<Aula> listarTodas();  
    List<Aula> buscarPorEspacoEPeriodo(Espaco espaco, LocalDateTime inicio, LocalDateTime fim);  
}
```

```
aulaCriada = contexto.aulaService.criarAulaUnica(  
    Modalidade.YOGA,  
    Espaco.ESTUDIO_PILATES,  
    20,  
    inicio,  
    fim  
);  
  
reservaCriada = contexto.reservaService.reservarVaga(cpfaAluno, aulaCriada.getId());
```

```
private void verificarConflitoHorario(Espaco espaco, LocalDateTime inicio, LocalDateTime fim) {  
    verificarConflitoHorario(espaco, inicio, fim, null);  
}  
  
private void verificarConflitoHorario(Espaco espaco, LocalDateTime inicio, LocalDateTime fim, AulaId aulaIdExcluir) {  
    if (temConflito(espaco, inicio, fim, aulaIdExcluir)) {  
        throw new IllegalStateException(  
            String.format("%s às %s já está ocupado para aula de %s",  
                inicio.toLocalDate().format(java.time.format.DateTimeFormatter.ofPattern("dd/MM/yyyy")),  
                inicio.toLocalTime().format(java.time.format.DateTimeFormatter.ofPattern("HH:mm")),  
                espaco.name().toLowerCase().replace("_", " ")));  
    }  
}  
  
private boolean temConflito(Espaco espaco, LocalDateTime inicio, LocalDateTime fim) {  
    return temConflito(espaco, inicio, fim, null);  
}  
  
private boolean temConflito(Espaco espaco, LocalDateTime inicio, LocalDateTime fim, AulaId aulaIdExcluir) {  
    List<Aula> aulasNoEspaco = aulaRepositorio.buscarPorEspacoEPeriodo(espaco, inicio, fim);  
  
    return aulasNoEspaco.stream()  
        .filter(a → a.getStatus() == StatusAula.ATIVA)  
        .filter(a → aulaIdExcluir == null || !a.getId().equals(aulaIdExcluir))  
        .anyMatch(a →  
            (inicio.isBefore(a.getFim()) && fim.isAfter(a.getInicio())) ||  
            inicio.equals(a.getInicio())  
        );  
}
```



PERSONAS - SISTEMA

- **Funcionalidade:** Cancelamento de Reserva com política de reembolso
- **Código necessário para os testes serem bem sucedidos:**
 - ReservaService, Reserva, PosicaoListaDeEspera, reservarVaga(), entrarNaListaDeEspera()

```
public class ReservaService {  
    private final AulaRepositorio aulaRepositorio;  
    private final AlunoRepositorio alunoRepositorio;  
    private final ReembolsoService reembolsoService;  
  
    public ReservaService(AulaRepositorio aulaRepositorio, AlunoRepositorio alunoRepositorio,  
        ReembolsoService reembolsoService) {  
        notNull(aulaRepositorio, "O repositório de aulas não pode ser nulo");  
        notNull(alunoRepositorio, "O repositório de alunos não pode ser nulo");  
        notNull(reembolsoService, "O serviço de reembolso não pode ser nulo");  
        this.aulaRepositorio = aulaRepositorio;  
        this.alunoRepositorio = alunoRepositorio;  
        this.reembolsoService = reembolsoService;  
    }  
  
    public Reserva reservarVaga(Cpf alunoId, AulaId aulaId) {  
        notNull(alunoId, "O CPF do aluno não pode ser nulo");  
        notNull(aulaId, "O id da aula não pode ser nulo");  
  
        var aula = aulaRepositorio.obterPorId(aulaId)  
            .orElseThrow(() -> new IllegalArgumentException("Aula não encontrada"));  
  
        if (aula.temVagaDisponivel()) {  
            var reserva = new Reserva(alunoId, LocalDateTime.now());  
            aula.adicionarReserva(reserva);  
            aulaRepositorio.salvar(aula);  
            return reserva;  
        } else {  
            entrarNaListaDeEspera(alunoId, aulaId);  
            return null;  
        }  
    }  
}
```

```
public void entrarNaListaDeEspera(Cpf alunoId, AulaId aulaId) {  
    notNull(alunoId, "O CPF do aluno não pode ser nulo");  
    notNull(aulaId, "O id da aula não pode ser nulo");  
  
    var aula = aulaRepositorio.obterPorId(aulaId)  
        .orElseThrow(() -> new IllegalArgumentException("Aula não encontrada"));  
  
    var posicao = new PosicaoListaDeEspera(alunoId, LocalDateTime.now());  
    aula.adicionarNaListaDeEspera(posicao);  
    aulaRepositorio.salvar(aula);  
}  
  
public class PosicaoListaDeEspera {  
    private final Cpf alunoId;  
    private final LocalDateTime timestampDeEntrada;  
  
    public PosicaoListaDeEspera(Cpf alunoId, LocalDateTime timestampDeEntrada) {  
        notNull(alunoId, "O CPF do aluno não pode ser nulo");  
        this.alunoId = alunoId;  
  
        notNull(timestampDeEntrada, "O timestamp de entrada não pode ser nulo");  
        this.timestampDeEntrada = timestampDeEntrada;  
    }  
  
    public Cpf getAlunoId() {  
        return alunoId;  
    }  
  
    public LocalDateTime getTimestampDeEntrada() {  
        return timestampDeEntrada;  
    }  
}  
  
public class Reserva {  
    private final Cpf alunoId;  
    private final LocalDateTime dataDaReserva;  
    private StatusReserva status;  
  
    public Reserva(Cpf alunoId, LocalDateTime dataDaReserva) {  
        notNull(alunoId, "O CPF do aluno não pode ser nulo");  
        notNull(dataDaReserva, "A data da reserva não pode ser nula");  
        this.alunoId = alunoId;  
        this.dataDaReserva = dataDaReserva;  
        this.status = StatusReserva.CONFIRMADA;  
    }  
}  
public Cpf getAlunoId() {  
    return alunoId;  
}
```

```
aulaCriada = contexto.aulaService.criarAulaUnica(  
    Modalidade.YOGA,  
    Espaco.ESTUDIO_PILATES,  
    20,  
    inicio,  
    fim  
);  
  
reservaCriada = contexto.reservaService.reservarVaga(cpfaAluno, aulaCriada.getId());
```



PERSONAS - SISTEMA

- Funcionalidade: Cancelamento de Reserva com política de reembolso
- Código necessário para os testes serem bem sucedidos:
 - ReembolsoService, cancelarReserva(), promoverPrimeiroDaListaSeHouver()

```
public class ReembolsoService {  
  
    private static final long HORAS_MINIMAS_REEMBOLSO_TOTAL = 15 * 24;  
    private static final double PERCENTUAL_REEMBOLSO_TOTAL = 1.0;  
    private static final double PERCENTUAL_REEMBOLSO_PARCIAL = 0.5;  
    private static final long HORAS_MINIMAS_REEMBOLSO_PARCIAL = 5 * 24;  
  
    public double calcularCreditoDeReembolso(AulaId aulaId, Aula aula, LocalDateTime momentoCancelamento) {  
        notNull(aulaId, "O ID da aula não pode ser nulo");  
        notNull(aula, "A aula não pode ser nula");  
        notNull(momentoCancelamento, "O momento do cancelamento não pode ser nulo");  
  
        LocalDateTime inicioAula = aula.getInicio();  
        Duration antecedencia = Duration.between(momentoCancelamento, inicioAula);  
        long horasAntecedencia = antecedencia.toHours();  
  
        if (horasAntecedencia >= HORAS_MINIMAS_REEMBOLSO_TOTAL) {  
            return calcularValorBase() * PERCENTUAL_REEMBOLSO_TOTAL;  
        }  
  
        if (horasAntecedencia >= HORAS_MINIMAS_REEMBOLSO_PARCIAL) {  
            return calcularValorBase() * PERCENTUAL_REEMBOLSO_PARCIAL;  
        }  
  
        return 0.0;  
    }  
  
    private double calcularValorBase() {  
        return 20.0;  
    }  
  
    public boolean temDireitoAReembolso(Aula aula, LocalDateTime momentoCancelamento) {  
        return calcularCreditoDeReembolso(aula.getId(), aula, momentoCancelamento) > 0;  
    }  
}
```

```
try {  
    double credito = contexto.reembolsoService.calcularCreditoDeReembolso(  
        aulaCriada.getId(),  
        aulaCriada,  
        momentoCancelamento  
    );  
  
    contexto.reservaService.cancelarReserva(cpfAluno, aulaCriada.getId(), momentoCancelamento);  
}  
  
public void cancelarReserva(Cpf alunoId, AulaId aulaId) {  
    cancelarReserva(alunoId, aulaId, LocalDateTime.now());  
}  
  
public void cancelarReserva(Cpf alunoId, AulaId aulaId, LocalDateTime momentoCancelamento) {  
    notNull(alunoId, "O CPF do aluno não pode ser nulo");  
    notNull(aulaId, "O id da aula não pode ser nulo");  
    notNull(momentoCancelamento, "O momento do cancelamento não pode ser nulo");  
  
    var aula = aulaRepository.obterPorId(aulaId)  
        .orElseThrow(() -> new IllegalArgumentException("Aula não encontrada"));  
  
    double credito = reembolsoService.calcularCreditoDeReembolso(aulaId, aula, momentoCancelamento);  
  
    if (credito > 0) {  
        var aluno = alunoRepository.obterPorCpf(alunoId)  
            .orElseThrow(() -> new IllegalArgumentException("Aluno não encontrado"));  
        aluno.adicionarCreditos(credito);  
        alunoRepository.salvar(aluno);  
    }  
  
    aula.cancelarReserva(alunoId);  
    aulaRepository.salvar(aula);  
  
    promoverPrimeiroDaListaSeHouver(aula);  
}  
  
private void promoverPrimeiroDaListaSeHouver(Aula aula) {  
    if (!aula.getListaDeEspera().isEmpty() && aula.temVagaDisponivel()) {  
        var proximoOpt = aula.removerPrimeiroDaListaDeEspera();  
        if (proximoOpt.isPresent()) {  
            var reserva = new Reserva(proximoOpt.get().getAlunoId(), LocalDateTime.now());  
            aula.adicionarReserva(reserva);  
            aulaRepository.salvar(aula);  
        }  
    }  
}
```

GUILDA FORGEFIT



GUSTAVO
MOURATO



LEONARDO
MATOS



PAULO
ROSADO



THOMAZ
LIMA



VINÍCIUS
DE ANDRADE



NA FORGEFIT NÓS NÃO APENAS LEVANTAMOS FERRO,
NÓS FORJAMOS FORÇA!