

*Registro de Mudanças*

| **Versão** | **Data de Mudança** | **Por** | **Descrição** |
| --- | --- | --- | --- |
| 1.0 | 03/11/2024 | Heron Lancellot e Savio Suhett | Criação do documento |
| 1.1 | 04/11/2024 | Eduardo Saldumbides | Atualização e formatação da documentação. |
| 1.2 | 04/11/2024 | Diego Camargo | Atualização da Documentação |
| 1.3 | 09/11/2024 | Diego Camargo e Rafaela Peçanha | Adaptação do documento segundo feedbacks |
| 1.4 | 27/01/2025 | Eduardo Saldumbides | Análise de Prioridade das Características de Qualidade. |
| 1.5 | 28/01/2025 | Eduardo Saldumbides e Heron Lancellot | Relatório de Inspeção do Código Fonte |
| 1.6 | 28/01/2025 | Eduardo Saldumbides | Inspeção do Código Fonte |
| 1.7 | 28/01/2025 | Heron Lancellot | Inspeção do Código Fonte |
| 1.8 | 28/01/2025 | Savio Suhett | Inspeção do Código Fonte |
| 1.9 | 28/01/2025 | Rafaela Peçanha | Inspeção do Código Fonte |
| 2.0 | 28/01/2025 | Diego Camargo | Inspeção do Código Fonte |
| 2.1 | 29/01/2024 | Heron Lancellot | Projeto de Melhora de Casos de Testes |

**1** **INTRODUÇÃO 2**

1.1 Escopo 2

*1.1.1* *No escopo 2*

*1.1.2* *Fora do escopo 2*

1.2 Objetivos de Qualidade 3

1.3 Papéis e Responsabilidades 3

**2** **METODOLOGIA DE TESTE 3**

2.1 Fases de Teste 3

2.2 Critérios de Suspensão e Requisitos de Retomada 3

2.3 Completude do Teste 4

2.4 Atividades do projeto, estimativas e cronograma 4

**3** **ENTREGÁVEIS DE TESTE 4**

**4** **NECESSIDADES DE RECURSOS E AMBIENTE 5**

4.1 Ferramentas de Teste 5

4.2 Ambiente de Teste 5

**5** **TERMOS / ACRÔNIMOS**

**6 ANÁLISE DE PRIORIDADE DAS CARACTERÍSITICAS DE QUALIDADE**

**7 PROJETO DE MELHORA DE CASOS DE TESTES**

**8 RELATÓRIO DE INSPEÇÃO DO CÓDIGO-FONTE**

# Introdução

O plano de testes do **Sistema de Gerenciamento de Assentos de Voo** visa garantir que o sistema atenda aos requisitos funcionais e não funcionais, assegurando sua qualidade, desempenho e segurança. Serão realizados testes de funcionalidade, segurança, desempenho, usabilidade e compatibilidade, com foco na validação de processos como a criação e atualização de assentos, aprovação de preços, e compra de passagens pelos clientes. O processo de testes seguirá uma abordagem ágil e estruturada. O objetivo é fornecer um software robusto e eficiente, proporcionando uma experiência satisfatória para todos os usuários, desde administradores até clientes finais.

## Escopo

### No escopo

O escopo define os recursos e funcionalidades que serão testados no sistema de gerenciamento de assentos de voo, incluindo os requisitos funcionais e não funcionais.

#### Requisitos Funcionais:

1. **Administração de Assentos e Preços:**
   * O Admin consegue adicionar, editar e definir os preços de diferentes tipos de assentos.
   * O Admin consegue definir o número total de assentos disponíveis por voo.
2. **Aprovação de Alterações pelo Gerente:**
   * O Gerente tem acesso a uma interface para visualizar e aprovar ou rejeitar as alterações feitas pelo Admin.
3. **Seleção de Assentos pelo Cliente:**
   * O Cliente consegue escolher assentos disponíveis com base na origem, destino, data da viagem e número de passageiros.
   * O Cliente consegue visualizar e aprovar o itinerário gerado após a seleção do assento.
4. **Compra de Assentos e Processamento de Pagamento:**
   * O Cliente é redirecionado para a página de pagamento onde o preço total é exibido.
   * O Cliente pode completar a transação de pagamento e concluir a compra do assento.
5. **Controle de Estoque de Assentos:**
   * O sistema impede a compra de mais assentos quando todos os assentos estiverem vendidos.
6. **Envio de E-mail de Confirmação:**
   * O sistema envia um e-mail de confirmação com o itinerário de viagem após a compra ser finalizada.

#### Requisitos Não Funcionais:

1. **Usabilidade:**
   * Validar o fluxo de reservas e pagamentos para garantir que o Cliente consiga completar sua compra de maneira simples e sem confusão.
2. **Manutenibilidade:**
   * O código e a estrutura do sistema devem ser modularizados para facilitar a manutenção e atualizações.

Na tabela abaixo é possível observar os módulos do sistema, os papéis dos usuários que podem desempenhar este módulo e a descrição.

| **Nome do Módulo** | **Papéis aplicáveis** | **Descrição** |
| --- | --- | --- |
| Login | Usuário | **Usuário:** utiliza suas credenciais para acessar no sistema |
| Logout | Usuário | **Usuário:** encerra a sua sessão do sistema |
| Gerenciar assentos de voo | Administrador | **Administrador:** Pode visualizar o número de assentos de cada classe e modificar esse número e seus preços |
| Aprovar o novo preço ou as atualizações | Manager | **Manager:** Após modificarem o preço ou número de assentos o Manager precisa aprová-los antes de atualizar para o cliente |
| Procurar vôos | Cliente | **Cliente:** Pode selecionar alguns filtros e o sistema retorna todos os vôos disponíveis baseados neles |
| Selecione Itinerário | Cliente | **Cliente:** Baseado em uma busca o cliente pode selecionar um vôo disponível e reserva-lo |
| Reservas atuais | Cliente | **Cliente:** Pode visualizar todos os itinerários que ele reservou |
| Recebimento de e-mail confirmando assentos | Cliente | **Cliente:** Recebe um email de confirmando o assento dele ao comprar um Itinerário |

### Fora do escopo

Esses recursos não serão testados porque não estão incluídos nas especificações de requisitos do software

* **Desempenho:**
  + O sistema deve ser capaz de suportar múltiplos usuários simultâneos sem degradação de desempenho.
  + A resposta do sistema ao selecionar assentos e ao processar pagamentos deve ser rápida (idealmente abaixo de 2 segundos para as ações comuns).
* **Segurança:**
  + As transações financeiras devem ser seguras, com criptografia de dados.
  + O sistema deve garantir que informações sensíveis, como dados de pagamento, sejam protegidas.
  + Deve ser testada a autenticação e autorização para Admins, Gerentes e Clientes.
* **Escalabilidade:**
  + O sistema deve ser capaz de lidar com uma quantidade crescente de assentos, voos e usuários, sem perda de performance.
* **Compatibilidade:**
  + O sistema deve ser compatível com os navegadores mais populares (Chrome, Firefox, Safari, Edge).
  + A interface deve ser responsiva, funcionando bem em dispositivos móveis e desktops.
* **Backup e Recuperação:**
  + O sistema deve realizar backups automáticos para garantir a recuperação de dados em caso de falha.
* **Monitoramento e Logs:**
  + O sistema deve registrar logs de atividades críticas, como transações de pagamento e alterações nos assentos, para auditoria e análise.

## Objetivos de Qualidade

O objetivo principal deste projeto de teste é garantir que o **Sistema de Gerenciamento de Assentos de Voo** esteja em conformidade com os requisitos funcionais e não funcionais definidos, atendendo às especificações de qualidade estabelecidas pelo cliente. Buscamos validar que todas as funcionalidades do sistema operam conforme esperado, sem falhas. Além disso, o objetivo é identificar quaisquer defeitos ou problemas, assegurando que o software seja robusto e capaz de oferecer uma experiência de usuário satisfatória. Isso permitirá que o sistema seja confiável e eficiente para todos os stakeholders envolvidos, desde administradores até os clientes finais.

## Papéis e Responsabilidades

Em nosso projeto, todos os membros da equipe atuam de forma colaborativa. Cada membro é tanto **projetista de teste** quanto **testador** e **analista de QA** para as áreas de sua especialização. Dessa forma, buscamos uma abordagem ágil e integrada, onde a comunicação entre os membros da equipe é constante e transparente, garantindo um ciclo contínuo de melhoria nos testes do sistema.

# Metodologia de Teste

## Visão Geral

A **metodologia ágil** foi escolhida para o projeto do **Sistema de Gerenciamento de Assentos de Voo** devido à sua flexibilidade, capacidade de adaptação a mudanças de requisitos e foco na colaboração contínua entre as equipes. Essa abordagem permite entregas incrementais e testes contínuos, garantindo que o sistema esteja sempre em um estado testável e alinhado com as expectativas do cliente. Com ciclos curtos de desenvolvimento e validação (sprints), a metodologia ágil facilita a identificação precoce de defeitos, a implementação de ajustes rápidos e a melhoria constante do software, proporcionando uma solução de alta qualidade e prontidão para o mercado.

## Fases de Teste

No projeto Flight Booking System, serão conduzidas as seguintes fases de teste:

* **Testes Unitários (JUnit)**: Teste de módulos isolados para garantir o funcionamento correto de funções individuais, utilizando abordagens de caixa branca.
* **Testes de Integração**: Validação da interação entre os módulos do sistema, garantindo que as integrações funcionem como esperado, utilizando abordagens de caixa branca e caixa preta.
* **Testes Manuais (TestLink)**: Testes focados na validação de funcionalidades do sistema a partir da perspectiva do usuário final, usando a ferramenta TestLink para gerenciar os cenários e resultados dos testes manuais, utilizando abordagens de caixa preta.

Essas abordagens combinadas garantirão que o sistema seja testado de maneira abrangente, identificando defeitos em diferentes níveis e assegurando a qualidade e a confiabilidade do **Sistema de Gerenciamento de Assentos de Voo**.

## Triagem de Erros

### 2.3.1 Tabela de Definição de Prioridade de Erros

A seguir está uma tabela para definir a **priorização de erros** de acordo com a gravidade, impacto no sistema e nas funcionalidades do projeto. Ela serve para guiar a equipe na triagem de erros, ajudando a determinar a urgência e a abordagem para cada tipo de falha.

| **Critério** | **Crítico (1)** | **Alta Prioridade (2)** | **Média Prioridade (3)** | **Baixa Prioridade (4)** |
| --- | --- | --- | --- | --- |
| Impacto no funcionamento principal | Impede o funcionamento de funções essenciais do sistema (ex: compra de assento, processamento de pagamento). | Afeta funções importantes, mas não impede a operação total (ex: visualização de itinerário). | Afeta a experiência do usuário, mas não impede o funcionamento (ex: erro de usabilidade, lentidão). | Não afeta o funcionamento principal, apenas a estética ou conveniência (ex: problemas gráficos menores). |
| Urgência para correção | Correção deve ser imediata, pois compromete a integridade e operação do sistema. | Correção rápida necessária, mas não impede o sistema de operar de maneira geral. | Correção deve ocorrer, mas não é urgente. Pode ser corrigido em ciclos de desenvolvimento futuros. | Correção pode ser planejada para o futuro, sem impacto significativo no uso imediato. |
| Impacto no Usuário/Cliente | Usuários não conseguem realizar a principal função do sistema (compra de assento), causando frustração e perda de vendas. | Usuário pode continuar a navegar no sistema, mas tem dificuldades ou falta de funcionalidades para concluir a tarefa. | Usuário pode completar a tarefa, mas com uma experiência reduzida ou inconveniente. | Usuário pode completar a tarefa sem problemas, mas a experiência visual é afetada. |

A seguir a priorização de erros para o **Sistema de Gerenciamento de Assentos de Voo**, cobrindo tanto os requisitos funcionais quanto os não funcionais. O objetivo é categorizar os erros encontrados durante os testes, de acordo com a gravidade e impacto no sistema, para garantir que as falhas mais críticas sejam corrigidas primeiro, enquanto as menores são tratadas conforme a disponibilidade de tempo e recursos.

### 2.3.2 Tabela de Priorização de Erros

| **Erro Identificado** | **Classificação** | **Tipo de Resolução** | **Impacto no Sistema** | **Cronograma de Correção** |
| --- | --- | --- | --- | --- |
| Admin não consegue adicionar preços de assentos. | Crítico | Correção imediata | Impede a administração de preços de assentos, impactando diretamente na gestão do voo. | Corrigido imediatamente (prioridade alta). |
| Sistema de pagamento falha ao concluir a transação. | Crítico | Correção imediata | Impede que o cliente não efetue a compra, impactando diretamente na reserva do assento. | Prioridade máxima - A ser corrigido no próximo ciclo de desenvolvimento |
| O cliente não consegue pesquisar as viagens disponíveis. | Crítico | Correção imediata | Retorna uma lista de voos incorreta para o cliente com base nos filtros do mesmo. | Corrigido imediatamente (prioridade alta). |
| Erro no processamento do pagamento, a página não exibe o preço total corretamente. | Crítico | Correção imediata | Afeta diretamente a conclusão da compra, impactando as transações financeiras. | Resolvido imediatamente (prioridade máxima). |
| O sistema permite comprar mais assentos do que a capacidade disponível. | Crítico | Correção imediata | Viola o limite de assentos disponíveis e compromete o controle de inventário de assentos. | Correção imediata, validação de estoque. |
| O sistema falha em enviar o e-mail de confirmação após a compra. | Alta Prioridade | Correção imediata | Afeta a experiência do cliente e compromete a confirmação da compra, causando insatisfação. | Correção não imediata, depende do concerto da função de pagamento. |

## Critérios de Suspensão e Requisitos de Retomada

A **suspensão** dos testes deve ocorrer sempre que uma falha crítica ou à performance do sistema for identificado. A **retomada** dos testes ocorrerá quando as causas da suspensão forem corrigidas e verificadas pela equipe de testes. Esses critérios garantem que o processo de teste seja realizado de forma controlada, eficiente e sem comprometer a qualidade do sistema.

## Completude do Teste

* A taxa de **100%** de **execução** dos testes é **obrigatória**, a menos que um motivo claro seja fornecido.
* A taxa de **80%** de **aprovação** dos testes é **obrigatória**.
* A cobertura dos testes dos requisitos listados no escopo deve ser igual a **100% obrigatoriamente**.

## Atividades do projeto, estimativas e cronograma

### 2.6.1 Atividades do Projeto

#### 1. Planejamento de Testes

**Objetivo**: Definir a abordagem de teste e garantir que todos os aspectos do sistema sejam cobertos.

* **Definição de Estratégia de Teste**: Escolher os tipos de testes que serão realizados (testes unitários, de integração, de sistema, de aceitação, etc.).
* **Planejamento de Recursos**: Identificar as ferramentas de teste necessárias, como **JUnit** para testes unitários, **TestLink** para testes manuais, e ferramentas de automação (se aplicável).
* **Criação de Casos de Teste**: Desenvolver casos de teste baseados nos **requisitos funcionais e não funcionais**. Isso inclui testes para os principais fluxos do sistema, como:
  + Compra de assentos
  + Aprovação de alterações de preços
  + Geração de itinerários
  + Processamento de pagamento
  + Controle de estoque de assentos
* **Definição de Critérios de Suspensão e Retomada**: Estabelecer os critérios para suspender ou retomar os testes, caso ocorram falhas graves.

**Resultado Esperado**: Documentação completa com a estratégia de testes, recursos, e casos de teste definidos.

#### 2. Teste Manuais

**Objetivo**: Planejar e preparar todos os recursos necessários para a execução dos testes manuais.

* **Definir Casos de Teste**: Com base nos **requisitos funcionais e não funcionais**, criar casos de teste manuais detalhados que cubram todas as funcionalidades críticas do sistema, como:
  + Cadastro e login de administradores e clientes.
  + Seleção e reserva de assentos.
  + Processamento de pagamentos.
  + Geração e envio de itinerários por e-mail.
  + Aprovação de alterações de preços e assentos pelo gerente.
  + Exibição de preços e disponibilidade de assentos.
  + Funcionalidade de envio de e-mails após a compra.
* **Preparar Dados de Teste**: Criar dados de teste representativos para cada cenário, incluindo diferentes usuários (admin, gerente, cliente), informações de voo, preços e itinerários.
* **Configurar o Ambiente de Teste**: Preparar o ambiente para testes manuais, incluindo a instalação de ferramentas como **TestLink** para rastrear os testes, configuração dos navegadores e dispositivos de teste (se necessário).

**Resultado Esperado**: Casos de teste bem definidos, dados de teste configurados e ambiente pronto para execução.

#### 3. Testes Unitários

**Objetivo**: Garantir que cada módulo ou componente do sistema funcione de forma independente e conforme esperado.

* **Testar Funções e Métodos Individuais**: Usando o **JUnit**, realizar testes em funções e métodos do código para verificar se cada componente individual funciona corretamente.
* **Cobertura de Código**: Garantir que todos os caminhos de código e cenários sejam testados, incluindo condições de erro e exceções.
* **Testar a Lógica de Cálculo**: Verificar os cálculos de preços, disponibilidade de assentos e outras lógicas de negócios.

**Resultado Esperado**: Todas as funções e métodos do sistema são validados individualmente, sem falhas críticas.

#### 4. Testes de Integração

**Objetivo**: Validar a interação entre diferentes módulos ou componentes do sistema para garantir que trabalhem juntos corretamente.

* **Testar a Comunicação Entre Módulos**: Verificar se a comunicação entre os módulos (como os módulos de pagamento, reservas e gerenciamento de assentos) ocorre corretamente e sem erros.
* **Verificar Fluxos de Dados**: Garantir que os dados fluem corretamente entre o banco de dados, a interface do usuário e os serviços backend.
* **Testar Funcionalidades Interdependentes**: Testar funcionalidades que dependem de outras, como a aprovação de preços pelo gerente antes de disponibilizar a compra para o cliente.

**Resultado Esperado**: Os módulos interagem de maneira correta, sem falhas ou perda de dados entre as transações.

#### 5. Relatórios de Teste e Análise de Resultados

**Objetivo**: Documentar os resultados dos testes e garantir que todos os erros identificados sejam corrigidos.

* **Gerar Relatórios de Testes**: Documentar os testes executados, os resultados obtidos e os defeitos encontrados.
* **Analisar Defeitos**: Analisar os defeitos encontrados para garantir que sejam corrigidos antes do lançamento.
* **Revisão e Aprovação**: Os testes devem ser revisados por líderes de QA e aprovados antes do lançamento final.

**Resultado Esperado**: Relatórios completos de teste, com defeitos corrigidos e validação final de qualidade.

### 2.6.2 Estimativas

| **Tarefa** | **Membros** | **Estimativa de esforço** |
| --- | --- | --- |
| Criar a especificação de testes | Projetista de teste | 170 homens/hora |
| Executar os testes | Testador, Administrador de teste | 80 homens/hora |
| Criar o relatório de testes | Testador | 10 homens/hora |
| Entregar os testes |  | 20 homens/hora |
| **Total** | | 280 homens/hora |

### 2.6.3 Cronograma

A seguir, apresento o cronograma de testes revisado, com a **entrega final prevista para 29/01/2025**, considerando todas as atividades de teste, incluindo planejamento, execução e validação final. O cronograma leva em conta que até o dia **09/11/2024** já foi realizado o **planejamento de testes**, a **criação de alguns casos de teste**, e a execução de **testes unitários e manuais**. A partir dessa data, o foco será na conclusão das demais fases de teste.

| **Atividade** | **Início** | **Término** | **Status** | **Comentários** |
| --- | --- | --- | --- | --- |
| 1. Planejamento de Testes | 01/11/2024 | 09/11/2024 | Concluído | Estratégia de testes e casos de teste iniciais definidos. |
| 2. Testes Manuais - Definir Casos de Teste | 09/11/2024 | 14/11/2024 | Em andamento | Definição de casos de teste manuais detalhados. |
| 3. Testes Manuais - Preparar Dados e Ambiente | 15/11/2024 | 18/11/2024 | Planejado | Preparação de dados de teste e configuração de ferramentas. |
| 4. Testes Unitários (JUnit) | 09/11/2024 | 15/11/2024 | Em andamento | Testes unitários em funções e cálculos de preços. |
| 5. Testes de Integração | 16/11/2024 | 22/11/2024 | Planejado | Testar interação entre módulos (reservas, pagamentos, etc.). |
| 6. Testes Manuais - Execução de Casos de Teste | 23/11/2024 | 30/11/2024 | Planejado | Execução dos testes manuais com base nos casos definidos. |
| 7. Testes de Aceitação (Validação com Stakeholders) | 01/12/2024 | 05/12/2024 | Planejado | Validação das funcionalidades pelos stakeholders (admin, cliente). |
| 8. Relatórios de Teste e Análise de Resultados | 06/12/2024 | 12/12/2024 | Planejado | Documentação de resultados, análise de defeitos e correções. |
| 9. Revisão de Testes e Aprovação Final | 13/12/2024 | 16/12/2024 | Planejado | Revisão final dos testes e aprovação para lançamento. |
| 10. Testes de Regressão | 17/12/2024 | 21/12/2024 | Planejado | Testar se as correções e atualizações não impactaram o sistema. |
| 11. Testes de Performance Manual | 22/12/2024 | 24/12/2024 | Planejado | Testes de carga e performance com múltiplos usuários. |
| 12. Relatório Final de Teste | 25/12/2024 | 29/12/2024 | Planejado | Elaboração do relatório final de testes com status de qualidade. |
| 13. Testes Pós-Correção e Últimos Ajustes | 02/01/2025 | 08/01/2025 | Planejado | Execução de testes após correção de bugs e ajustes finais. |
| 14. Revisão Final e Preparação para a Entrega | 09/01/2025 | 12/01/2025 | Planejado | Revisão dos testes finais, preparação para entrega e feedback. |
| 15. Entrega Final | 13/01/2025 | 29/01/2025 | Planejado | Entrega do sistema validado e aprovado para produção. |

# Entregáveis de Teste

Os entregáveis de teste são fornecidos conforme abaixo

**Antes da fase de teste**

* Documento de planos de teste.
* Documentos de casos de teste
* Especificações de projeto de teste.

**Durante o teste**

* Simuladores de ferramentas de teste.
* Dados de teste.
* Matriz de rastreabilidade de testes
* Logs de erros e logs de execução.

**Após o término dos ciclos de teste**

* Resultados de teste / relatórios
* Relatório de erros
* Diretrizes para instalação e para procedimentos de teste
* Notas de lançamento

# Necessidades de Recursos e Ambiente

## **Ferramentas de Teste**

Lista de ferramentas necessárias para testar o projeto:

* JUnit 5
* JUnit 4
* TestLink
* Mockito

## Ambiente de Teste

Lista de softwaresnecessários para a execução dos testes:

* Github
* Apache NetBeans 14
* Apache Tomcat 8.5.86
* JDK 8

## Repositórios

Link do repositório (Fork): <https://github.com/heronlancellot/Flight-Booking-System-JavaServlets_App-UFF.git>

Link do repositório do Commit Inicial:

<https://github.com/heronlancellot/Flight-Booking-System-JavaServlets_App-UFF/commit/f952bb4b37adcc70470d5b1e7c433ba4a47810d4>

Link do repositório do trabalho: <https://github.com/repo-software-testing-courses/Flight-Booking-System-JavaServlets_App>

# Termos / Acrônimos

| **TERMO / ACRÔNIMO** | **DEFINIÇÃO** |
| --- | --- |
| API | *Application Program Interface* |
| JUnit 5 | *Framework de Testes Automatizados* |
| JDK 8 | *Kit de desenvolvimento que inclui ferramentas essenciais para criar e compilar aplicativos Java* |
| Github | *Serviço responsável pelo controle de versão da aplicação* |

**6 ANÁLISE DE PRIORIDADE DAS CARACTERÍSITICAS DE QUALIDADE**

# 6.1 Funcionalidade

## Prioridade: Alta

## Justificativa:

A funcionalidade é a característica mais crítica para o sistema de reserva de passagens. O sistema deve ser capaz de realizar as operações principais de forma eficaz, incluindo a reserva de passagens, o processamento de pagamentos, a notificação por e-mail e a validação de dados. A ausência ou falha nessas funcionalidades essenciais comprometeria gravemente a experiência do usuário e a confiança no sistema. Como as funcionalidades são fundamentais para o objetivo do software, elas devem ser a principal prioridade no desenvolvimento.

# 6.2 Confiabilidade

## Prioridade: Alta

## Justificativa:

A confiabilidade também é uma característica crucial, especialmente em um sistema que envolve transações financeiras (pagamentos) e comunicação com os usuários (notificações por e-mail). A confiabilidade do sistema, em termos de sua capacidade de operar sem falhas, identificar exceções de maneira eficaz e garantir que transações sejam realizadas com segurança, é essencial para a credibilidade do serviço. Caso o sistema falhe durante a reserva de passagens ou no processamento de pagamentos, isso pode levar a perdas financeiras e frustração dos usuários.

## Subcaracterísticas Importantes:

* **Tratamento de exceções**: Garantir que exceções, especialmente em transações financeiras, sejam tratadas corretamente, sem impactar o usuário.
* **Logs de transações**: Garantir rastreabilidade e análise pós-falha para identificar problemas rapidamente.

# 6.3 Usabilidade

## Prioridade: Alta

## Justificativa:

A usabilidade é fundamental para garantir que o sistema seja acessível e intuitivo para os usuários finais. Em um sistema de reserva de passagens, a facilidade de uso impacta diretamente a experiência do cliente. Se o sistema for difícil de navegar, o usuário pode desistir da compra, afetando as taxas de conversão e a satisfação do cliente. A interface deve ser simples, amigável e clara, com mensagens de erro compreensíveis e suporte multilíngue para atender a uma audiência global.

## Subcaracterísticas Importantes:

* **Interface intuitiva**: Uma interface fácil de usar é fundamental para que qualquer usuário, independentemente de seu nível técnico, consiga realizar a reserva sem dificuldades.
* **Mensagens de erro claras**: Facilitar o entendimento sobre o que ocorreu de errado e como corrigir.

# 6.4 Eficiência

## Prioridade: Média

## Justificativa:

Embora a eficiência seja muito importante, ela pode ser considerada secundária em comparação com funcionalidade, confiabilidade e usabilidade para um sistema de reserva de passagens. No entanto, um sistema eficiente, que minimize o uso de recursos e otimize o tempo de resposta (como no caso de processamento assíncrono de emails ou otimização de consultas ao banco), oferece uma melhor experiência ao usuário. O tempo de resposta das transações (como a conclusão de uma reserva ou o envio de uma confirmação de pagamento) deve ser rápido para não gerar frustração.

## Subcaracterísticas Importantes:

* **Otimização de consultas**: Como as consultas ao banco de dados podem ser pesadas, otimizar essas operações é importante para garantir um desempenho rápido.
* **Processamento assíncrono**: A execução de tarefas como o envio de e-mails e atualizações de status deve ser feita de forma assíncrona para melhorar a resposta do sistema.

# 6.5 Manutenibilidade

## Prioridade: Média

## Justificativa:

A manutenção é importante para garantir que o sistema possa ser facilmente atualizado e mantido ao longo do tempo. No entanto, em um primeiro momento, a prioridade é garantir que o sistema atenda aos requisitos essenciais. No futuro, conforme o sistema cresce e novos recursos são adicionados (como novos métodos de pagamento ou integração com novos provedores de serviços), a facilidade de manutenção do sistema será crítica. A utilização de uma arquitetura modular e bem documentada facilitará futuras melhorias e correções de erros.

## Subcaracterísticas Importantes:

* **Arquitetura em camadas**: Facilita a manutenção e expansão do sistema sem impactar outras áreas.
* **Documentação clara no código**: Ajuda desenvolvedores a entender rapidamente a lógica do sistema para realizar ajustes ou correções.

# 6.6 Portabilidade

## Prioridade: Média

## Justificativa:

A portabilidade é relevante, mas sua prioridade depende do público-alvo e do ambiente operacional do sistema. Para um sistema de reserva de passagens que opera principalmente em servidores específicos ou plataformas de nuvem, a portabilidade pode ser menos crítica. No entanto, garantir que o sistema seja executado de forma consistente em diferentes plataformas e ambientes (como diferentes provedores de e-mail ou sistemas operacionais) ainda é importante para maximizar sua acessibilidade e reduzir custos de operação a longo prazo.

## Subcaracterísticas Importantes:

* **Independência de provedor de email**: A flexibilidade para utilizar diferentes provedores de e-mail sem comprometer o funcionamento do sistema.
* **Compatibilidade cross-platform**: A garantia de que o sistema funcione corretaments em diferentes plataformas e dispositivos.

# 6.7 Segurança

## Prioridade: Alta

## Justificativa:

Em sistemas que lidam com informações pessoais e financeiras, como um sistema de reserva de passagens, a segurança é uma prioridade essencial. Isso inclui a proteção de dados do usuário, a prevenção de fraudes no processo de pagamento e a garantia de que a comunicação entre o cliente e o servidor seja criptografada de forma segura. A segurança é vital para proteger tanto os usuários quanto a reputação do sistema.

# 7 **Projetar e melhorar o conjunto de casos de teste, utilizando as técnicas**

- Funcional

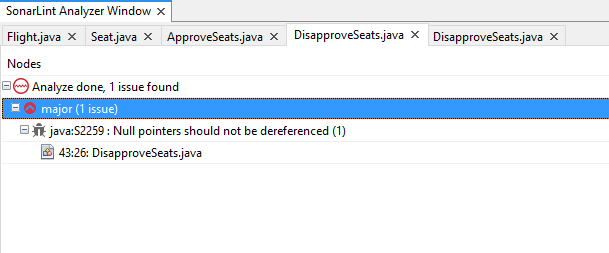
* Estrutural
* Baseada em defeitos

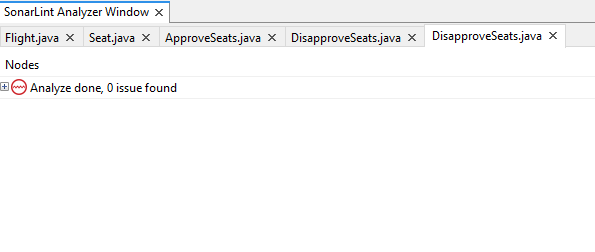
# 8 Relatório de Inspeção do Código Fonte:

[Aluno: Sávio Suhett]

1. Executar a ferramenta e enviar o print:

***DisapproveSeats.java***

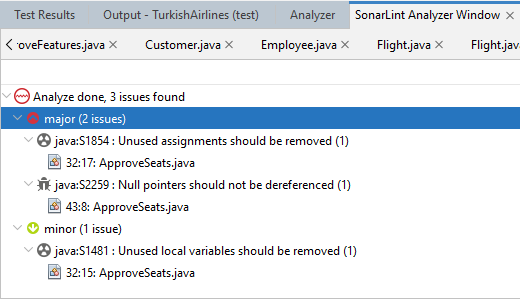
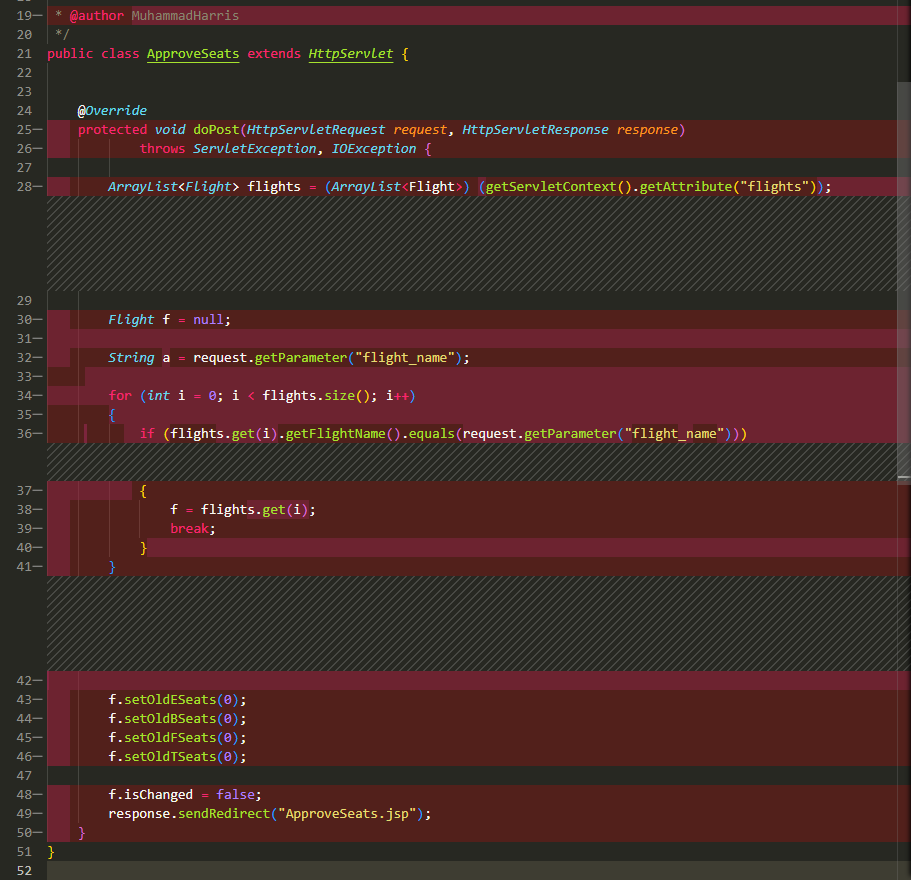
******

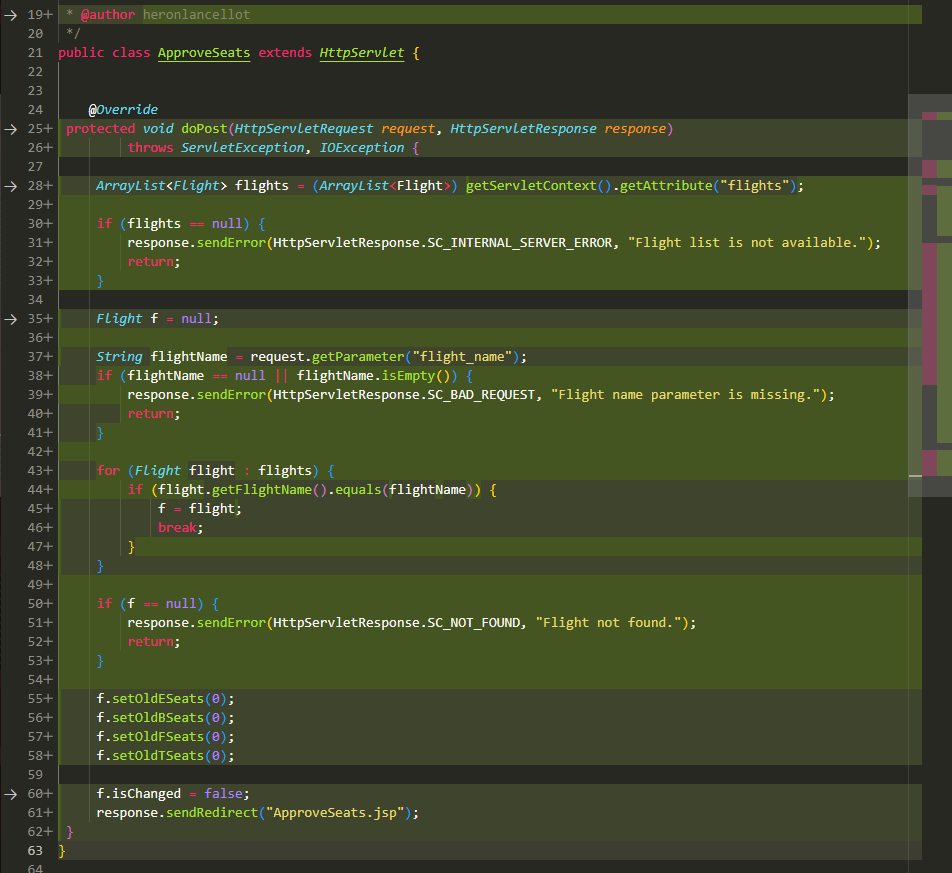
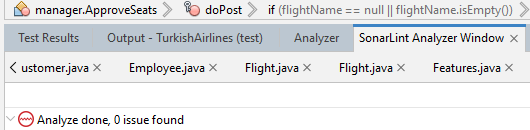
1. Resolução do problema:  
     
   
2. Print pós correções:  
     
   

[Aluno: Heron Lancellot]

1. Executar a ferramenta e enviar o print:

***ApproveSeats***

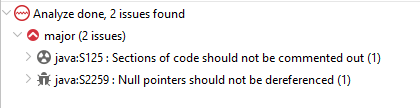
* 1. 
  2. 

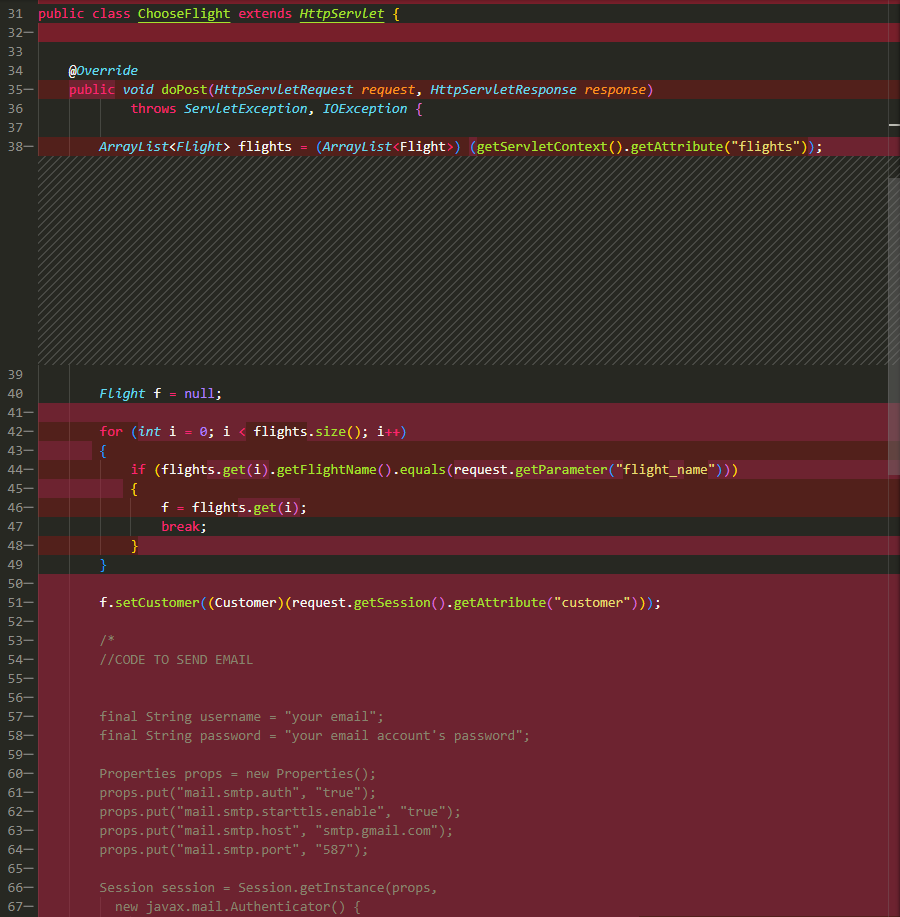
1. Resolução do problema
   1. 
2. Print pós correções:  
   

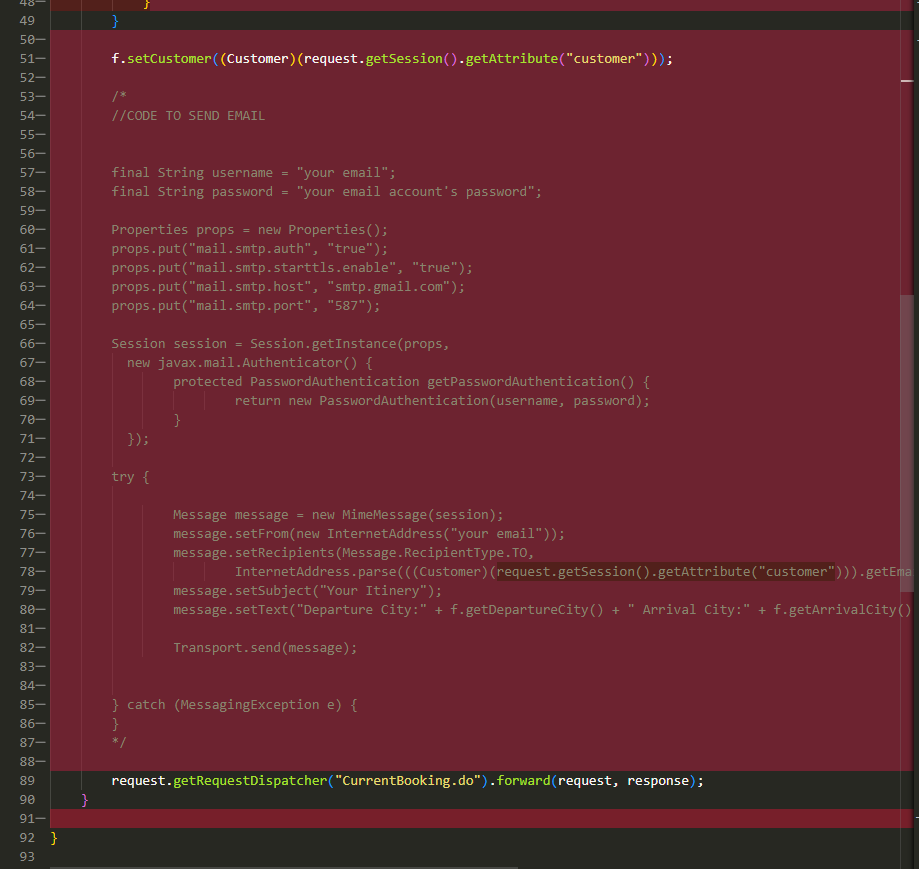
[Aluno: Eduardo Saldumbides]

1. Executar a ferramenta e enviar o print:

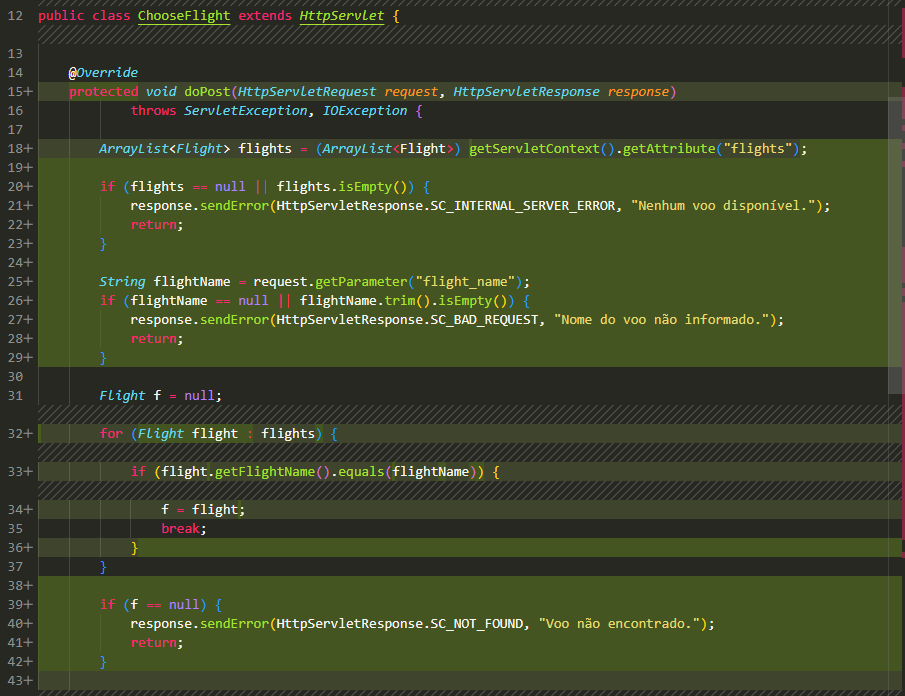
***ChooseFlight***



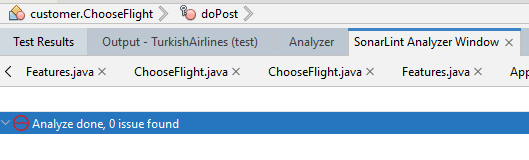




1. Resolução do problema



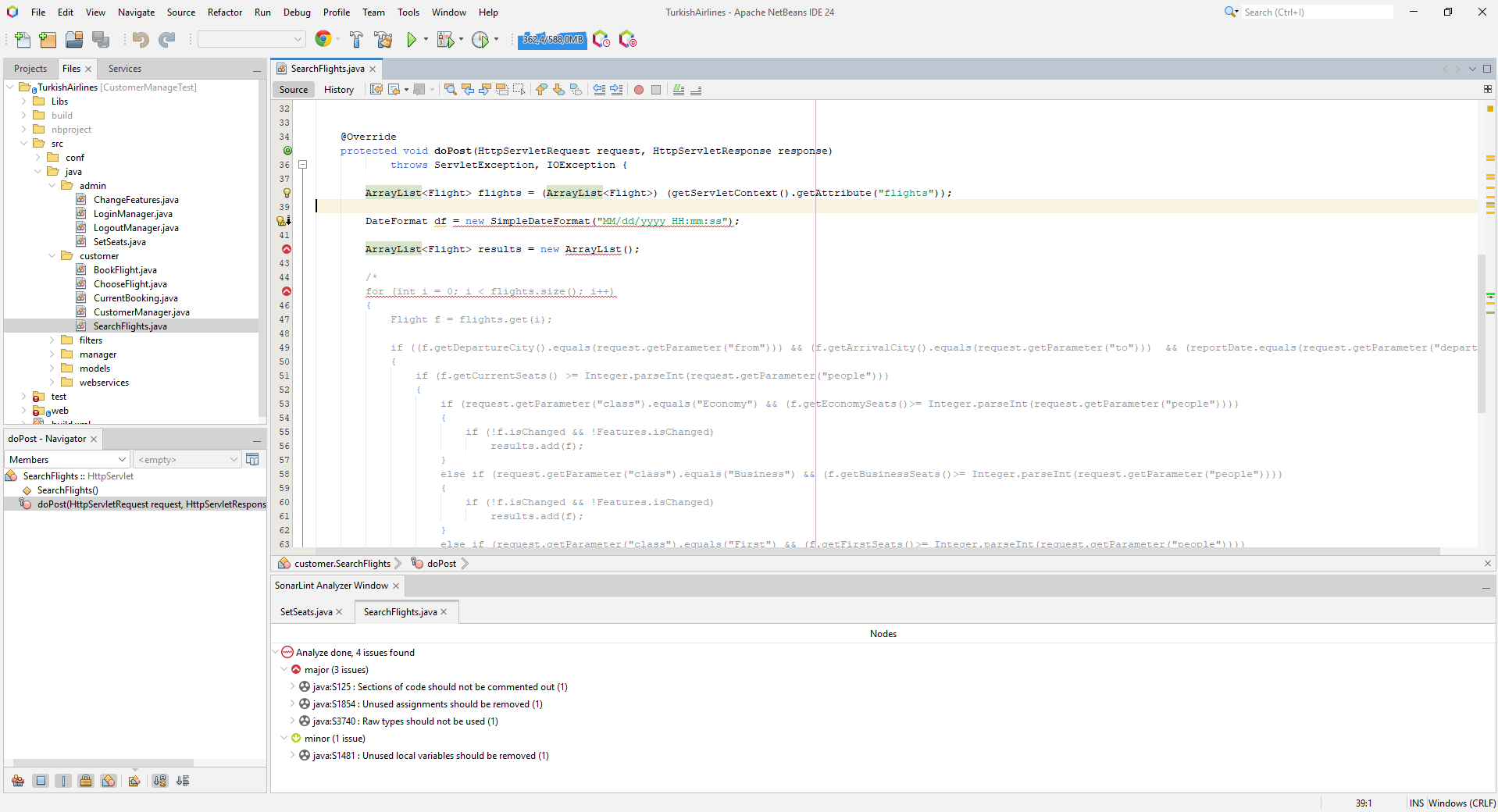
1. Print pós correções



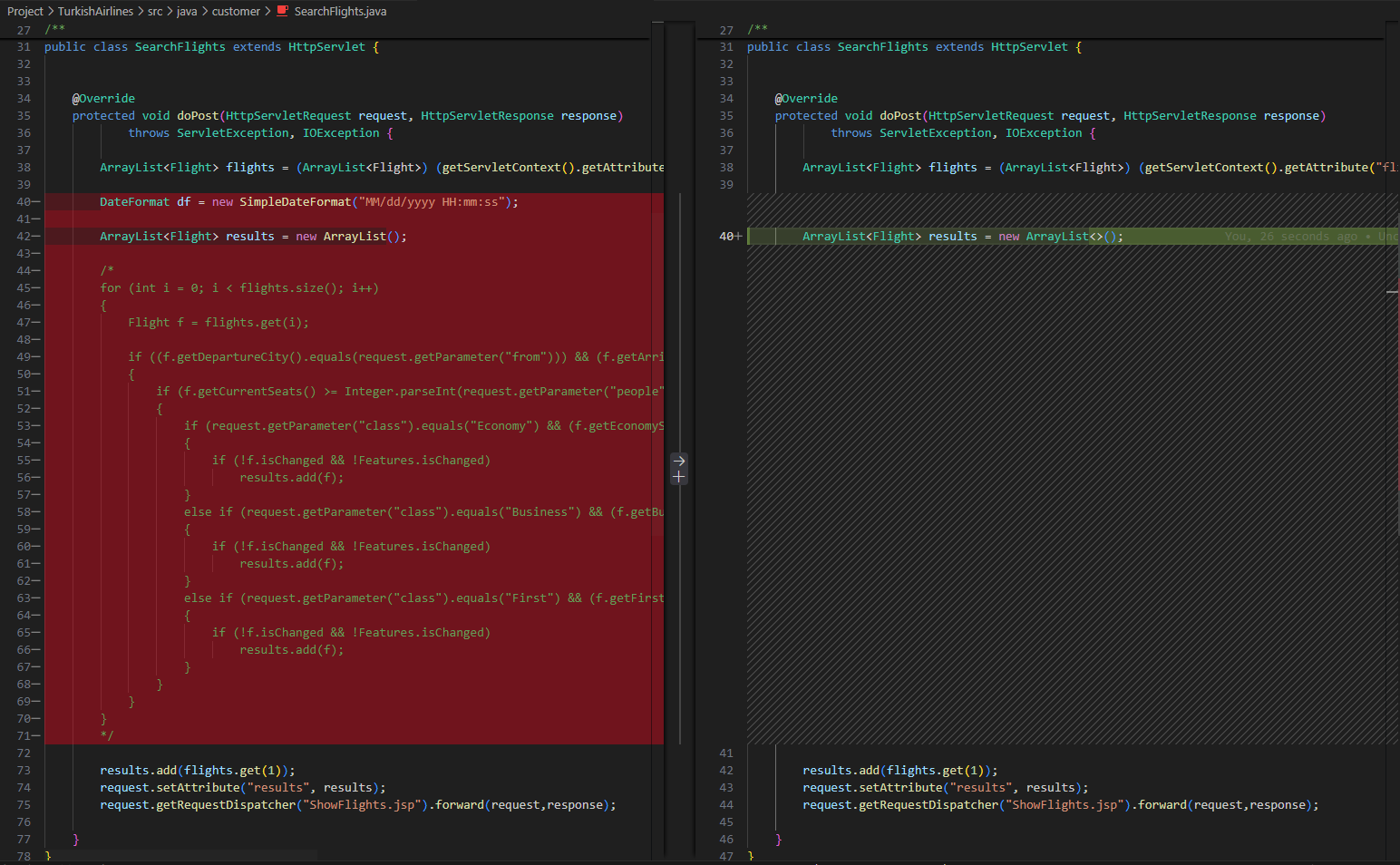
[Aluno: Diego Camargo]

1. Executar a ferramenta e enviar o print:

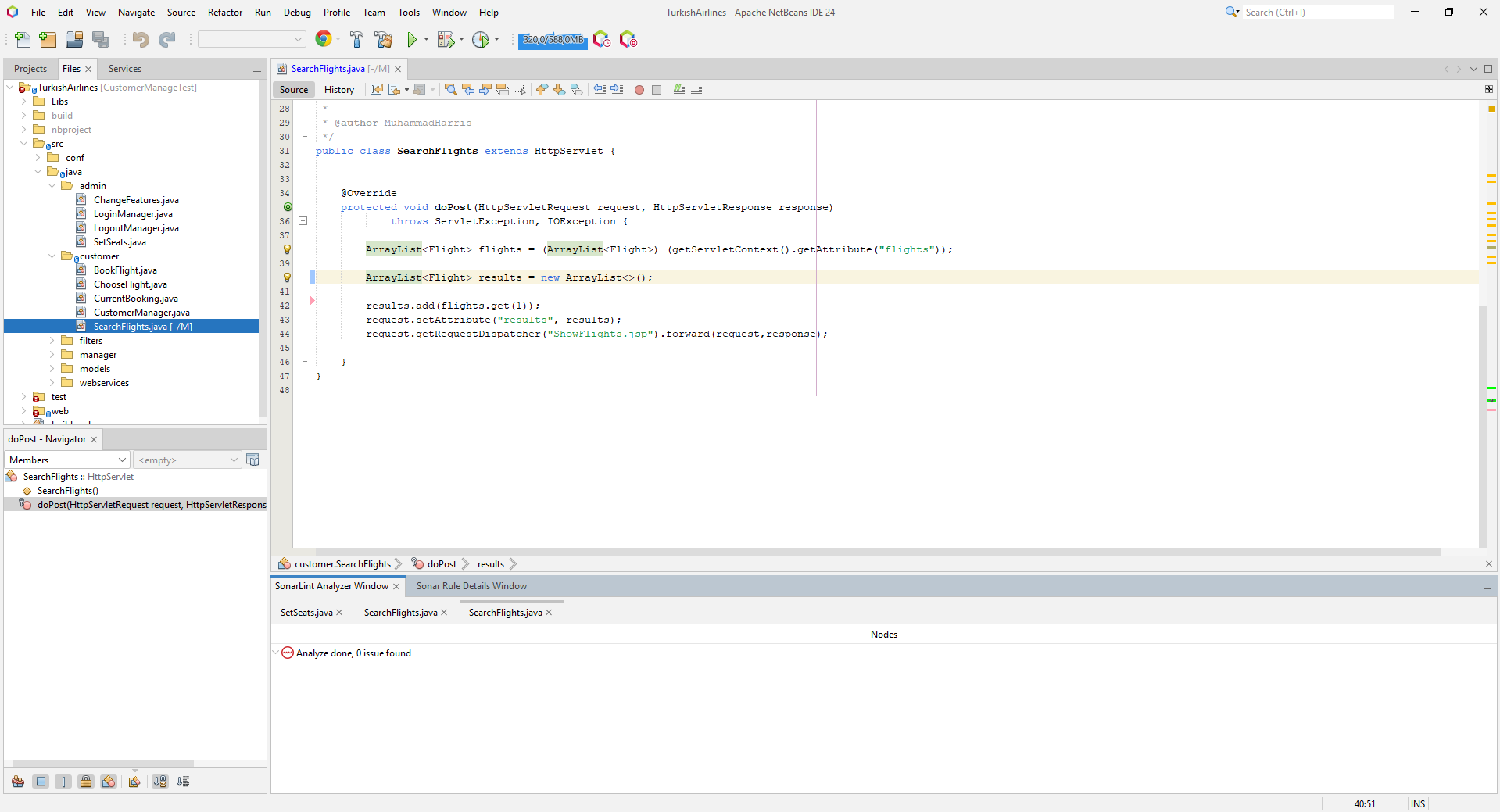
**SearchFlights**

****

1. Resolução do problema



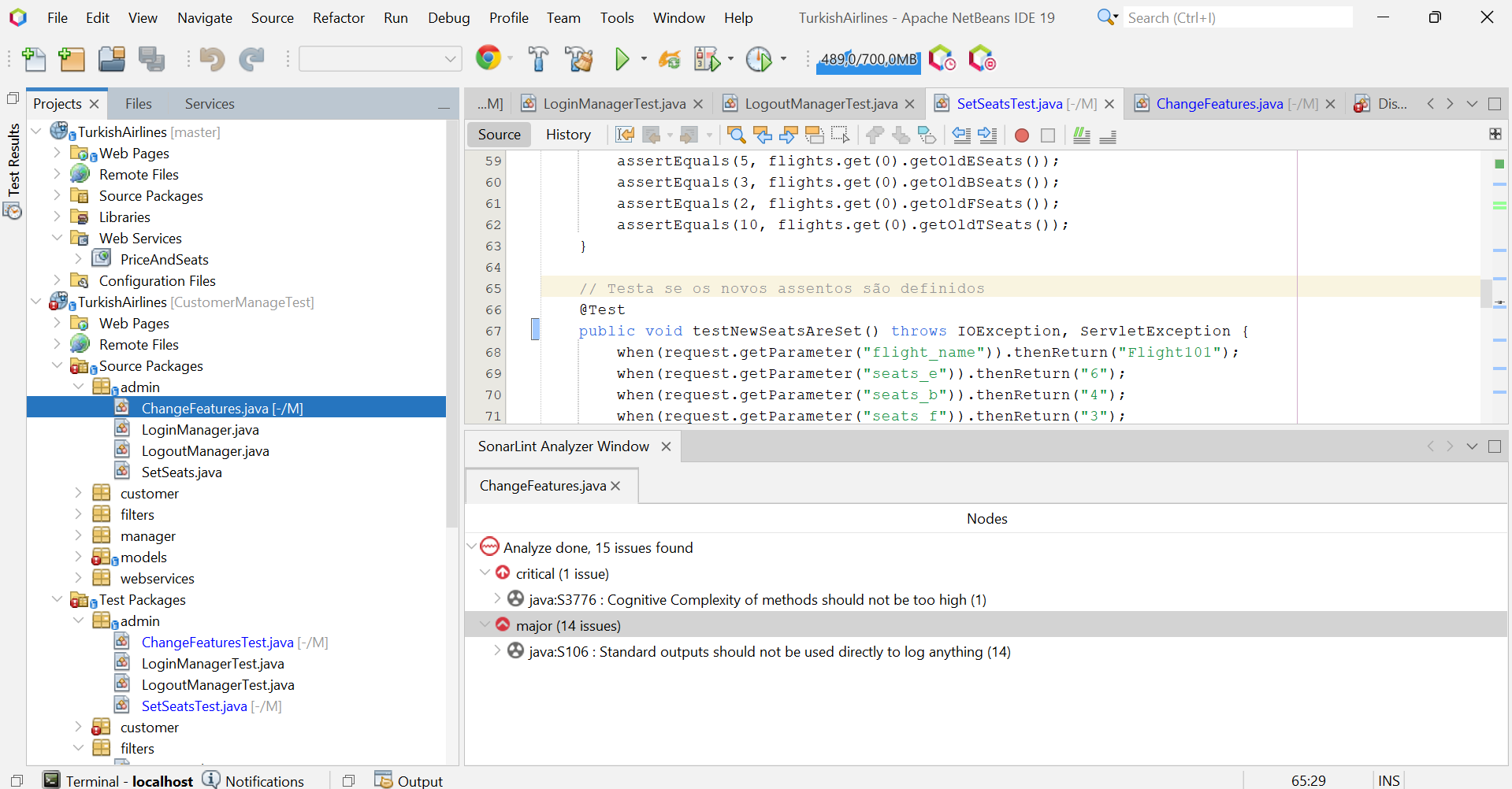
1. Print pós correções



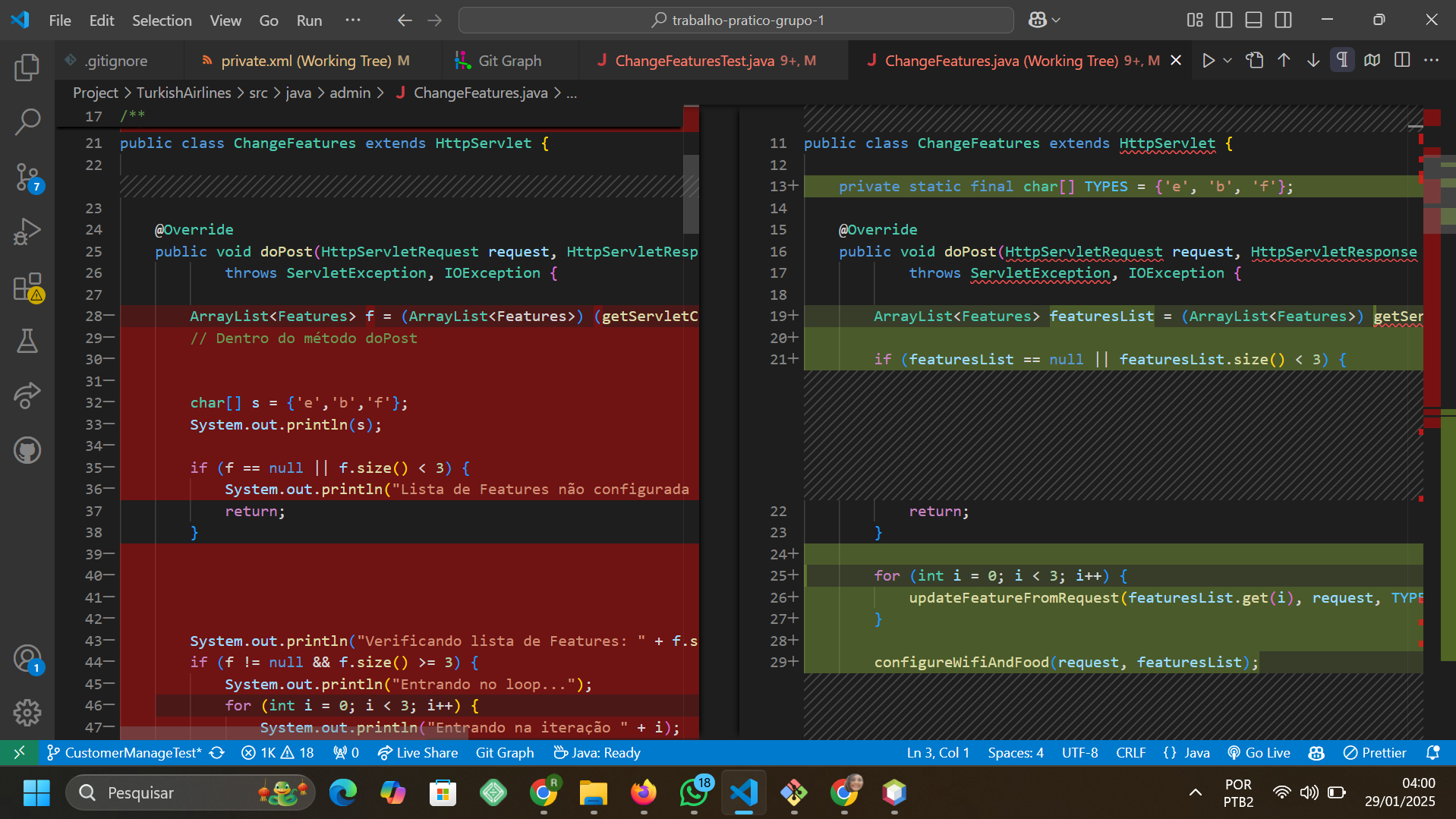
[Aluno: Rafaela Peçanha]

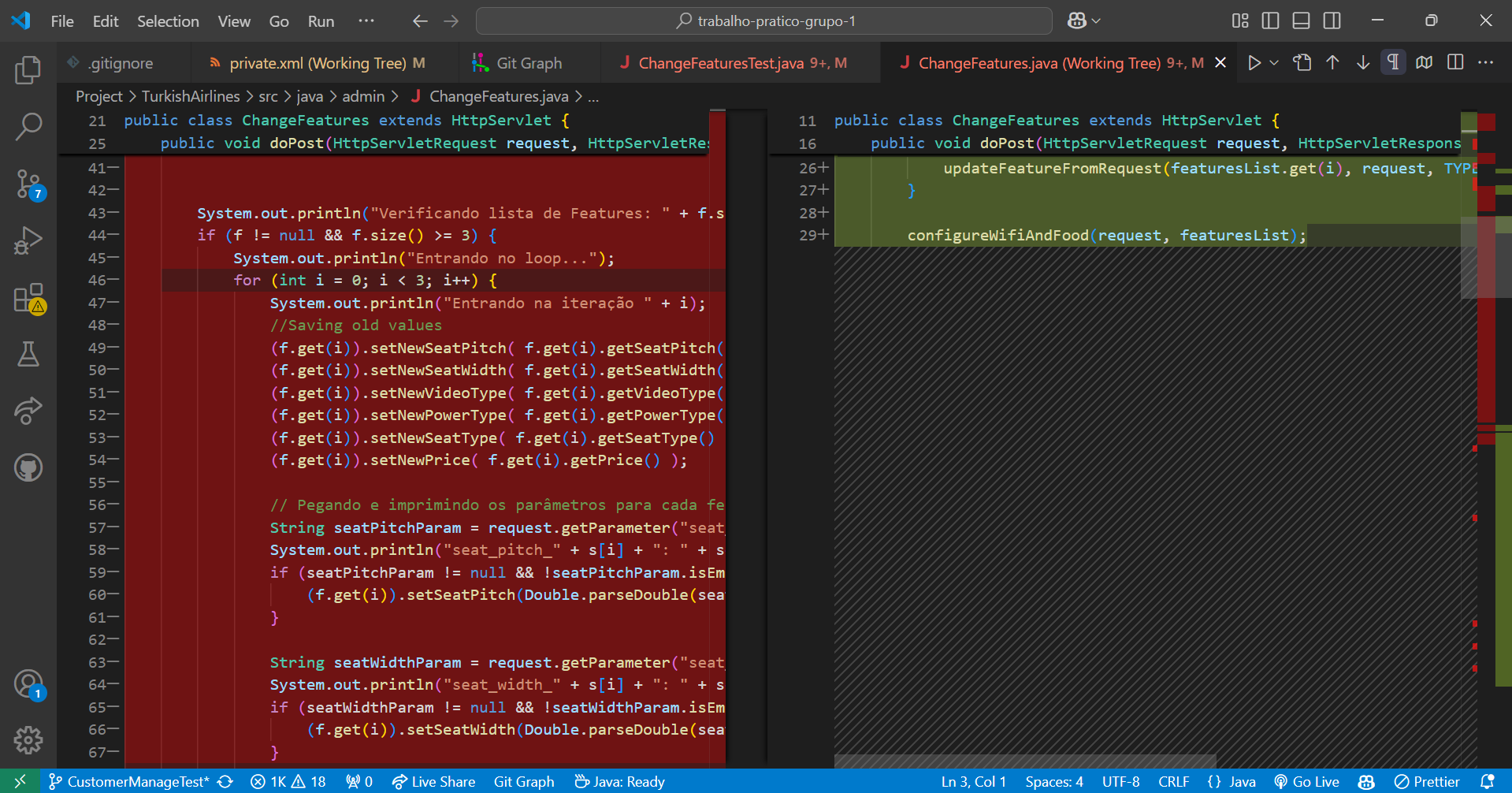
1. Executar a ferramenta e enviar o print:

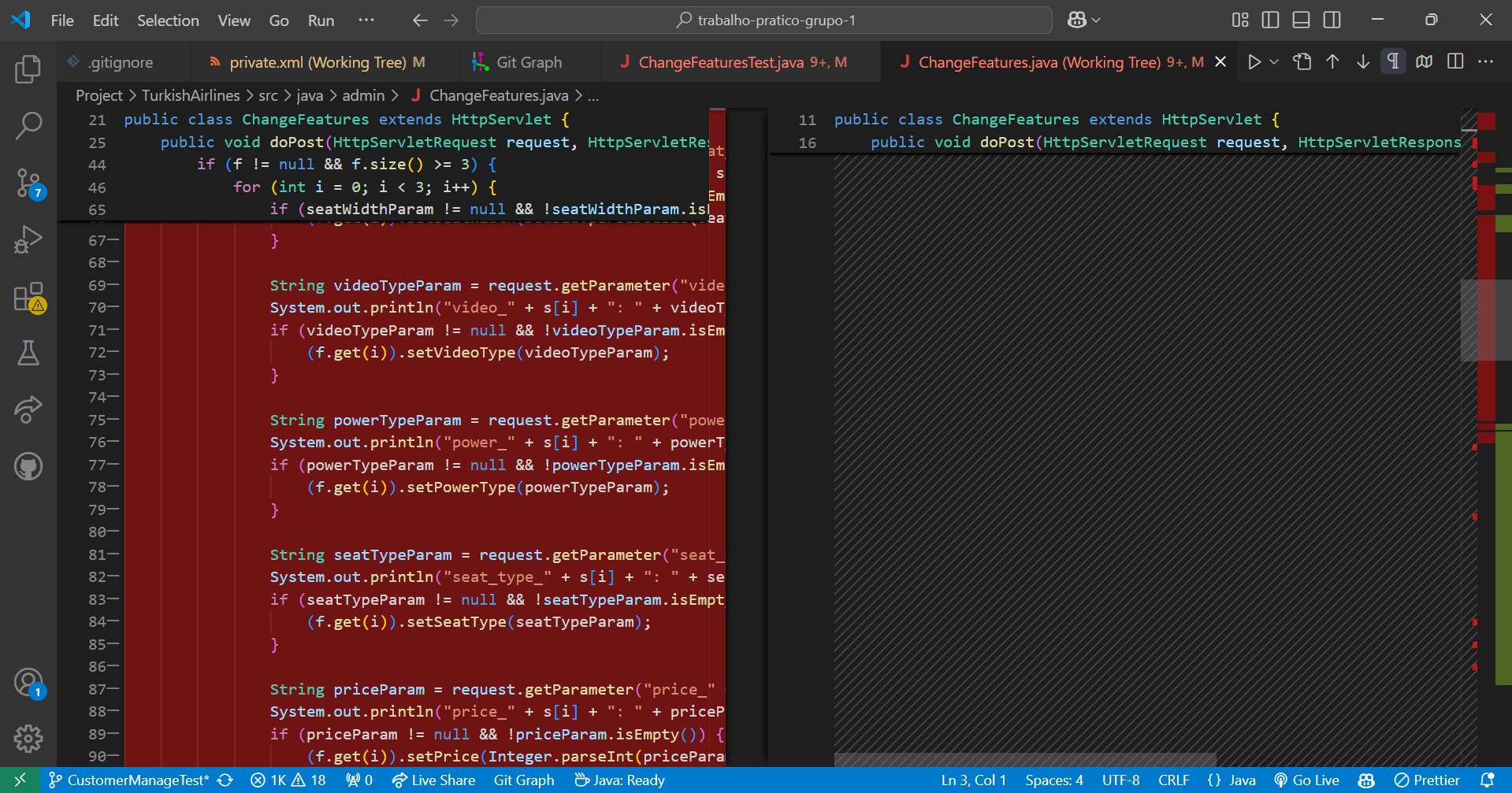
**ChangeFeatures**

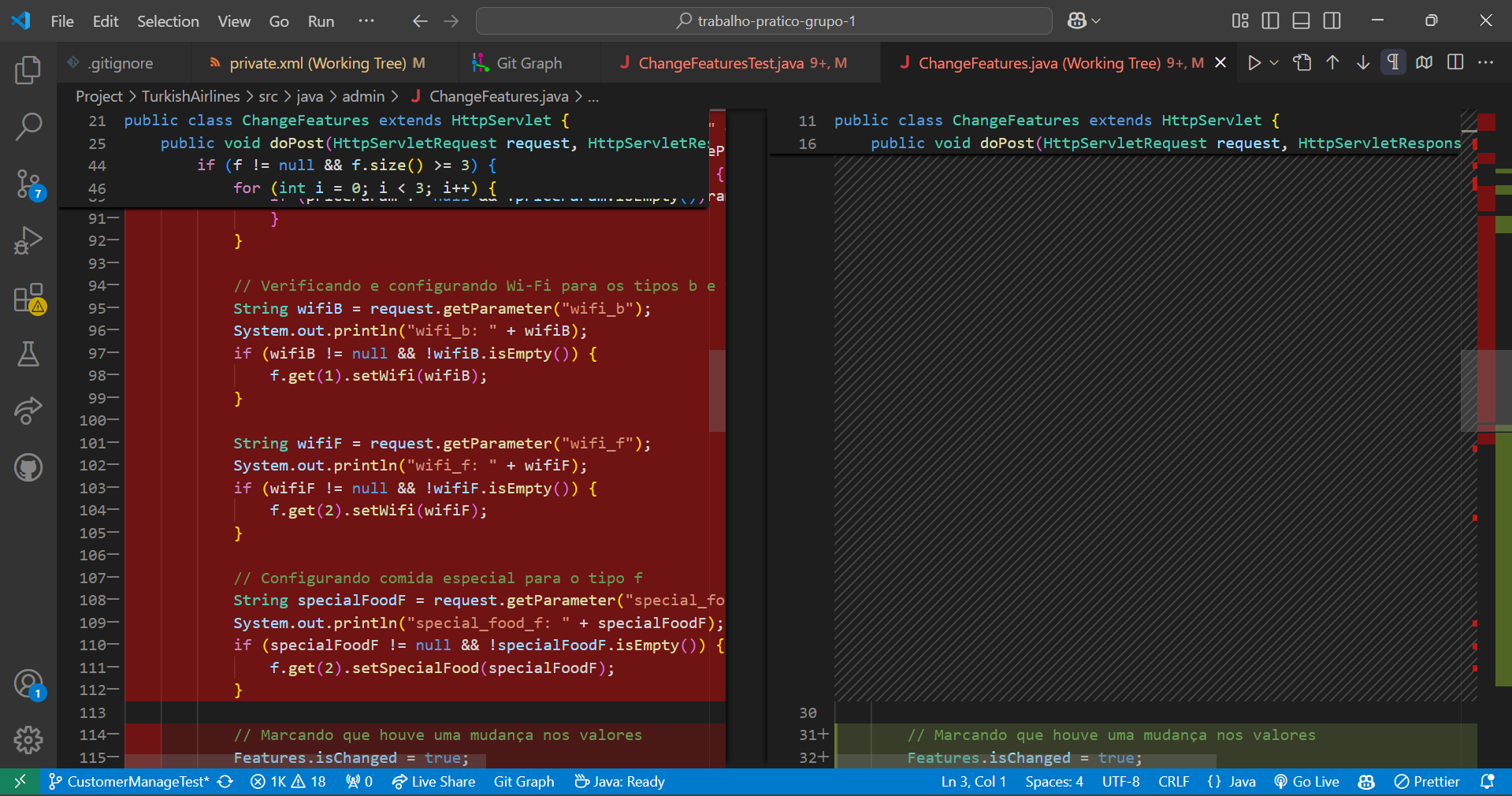


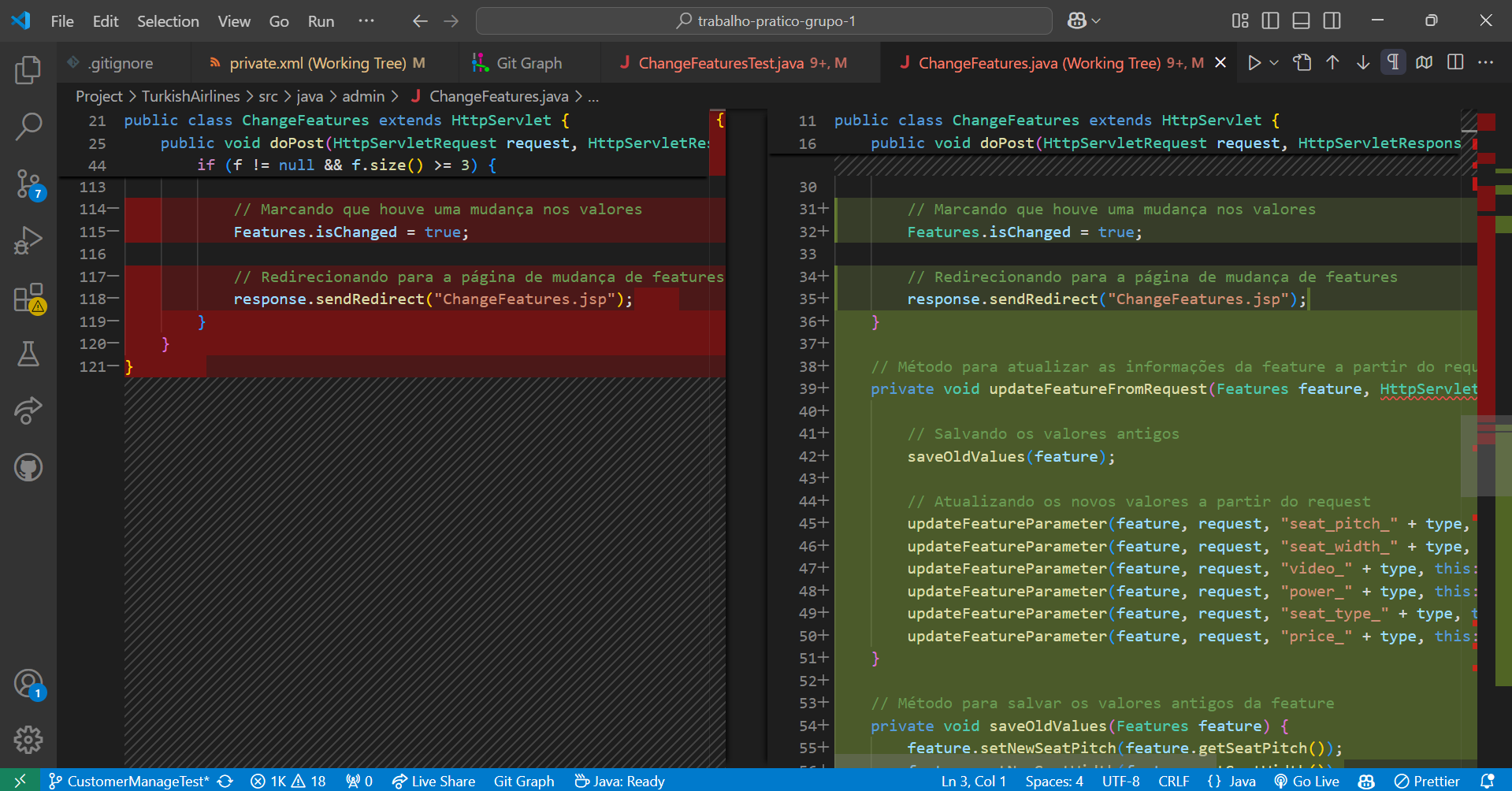
1. Resolução do problema

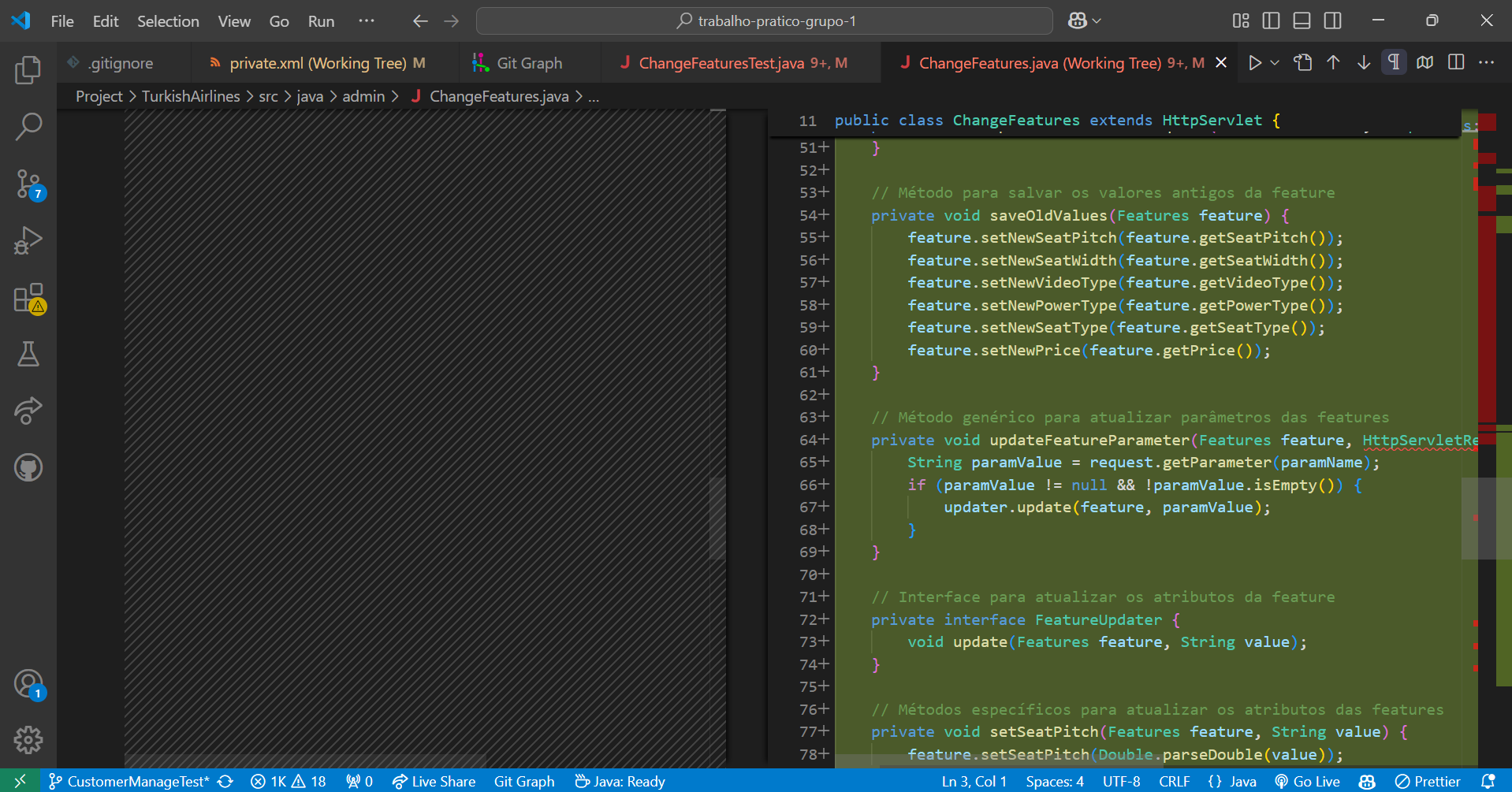


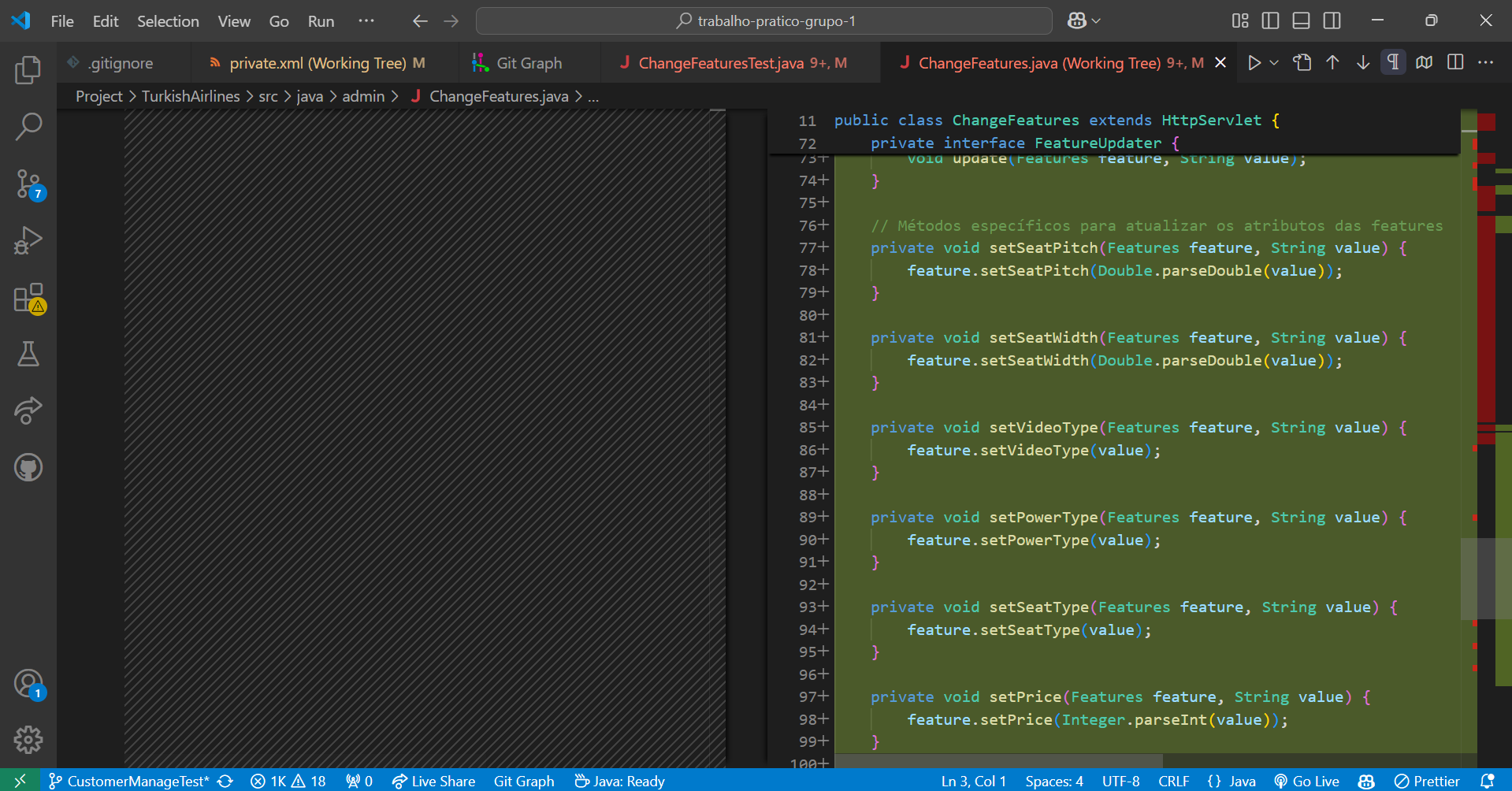


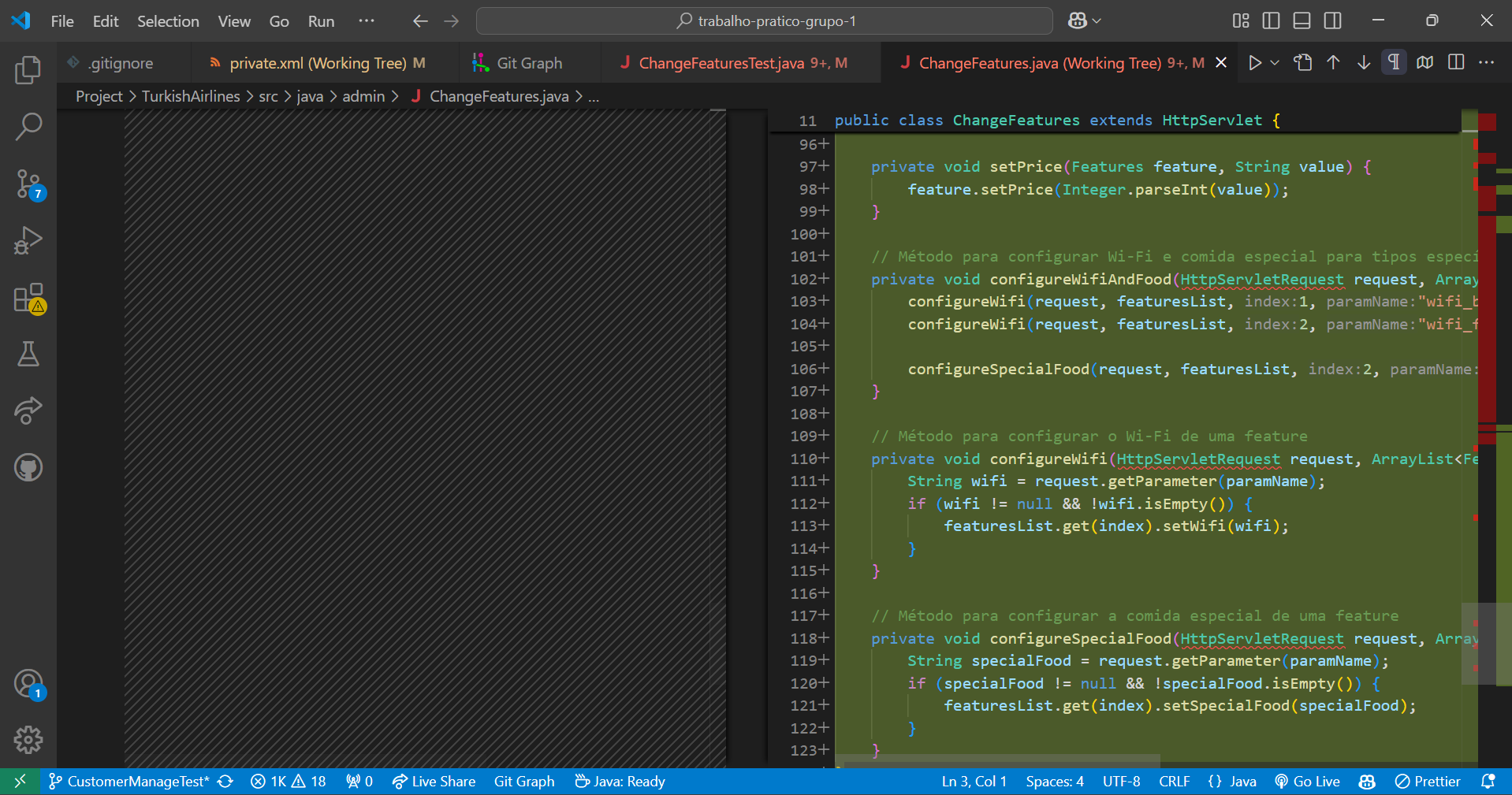




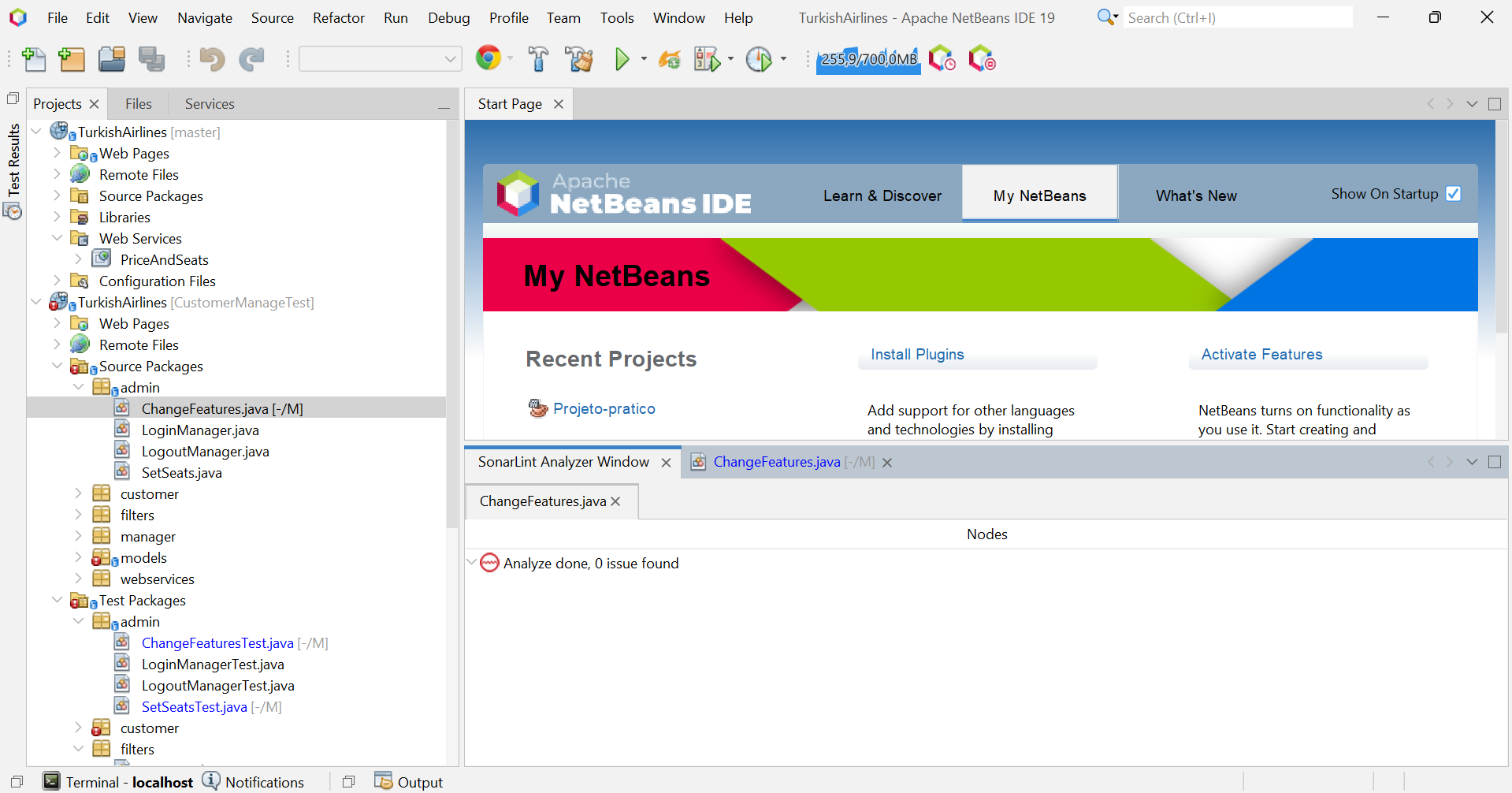






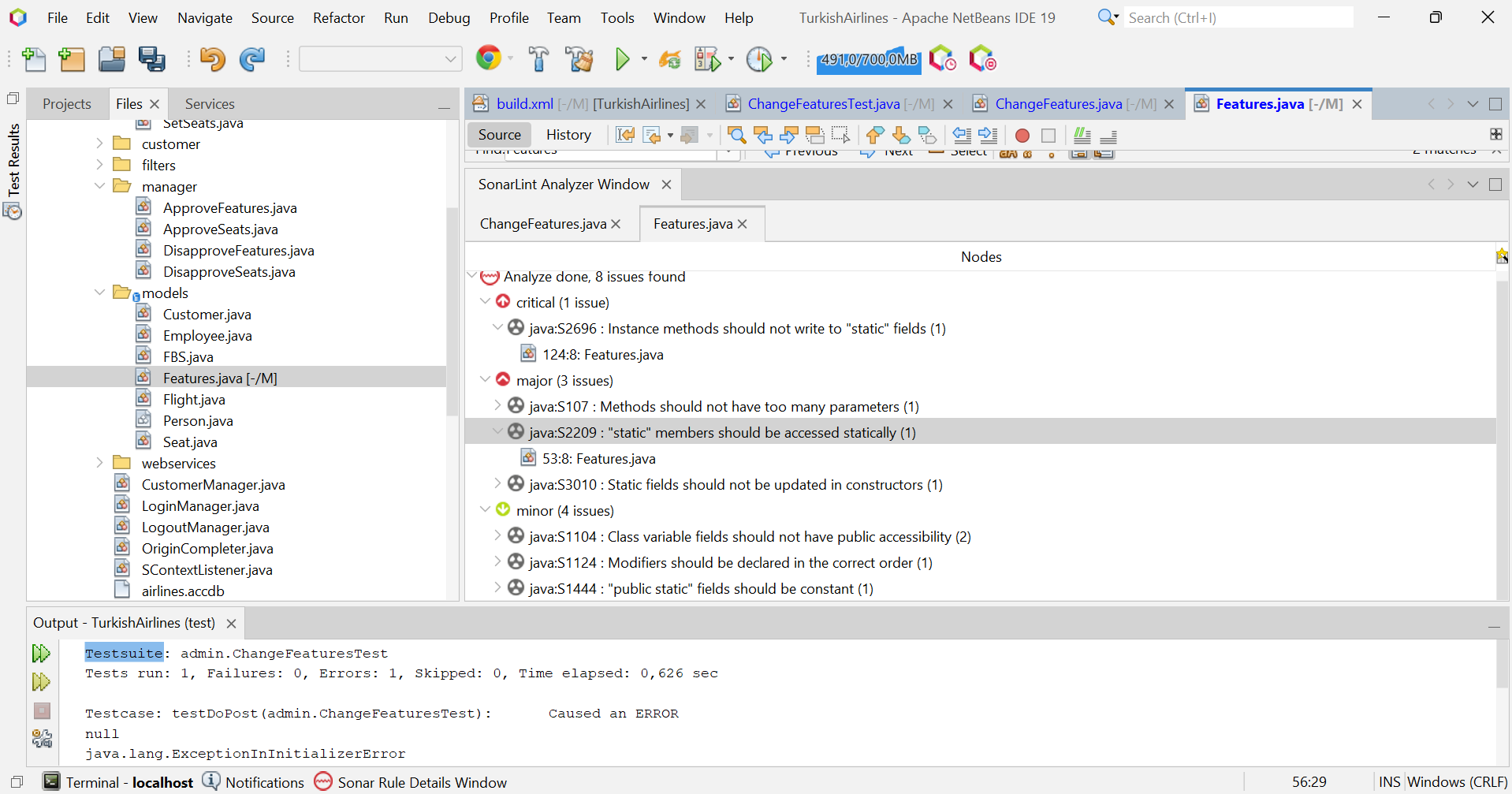


1. Print pós correções



**Feature**

1. Executar a ferramenta e enviar o print:



1. Resolução do problema
2. Print pós correções